Abstract

This manual documents Tungsten Clustering (for MySQL), a high performance, High Availability and Disaster Recovery for MySQL clustering.

This manual includes information for 6.1, up to and including 6.1.3.

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

Up to date builds of this document: Tungsten Clustering (for MySQL) 6.1 Manual [Online], Tungsten Clustering (for MySQL) 6.1 Manual [PDF]
## Table of Contents

Preface ................................................................................................................................. xx

1. Legal Notice ..................................................................................................................... xx

2. Conventions ...................................................................................................................... xx

3. Quickstart Guide ............................................................................................................. xxi

1. Introduction ....................................................................................................................... 22

   1.1. Tungsten Replicator ................................................................................................. 22
       1.1.1. Transaction History Log [THL] ....................................................................... 23

   1.2. Tungsten Manager .................................................................................................. 23

   1.3. Tungsten Connector ................................................................................................. 24

2. Deployment ....................................................................................................................... 26

   2.1. Host Types ............................................................................................................... 26
       2.1.1. Manager Hosts .................................................................................................. 26
       2.1.2. Connector [Router] Hosts ............................................................................... 26
       2.1.3. Replicator Hosts .............................................................................................. 27
       2.1.4. Witness Hosts .................................................................................................. 27

   2.2. Requirements .......................................................................................................... 29
       2.2.1. Operating Systems Support ............................................................................. 29
       2.2.2. Database Support ............................................................................................. 30
       2.2.3. RAM Requirements ......................................................................................... 30
       2.2.4. Disk Requirements ........................................................................................... 30
       2.2.5. Java Requirements ............................................................................................ 31
       2.2.6. Cloud Deployment Requirements .................................................................... 31
       2.2.7. Docker Support Policy ...................................................................................... 31
           2.2.7.1. Overview ..................................................................................................... 31
           2.2.7.2. Background ................................................................................................. 31
           2.2.7.3. Current State ............................................................................................... 32
           2.2.7.4. What to Expect in the Future ...................................................................... 32
           2.2.7.5. Summary ..................................................................................................... 32

   2.3. Deployment Sources ................................................................................................. 32
       2.3.1. Using the TAR/GZipped files ........................................................................... 33
       2.3.2. Using the RPM and DEB package files ............................................................. 33

   2.4. Common rpm Options During Deployment .............................................................. 34

   2.5. Best Practices .......................................................................................................... 34
       2.5.1. Best Practices: Deployment .............................................................................. 34
       2.5.2. Best Practices: Operations ............................................................................... 35
       2.5.3. Best Practices: Maintenance ............................................................................ 35

   2.6. Prepare Hosts .......................................................................................................... 35
       2.6.1. Prepare MySQL Hosts ...................................................................................... 35
       2.6.2. Prepare Connector Hosts .................................................................................. 36
       2.6.3. Prepare Active Witness Hosts .......................................................................... 36
       2.6.4. Deploy SSH Keys .............................................................................................. 36

   2.7. Deployment Security ................................................................................................. 37
       2.7.1. Enabling Security ............................................................................................. 37
           2.7.1.1. Enabling Security using the Staging Method ................................................. 38
           2.7.1.2. Enabling Security using the INI Method ...................................................... 38
       2.7.2. Disabling Security ............................................................................................ 39
       2.7.3. Creating Suitable Certificates .......................................................................... 39
       2.7.4. Installing from a Staging Host with Manually Generated Certificates ............. 40
       2.7.5. Installing via INI File with Manually Generated Certificates ........................... 40
       2.7.6. Installing via INI File with CA-Signed Certificates ......................................... 41
       2.7.7. Replacing the JGroups Certificate from a Staging Directory ......................... 41
       2.7.8. Replacing the TLS Certificate from a Staging Directory ................................ 41
       2.7.9. Removing JGroups Encryption from a Staging Directory .............................. 43
       2.7.10. Removing JGroups Encryption via INI File ................................................... 43
       2.7.11. Removing TLS Encryption from a Staging Directory ..................................... 43
       2.7.12. Removing TLS Encryption via INI File .......................................................... 43
       2.7.13. Configuring Connector SSL ............................................................................ 44
           2.7.13.1. Setup Environment and Paths .................................................................. 44
           2.7.13.2. Configuring SSL for MySQL Server ......................................................... 45
           2.7.13.3. Enable and Test SSL encryption from the Connector to the Database .... 46
           2.7.13.4. Test SSL encryption from the Application to the Database .................... 48
       2.7.14. Creating the Truststore and Keystore ................................................................. 49
           2.7.14.1. Creating Your Own Client and Server Certificates .................................. 49
           2.7.14.2. Creating a Custom Certificate and Getting it Signed ............................... 51
3. Deployment: MySQL Topologies

3.1. Deploying Standalone HA Clusters
3.1.1. Prepare: Standalone HA Cluster
3.1.2. Install: Standalone HA Cluster
3.1.3. Best Practices: Standalone HA Cluster

3.2. Deploying Composite Master/Slave Clustering
3.2.1. Prepare: Composite Master/Slave Cluster
3.2.2. Install: Composite Master/Slave Cluster
3.2.3. Best Practices: Composite Master/Slave Cluster

3.3. Deploying Multisite/Multimaster Clustering
3.3.1. Prepare: Multisite/Multimaster Clusters
3.3.2. Install: Multisite/Multimaster Clusters
3.3.3. Best Practices: Multisite/Multimaster Clusters
3.3.4. Configuring Startup on Boot
3.3.5. Resetting a single datasource
3.3.6. Resetting all dataservices
3.3.7. Provisioning during live operations
3.3.8. Adding a new Cluster/Dataservice
3.3.9. Enabling SSL for Replicators Only
3.3.10. Dataserver maintenance
3.3.10.1. Fixing Replication Errors

3.4. Deploying Composite Multimaster Clustering
3.4.1. Prepare: Composite Multimaster Clusters
3.4.2. Install: Composite Multimaster Clusters
3.4.3. Best Practices: Composite Multimaster Clusters
3.4.4. Configuring Startup on Boot
3.4.5. Resetting a single datasource
3.4.6. Resetting all dataservices
3.4.7. Dataserver maintenance
3.4.7.1. Fixing Replication Errors
3.4.8. Adding a Cluster to a Composite Multimaster Topology
3.4.8.1. Pre-Requisites
3.4.8.2. Backup and Restore
3.4.8.3. Update Existing Configuration
3.4.8.4. New Host Configuration
3.4.8.5. Install on new nodes
3.4.8.6. Update existing nodes
3.4.8.7. Start the new cluster
3.4.8.8. Validate and check

3.5. Deploying Tungsten Connector Only

3.6. Deploying Additional Datasources, Managers, or Connectors
3.6.1. Adding Datasources to an Existing Deployment
3.6.2. Adding Active Witnesses to an Existing Deployment
3.6.3. Adding Passive Witnesses to an Existing Deployment
3.6.4. Adding Connectors to an Existing Deployment
3.6.5. Adding a remote Composite Cluster
3.6.6. Converting from a single cluster to a composite cluster
3.6.6.1. Convert and add new nodes as a new service
3.6.6.2. Convert and move nodes to a new service

3.7. Replicating Data into an Existing Dataservice

3.8. Replicating Data Out of a Cluster
3.8.1. Prepare: Replicating Data Out of a Cluster
3.8.2. Deploy: Replicating Data Out of a Cluster

3.9. Replicating from a Cluster to a Datawarehouse
3.9.1. Replicating from a Cluster to a Datawarehouse - Prerequisites
3.9.2. Replicating from a Cluster to a Datawarehouse - Configuring the Cluster Nodes
3.9.3. Replicating from a Cluster to a Datawarehouse - Configuring the Cluster-Slave
3.9.3.1. Replicating Data from a Cluster to a Datawarehouse [Staging Use Case]
3.9.3.2. Replicating Data from a Cluster to a Datawarehouse [INI Use Case]

3.10. Migrating and Seeding Data
3.10.1. Migrating from MySQL Native Replication 'In-Place'
3.10.2. Migrating from MySQL Native Replication Using a New Service
3.10.3. Seeding Data through MySQL
3.10.4. Seeding Data through tungsten_provision_thl

4. Deployment: Advanced
## 6. Command-line Tools

- 6.7.2. Connections During Manual Switch ................................................................. 246
- 6.7.3. Connections During Connection Failures .......................................................... 246
- 6.7.4. Other Errors ......................................................................................................... 247
- 6.7.5. Connector Keepalive ............................................................................................. 247
- 6.7.6. Connector Change User as Ping ............................................................................. 247

## 7. Tungsten Manager

- 7.1. Tungsten Manager Introduction .............................................................................. 273
- 7.2. Tungsten Manager Failover Behavior ....................................................................... 273
- 7.2.1. Failover Replication State Scenarios .................................................................... 273
- 7.2.2. Recovery Behavior After Failover ....................................................................... 275
- 7.2.3. Failover Response when MySQL Server Fails ...................................................... 275
- 7.2.4. Failover Response when Slave Applier is Latent ................................................. 276

## 8. Connector/Manager Interface

- 8.1. The cctrl Command ................................................................................................. 283
- 8.1.1. cctrl Command-line Options ................................................................................ 283
- 8.1.2. cctrl Modes ........................................................................................................... 283
- 8.1.3. cctrl Commands .................................................................................................... 286
- 8.1.3.1. cctrl admin Command ....................................................................................... 286
- 8.1.3.2. cctrl cd Command .............................................................................................. 286
- 8.1.3.3. cctrl cluster Command ....................................................................................... 287
- 8.1.3.4. cctrl create composite Command ...................................................................... 289
- 8.1.3.5. cctrl datasource Command ............................................................................... 289
- 8.1.3.6. cctrl expert Command ....................................................................................... 292
- 8.1.3.7. cctrl failover Command ...................................................................................... 292
- 8.1.3.8. cctrl help Command ........................................................................................ 292
- 8.1.3.9. cctrl ls Command ............................................................................................. 293
- 8.1.3.10. cctrl members Command ............................................................................... 294
- 8.1.3.11. cctrl physical Command ............................................................................... 294
11.1 Monitoring Block Commit Status ................................................................. 532
11.2 Improving Network Performance ................................................................. 533
11.3 Enable Heterogeneous Replication from a Cluster ........................................ 534
11.4 Tungsten Replicator Block Commit and Memory Usage ............................. 534
11.5 Connector Memory Management ................................................................. 536
11.6 Functional Testing ......................................................................................... 536
  11.6.1 Manual and Automatic Failover .............................................................. 536
  11.6.2 Backup and Restore ................................................................................. 537
  11.6.3 Connectivity .......................................................................................... 537
  11.6.4 Performance and Other Tests ................................................................. 538

A. Release Notes .................................................................................................. 539
  A.1 Tungsten Clustering 6.1.3 GA (17 February 2020) ........................................ 539
  A.2 Tungsten Clustering 6.1.2 GA (20 January 2020) ........................................ 541
  A.3 Tungsten Clustering 6.1.1 GA (28 October 2019) ......................................... 546
  A.4 Tungsten Clustering 6.1.0 GA (31 July 2019) ............................................. 549

B. Prerequisites .................................................................................................... 553
  B.1 Staging Host Configuration ......................................................................... 553
  B.2 Host Configuration ....................................................................................... 554
    1. Creating the User Environment ................................................................. 554
    1.2.1 Configuring Network and SSH Environment ................................... 555
    1.2.2.1 Network Ports .............................................................................. 556
    1.2.2.2 SSH Configuration ..................................................................... 557
    1.2.2.3 Host Availability Checks ............................................................. 558
    1.2.3 Directory Locations and Configuration ............................................ 558
    1.2.4 Configure Software ......................................................................... 558
    1.2.5. sudo Configuration ...................................................................... 559
    1.2.6 SELinux Configuration ................................................................... 560

  B.3 MySQL Database Setup ............................................................................. 560
    1. MySQL Version Support ......................................................................... 561
    1.3 MySQL Configuration ........................................................................... 561
    1.3.1 MySQL Configuration for Multimaster Deployments ...................... 564
    1.3.4 MySQL Configuration for Heterogeneous Deployments ................ 564
    1.3.5 MySQL User Configuration ............................................................... 564

  B.4 Oracle Database Setup ................................................................................ 565
    1. Oracle Version Support ......................................................................... 565
    1.4.2.1 Oracle Environment Variables .................................................... 565

  B.5 Prerequisite Checklist ................................................................................ 566

C. Troubleshooting .............................................................................................. 568
  C.1 Contacting Support ................................................................................... 568
    1. Support Request Procedure ................................................................... 568
    1.2 Creating a Support Account .................................................................. 568
    1.3 Generating Diagnostic Information ...................................................... 568
    1.4 Open a Support Ticket .......................................................................... 569
    1.5 Open a Support Ticket via Email ............................................................ 569
    1.6 Getting Updates for all Company Support Tickets ............................ 569
    1.7 Support Severity Level Definitions ....................................................... 569
    1.8 Generating Advanced Diagnostic Information ................................... 570

  C.2 Error/Cause/Solution .................................................................................. 572
    1. Lots of entries added to replicator log ...................................................... 572
    2. Error: could not settle on encryption_client algorithm ........................ 573
    2.3 The master replicator stopped with a JDBC error ............................... 573
    2.4 ERROR backupBackupTask Backup operation failed: null .......... 574
    2.5 Event application failed: seqno=20725782 fragno=0 message=java.sql.SQLException: Data too long for column 'eventid' at row 1 .... 574
    2.6 Unable to update the configuration of an installed directory .......... 575
    2.7 [S1000][unixODBC][MySQL][ODBC 5.3.11] Driver][SSL connection error: unknown error number [SQL]ERROR: Could not SQLConnect .......................... 575
    2.8 Connector shows errors with "java.net.SocketException: Broken pipe" ................................. 576
    2.9. ctrlt reports MANAGER (state=STOPPED) ........................................ 576
    2.10. Backup agent name not found: xtrabackup-full ......................... 576
    2.11. Replicator runs out of memory .......................................................... 577
    2.12. trepctl status hangs ................................................................. 577
    2.13. Replicator fails to connect after updating password ....................... 578
    2.14. Attempt to write new log record with equal or lower fragno: seqno=3 previous stored fragno=32767 attempted new fragno=32768 .......... 578
    2.15. ORA-00257: ARCHIVER ERROR. CONNECT INTERNAL ONLY, UNTIL FREED 579
    2.16. ERROR 1010 (HY000) at line 5094506: Error dropping database 579
     (can't rmdir './mysql-bin/', errno: 17) .................................................. 579
List of Figures

2.1. Witness: Active Service ................................................................. 28
2.2. Witness: Inactive Service ............................................................. 29
3.1. Topologies: Standalone HA Cluster ................................................. 54
3.2. Topologies: Composite Master/Slave Cluster ..................................... 58
3.3. Topologies: Multisite/Multimaster Clusters ....................................... 63
3.4. Topologies: Composite Multimaster Clusters ..................................... 77
3.5. Topologies: Replicating into a Dataservice ....................................... 104
3.6. Topologies: Replicating Data Out of a Cluster ................................... 107
3.7. Topologies: Replication from a Cluster to an Offboard Datawarehouse ..... 111
3.8. Topologies: Replication from a Cluster to an Offboard Datawarehouse ..... 113
3.9. Migration: Migrating Native Replication using a New Service ............... 122
5.1. Sequence: Shunning a Datasource ................................................ 174
5.2. Migration: Migrating Native Replication using a New Service ............... 207
5.3. Cacti Monitoring: Example Graphs ............................................... 219
6.1. Tungsten Connector Basic Architecture .......................................... 223
6.2. Basic MySQL/Application Connectivity ........................................... 224
6.3. Advanced MySQL/Application Connectivity ...................................... 225
6.4. Using Tungsten Connector for MySQL/Application Connectivity .......... 225
6.5. Tungsten Connector during a failed datasource .................................. 226
6.6. Tungsten Connector routing architecture ........................................ 228
6.7. Tungsten Connector Bridge Mode Architecture .................................. 238
6.8. Tungsten Connector Authentication ................................................ 240
7.1. Failover Scenario 1 ........................................................................ 274
7.2. Failover Scenario 2 ........................................................................ 274
7.3. Failover Scenario 3 ........................................................................ 275
9.1. tpm Staging Based Deployment ...................................................... 385
9.2. tpm INI Based Deployment ........................................................... 386
10.1. Filters: Pipeline Stages on Masters ............................................... 478
10.2. Filters: Pipeline Stages on Slaves .................................................. 479
8.1. Tungsten Deployment .................................................................... 553
List of Tables

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1. Key Terminology</td>
<td>22</td>
</tr>
<tr>
<td>2.1. Key Terminology</td>
<td>26</td>
</tr>
<tr>
<td>6.1. Routing Method Selection</td>
<td>229</td>
</tr>
<tr>
<td>6.2. Connector Command Line Sub-Commands</td>
<td>248</td>
</tr>
<tr>
<td>6.3. Inline Interface Commands</td>
<td>249</td>
</tr>
<tr>
<td>8.1. cctrl Command-line Options</td>
<td>283</td>
</tr>
<tr>
<td>8.2. cctrl Command-line Options</td>
<td>284</td>
</tr>
<tr>
<td>8.3. cctrl Command-line Options</td>
<td>284</td>
</tr>
<tr>
<td>8.4. cctrl Command-line Options</td>
<td>284</td>
</tr>
<tr>
<td>8.5. cctrl Command-line Options</td>
<td>284</td>
</tr>
<tr>
<td>8.6. cctrl Command-line Options</td>
<td>284</td>
</tr>
<tr>
<td>8.7. cctrl Command-line Options</td>
<td>285</td>
</tr>
<tr>
<td>8.8. cctrl Command-line Options</td>
<td>285</td>
</tr>
<tr>
<td>8.9. cctrl Command-line Options</td>
<td>285</td>
</tr>
<tr>
<td>8.10. cctrl Command-line Options</td>
<td>285</td>
</tr>
<tr>
<td>8.11. cctrl Commands</td>
<td>286</td>
</tr>
<tr>
<td>8.12. cctldatasource Commands</td>
<td>289</td>
</tr>
<tr>
<td>8.13. check_tungsten_latency Options</td>
<td>297</td>
</tr>
<tr>
<td>8.14. check_tungsten_online Options</td>
<td>298</td>
</tr>
<tr>
<td>8.15. check_tungsten_policy Options</td>
<td>299</td>
</tr>
<tr>
<td>8.16. check_tungsten_progress Options</td>
<td>300</td>
</tr>
<tr>
<td>8.17. check_tungsten_services Options</td>
<td>301</td>
</tr>
<tr>
<td>8.18. cluster_backup Command-line Options</td>
<td>302</td>
</tr>
<tr>
<td>8.19. connector Commands</td>
<td>305</td>
</tr>
<tr>
<td>8.20. ddlsan Command-line Options</td>
<td>306</td>
</tr>
<tr>
<td>8.21. ddlsan Supported Templates</td>
<td>317</td>
</tr>
<tr>
<td>8.22. dsctl Commands</td>
<td>317</td>
</tr>
<tr>
<td>8.23. dsctl Command-line Options</td>
<td>317</td>
</tr>
<tr>
<td>8.24. dsctl Command-line Options</td>
<td>317</td>
</tr>
<tr>
<td>8.25. dsctl Command-line Options</td>
<td>318</td>
</tr>
<tr>
<td>8.26. manager Commands</td>
<td>319</td>
</tr>
<tr>
<td>8.27. multi_trepctl Command-line Options</td>
<td>321</td>
</tr>
<tr>
<td>8.28. multi_trepctl .output Option</td>
<td>322</td>
</tr>
<tr>
<td>8.29. multi_trepctl Commands</td>
<td>323</td>
</tr>
<tr>
<td>8.30. query Common Options</td>
<td>324</td>
</tr>
<tr>
<td>8.31. replicator Commands</td>
<td>325</td>
</tr>
<tr>
<td>8.32. replicator Commands Options for condrestart</td>
<td>325</td>
</tr>
<tr>
<td>8.33. replicator Commands Options for console</td>
<td>325</td>
</tr>
<tr>
<td>8.34. replicator Commands Options for restart</td>
<td>326</td>
</tr>
<tr>
<td>8.35. replicator Commands Options for start</td>
<td>326</td>
</tr>
<tr>
<td>8.36. setupCCD.conf Configuration Options</td>
<td>327</td>
</tr>
<tr>
<td>8.37. tbl Options</td>
<td>330</td>
</tr>
<tr>
<td>8.38. trepctl Command-line Options</td>
<td>338</td>
</tr>
<tr>
<td>8.39. trepctl Replicator Wide Commands</td>
<td>339</td>
</tr>
<tr>
<td>8.40. trepctl Service Commands</td>
<td>341</td>
</tr>
<tr>
<td>8.41. trepctl backup Command Options</td>
<td>342</td>
</tr>
<tr>
<td>8.42. trepctl clients Command Options</td>
<td>344</td>
</tr>
<tr>
<td>8.43. trepctl offline-deferred Command Options</td>
<td>347</td>
</tr>
<tr>
<td>8.44. trepctl online Command Options</td>
<td>348</td>
</tr>
<tr>
<td>8.45. trepctl purge Command Options</td>
<td>352</td>
</tr>
<tr>
<td>8.46. trepctl reset Command Options</td>
<td>354</td>
</tr>
<tr>
<td>8.47. trepctl setrole Command Options</td>
<td>355</td>
</tr>
<tr>
<td>8.48. trepctl shard Command Options</td>
<td>355</td>
</tr>
<tr>
<td>8.49. trepctl status Command Options</td>
<td>356</td>
</tr>
<tr>
<td>8.50. trepctl wait Command Options</td>
<td>363</td>
</tr>
<tr>
<td>8.51. tpasswd Common Options</td>
<td>364</td>
</tr>
<tr>
<td>8.52. tungsten_find_orphaned Options</td>
<td>365</td>
</tr>
<tr>
<td>8.53. tungsten_find_position Options</td>
<td>366</td>
</tr>
<tr>
<td>8.54. tungsten_find_segment Options</td>
<td>366</td>
</tr>
<tr>
<td>8.55. tungsten_health_check Command-line Options</td>
<td>367</td>
</tr>
<tr>
<td>8.56. tungsten_monitor Command-line Options</td>
<td>368</td>
</tr>
<tr>
<td>8.57. tungsten_post_process Options</td>
<td>371</td>
</tr>
<tr>
<td>8.58. tungsten_prep_upgrade Command-line Options</td>
<td>371</td>
</tr>
<tr>
<td>8.59. tungsten Provision_slave Command-line Options</td>
<td>378</td>
</tr>
</tbody>
</table>
Preface

This manual documents Tungsten Clustering 6.1 up to and including 6.1.3 build 43. Differences between minor versions are highlighted stating the explicit minor release version, such as 6.1.3.x.

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

1. Legal Notice

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

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

All rights reserved.

2. Conventions

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

- **Text in this style** is used to show an important element or piece of information. It may be used and combined with other text styles as appropriate to the context.

- **Text in this style** is used to show a section heading, table heading, or particularly important emphasis of some kind.

- Program or configuration options are formatted using this style. Options are also automatically linked to their respective documentation page when this is known. For example, `tpm` and `--hosts` both link automatically to the corresponding reference page.

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

- Option values, for example on the command-line are marked up using this 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
shell> cd /opt/staging
shell> unzip tungsten-clustering-6.1.3-43.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;
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 user-input 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 B, Prerequisites?
  - Have you decided which installation method you will use? INI or Staging?
  - Have you chosen your 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 9.1, “Comparing Staging and INI tpm Methods”.
- Do you want to upgrade to the latest version?
  - See Section 9.5.17, “tpm update Command”.
- Are you trying to update or change the configuration of your system?
  - See Section 9.5.17, “tpm update Command”.
- Has your system suffered a failure?
  - For recovery methods and instructions, see Section 5.6, “Datasource Recovery Steps”.
- Would you like to perform database or operating system maintenance?
  - See Section 5.15, “Performing Database or OS Maintenance”.
- Do you need to backup or restore your system?
  - For backup instructions, see Section 5.10, “Creating a Backup”, and to restore a previously made backup, see Section 5.11, “Restoring a Backup”.
Chapter 1. Introduction

Tungsten Clustering™ provides a suite of tools to aid the deployment of database clusters using MySQL. Tungsten Clustering™ 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 Oracle, MongoDB, Vertica and others.

- **Tungsten Manager**
  
  The Tungsten Manager is responsible for monitoring and managing a Tungsten Clustering 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.

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

Tungsten Clustering 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>
<thead>
<tr>
<th>Continuent 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 Tungsten Clustering service consisting of multiple dataservices, typically at different physical locations.</td>
</tr>
<tr>
<td>datasource</td>
<td>Cluster</td>
<td>The collection of machines that make up a single Tungsten Dataservice. Individual hosts within the datasource are called datasources. Each datasource is identified by a unique name, and multiple dataservices can be managed from one server.</td>
</tr>
<tr>
<td>datasource</td>
<td>Database</td>
<td>The database on a host. Datasources include MySQL or Oracle.</td>
</tr>
<tr>
<td>datasource</td>
<td>Host or Node</td>
<td>One member of a datasource and the associated Tungsten components.</td>
</tr>
<tr>
<td>staging host</td>
<td>-</td>
<td>The machine [and directory] from which Tungsten Clustering™ is installed and configured. The machine does not need to be the same as any of the existing hosts in the datasource.</td>
</tr>
<tr>
<td>active witness</td>
<td>-</td>
<td>A machine in the datasource 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>passive witness</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 thedataservice.</td>
</tr>
<tr>
<td>coordinator</td>
<td></td>
<td>The datasource or active witness in a datasource that is responsible for making decisions on the state of the datasource. 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>

1.1. Tungsten Replicator

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

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

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

• Extractor

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

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

• Applier

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

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

• Transaction History Log (THL)

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

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

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

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

1.1. Transaction History Log (THL)

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

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

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

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

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

1.2. Tungsten Manager

The Tungsten Manager is responsible for monitoring and managing a Continuent Tungsten datasync. 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 datasync as a whole.

Primarily, the Tungsten Manager handles the following tasks:
Introduction

- Monitors the replication status of each datasource (node) 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
- Checks to determine the availability of datasources 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 B.2.2.3, “Host Availability Checks”.
- 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.

Please see the Tungsten Manager documentation section Chapter 7, Tungsten Manager for more information.

1.3. Tungsten Connector

The Tungsten Connector is a service that sits between your application server and your MySQL database. The connector routes connections from your application servers to the datasources 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 datasources 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 datasources 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:

- **Port Based Routing**
  Port based routing employs a second port on the connector host. All connections to this port are sent to an available slave.

- **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.

- **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.

- **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 routing 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 MySQL protocol, 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.
Chapter 2. Deployment

Creating a Tungsten Clustering (for MySQL) Dataservice using Tungsten Clustering requires careful preparation and configuration of the required components. This section provides guidance on these core operations, preparation and information such as licensing and best practice that should be used for all installations.

### 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 Tungsten Clustering service consisting of multiple dataservices, typically at different physical locations.</td>
</tr>
<tr>
<td>datasource</td>
<td>Cluster</td>
<td>A configured Tungsten Clustering service consisting of dataservers, datasources and connectors.</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>staging host</td>
<td>-</td>
<td>The machine from which Tungsten Clustering is installed and configured. The machine does not need to be the same as any of the existing hosts in the cluster.</td>
</tr>
<tr>
<td>staging directory</td>
<td>-</td>
<td>The directory where the installation files are located and the installer is executed. Further configuration and updates must be performed from this directory.</td>
</tr>
<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 ping 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 Tungsten Clustering 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 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 the system ping protocol or the Echo TCP/IP protocol on port 7 to determine whether a host is available. 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 B.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 6.3, “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 Tungsten Clustering 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 Tungsten Clustering deployment a master/slave or multi-master deployment model is used. For Tungsten Clustering 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 Tungsten Clustering 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 3.8, “Replicating Data Out of a Cluster”
- Section 3.9, “Replicating from a Cluster to a Datawarehouse”

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

### 2.1.4. Witness Hosts

Tungsten Clustering 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:

- **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.
- **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 is the root cause.

Passive Witness hosts are deprecated as of v6.1.0

Use Active Witness hosts instead
Passive Witness support will be removed in v6.2.0

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.

Tungsten Clustering 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.

Passive Witness hosts are deprecated as of v6.1.0
Use Active Witness hosts instead
Passive Witness support will be removed in v6.2.0

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

The rules for witness selection are as follows:

1. 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.

2. 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.

Passive Witness hosts are deprecated as of v6.1.0
Use Active Witness hosts instead
Passive Witness support will be removed in v6.2.0

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:

Figure 2.2. Witness: Inactive Service

Passive witnesses can be enabled when using tpm by using the \( \text{-witnesses} \) option:

```
shell> ./tools/tpm install alpha \( \text{-witnesses} \) hostC,hostD
```

Passive Witness hosts are deprecated as of v6.1.0

Use Active Witness hosts instead

Passive Witness support will be removed in v6.2.0

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

```
shell> ./tools/tpm install alpha \( \text{-enable-active-witnesses=true} \) \( \text{-members=hostA,hostB,hostC} \)
```

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, 7 as well as CentOS 5.x, 6.x, and 7.x versions are fully supported.</td>
</tr>
<tr>
<td>Linux</td>
<td>Ubuntu</td>
<td>Primary platform</td>
<td>Ubuntu 9.x-17.x versions are fully supported.</td>
</tr>
<tr>
<td>Linux</td>
<td>Debian/Suse/Other</td>
<td>Secondary Platform</td>
<td>Other Linux platforms are supported but are not regularly tested. We will fix any bugs reported by customers.</td>
</tr>
<tr>
<td>Linux</td>
<td>Docker</td>
<td>Unsupported</td>
<td>Unsupported - Use at your own risk. Docker containers are not well-suited for Tungsten deployments.</td>
</tr>
<tr>
<td>Solaris</td>
<td></td>
<td>Secondary Platform</td>
<td>Solaris 10 is fully supported. OpenSolaris is not supported at this time.</td>
</tr>
<tr>
<td>Mac OS X</td>
<td></td>
<td>Secondary platform</td>
<td>Mac OS/X is used for development at Continuent and could also be used as a Staging Host for Staging Deployments only, but it is not certified for us as a cluster or replicator node within a topology.</td>
</tr>
</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, 5.7 (without Geometry support in 5.3.0 or 6.0.0); full support was provided in 5.3.1 and 6.0.1</td>
<td>Primary platform</td>
<td>Statement and row based replication is supported. MyISAM and InnoDB table types are fully supported; InnoDB tables are recommended.</td>
</tr>
<tr>
<td>MySQL</td>
<td>5.7</td>
<td>Primary platform</td>
<td>Support is provided for compatibility with MySQL 5.7 in 5.0 and later, but new datatypes [JSON, virtual columns] in MySQL 5.7 are not yet supported.</td>
</tr>
<tr>
<td>Percona</td>
<td>5.5, 5.6, 5.7</td>
<td>Primary platform</td>
<td></td>
</tr>
<tr>
<td>MariaDB</td>
<td>5.5, 10.0, 10.1</td>
<td>Primary platform</td>
<td></td>
</tr>
<tr>
<td>Oracle [CDC]</td>
<td>10g Release 2 [10.2.0.5], 11g</td>
<td>Primary Platform</td>
<td>Synchronous CDC is supported on Standard Edition only; Synchronous and Asynchronous are supported on Enterprise Editions</td>
</tr>
<tr>
<td>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 heap space. 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 a INI-file based deployment, this space will be required on each server. For more information on the different methods, see Section 9.1, “Comparing Staging and INI Methods”.

- Deployment of a live installation also requires approximately 150MB.

- The THL files required for installation are based on the size of the binary logs generated by MySQL. THL size is typically twice the size of the binary log. This space will be required on each machine in the cluster. The retention times and rotation of THL data can be controlled, see Section D.1.5, “The thl Directory” for more information, including how to change the retention time and move files during operation.

  When replicating from Oracle, the size of the THL will depend on the quantity of Change Data Capture [CDC] information generated. This can be managed by altering the intervals used to check for and extract the information.

  A dedicated partition for THL or Tungsten Clustering is recommended to ensure that a full disk does not impact your OS or DBMS. Local disk, SAN, iSCSI and AWS EBS are suitable for storing THL. NFS is NOT recommended.
2.2.5. Java Requirements

Tungsten Replicator is known to work with with the following Java versions and JVMs:

- Oracle JVM/JDK 8
- OpenJDK 8

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:

<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><code>m4.xlarge</code> 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 `micro` instance can be used for a single cluster. For composite clusters, an instance size larger than `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>tpm Option</th>
<th>tpm Value</th>
<th>MySQL <code>my.cnf</code> Option</th>
<th>MySQL Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/root</code></td>
<td><code>datadlr</code></td>
<td><code>/volumes/mysql/data</code></td>
<td><code>datadlr</code></td>
<td><code>/volumes/mysql/data</code></td>
</tr>
<tr>
<td>MySQL Data</td>
<td><code>datasource-mysql-data-directory</code> [446]</td>
<td><code>/volumes/mysql/data</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MySQL Binary Logs</td>
<td><code>datasource-log-directory</code> [446]</td>
<td><code>/volumes/mysql/binlogs</code></td>
<td><code>log-bin</code></td>
<td><code>/volumes/mysql/binlogs/mysql-bin</code></td>
</tr>
<tr>
<td>Transaction History Logs</td>
<td><code>thl-directory [475]</code></td>
<td><code>/volumes/mysql/thl</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Recommended Replication Formats

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

2.2.7. Docker Support Policy

2.2.7.1. Overview

Continuent has traditionally had a relaxed policy about Linux platform support for customers using our products.

While it is possible to install and run Continuent Tungsten products (i.e., Clustering/Replicator/etc.) inside Docker containers, there are many reasons why this is not a good idea.
• MySQL Server [i.e. MySQL Community or Enterprise, MariaDB or Percona Server]
• Tungsten Manager, which handles health-checking, signaling and failover decisions (Java-based)
• Tungsten Replicator, which handles the movement of events from the MySQL master server binary logs to the slave databases nodes [Java-based]

Optionally, a fourth service, the Tungsten Connector (Java-based), may be installed as well, and often is.

2.2.7.3. Current State

As such, this means that the Docker container would also need to support these 3 or 4 layers and all the resources needed to run them.

This is not what containers were designed to do. In a proper containerized architecture, each container would contain one single layer of the operation, so there would be 3-4 containers per “node”. This sort of architecture is best managed by some underlying technology like Swarm, Kubernetes, or Mesos.

More reasons to avoid using Docker containers with Continuent Tungsten solutions:

• Our product is designed to run on a full Linux OS. By design Docker does not have a full init system like SystemD, SysV init, Upstart, etc... This means that if we have a process [Replicator, Manager, Connector, etc...] that process will run as PID 1. If this process dies the container will die. There are some solutions that let a Docker container to have a ‘full init’ system so the container can start more processes like ssh, replicator, manager, ... all at once. However this is almost a heavyweight VM kind of behavior, and Docker wasn't designed this way.

• Requires a mutable container – to use Tungsten Clustering inside a Docker container, the Docker container must be launched as a mutable Linux instance, which is not the classic, nor proper way to use containers.

• Our services are not designed as “serverless”. Serverless containers are totally stateless. Tungsten Clustering does not support this type of operation.

• Until we make the necessary changes to our software, using Docker as a cluster node results in a minimum 1.2GB docker image.

• Once Tungsten Clustering has been refactored using a microservices-based architecture, it will be much easier to scale our solution using containers.

• A Docker container would need to allow for updates in order for the Tungsten Cluster software to be re-configured as needed. Otherwise, a new Docker container would need to be launched every time a config change was required.

• There are known i/o and resource constraints for Docker containers, and therefore must be carefully deployed to avoid those pitfalls.

• We test on CentOS-derived Linux platforms.

2.2.7.4. What to Expect in the Future

Continuent does NOT have Docker containerization on the product roadmap at this time. That being said, we do intend to provide containerization support at some point in the future. Customer demand will contribute to the timing of the effort.

2.2.7.5. Summary

In closing, Continuent’s position on container support is as follows:

• Unsupported at this time for all products (i.e. Clustering/Replicator/etc.)

• Use at your own risk

2.3. Deployment Sources

Tungsten Clustering is available in a number of different distribution types, and the methods for configuration available for these different packages differs. See Section 9.1, “Comparing Staging and INI 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>tpm Command-line Configuration</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>tpm INI File Configuration</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Deploy Entire Cluster</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Deploy Per Machine</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Two primary deployment sources are available:
Deployment

- **Tar/GZip**

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

- **RPM/DEB Packages**

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

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

```
tungsten-clustering-6.1.3-43.tar.gz
```

The version number is 6.1.3 and build number 43. 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 tungsten-clustering-6.1.3-43.tar.gz
```

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

```
shell> cd tungsten-clustering-6.1.3-43
```

### 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 exists

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

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

---

**Note**

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

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

```
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 keysserver.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 exists

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

2.4. Common tpm Options During Deployment

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

Changes to the configuration should be made with `tpm update`. This continues the procedure of using `tpm install` during installation. See Section 9.5.17, “tpm update Command” for more information on using `tpm update`.

- `--datasource-systemctl-service [448]`

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

Depending on the system or environment other commands using the same basic structure may be used. For example, within CentOS 7, the command is `systemctl`. You can explicitly set the command to be used by using the `--datasource-systemctl-service [448]` to specify the name of the tool.

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

```
shell> systemctl mysql stop
```

Different commands must follow the same basic structure, the command configured by `--datasource-systemctl-service [448]`, the service-name, and the status (i.e. `stop`).

2.5. Best Practices

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

2.5.1. Best Practices: Deployment

- Identify the best deployment method for your environment and use that in production and testing. See Section 9.1, “Comparing Staging and INI tpm Methods”.
- Standardize the OS and database prerequisites. There are Puppet and Chef modules available for immediate use or as a template for modifications.
- For security purposes you should ensure that you secure the following areas of your deployment:
  - Ensure that you create a unique installation and deployment user, such as tungsten, and set the correct file permissions on installed directories. See Section B.2.3, “Directory Locations and Configuration”.
  - When using ssh and/or SSL, ensure that the ssh key or certificates are suitably protected. See Section B.2.2.2, “SSH Configuration”.
  - Use a firewall, such as `iptables` to protect the network ports that you need to use. The best solution is to ensure that only known hosts can connect to the required ports for Tungsten Clustering. For more information on the network ports required for Tungsten Clustering operation, see Section B.2.2.1, “Network Ports”.
  - If possible, use authentication and SSL connectivity between hosts to protect your data and authorisation for the tools used in your deployment. See Section 2.7, “Deployment Security” for more information.
  - Choose your topology from the deployment section and verify the configuration matches the basic settings. Additional settings may be included for custom features but the basics are needed to ensure proper operation. If your configuration is not listed or does not match our documented settings; we cannot guarantee correct operation.
  - If there are an even number of database servers in the cluster, configure the cluster with a witness host. An active witness is preferred but a passive one will ensure stability. See Section 2.1.4, “Witness Hosts” for an explanation of the differences and how to configure them.
• If you are using ROW replication, any triggers that run additional `INSERT/UPDATE/DELETE` operations must be updated so they do not run on the slave servers.

• Make sure you know the structure of the Tungsten Clustering home directory and how to initialize your environment for administration. See Section 5.1, “The Tungsten Clustering Home Directory” and Section 5.2, “Establishing the Shell Environment”.

• Prior to migrating applications to Tungsten Clustering test failover and recovery procedures from Chapter 5, Operations Guide. Be sure to try recovering a failed master and reprovisioning failed slaves.

### 2.5.2. Best Practices: Operations

• Setup proper monitoring for all servers as described in Section 5.17, “Monitoring Tungsten Clustering”.

• Configure the Tungsten Clustering services to startup and shutdown along with the server. See Section 4.3, “Configuring Startup on Boot”.

• Schedule the Section 8.8, “The cluster_backup Command” tool on each database server at least each night. The script will take a backup of at least one server. Skip this step if you have another backup method scheduled that takes consistent snapshots of your server.

### 2.5.3. Best Practices: Maintenance

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

• Schedule regular tests for local and DR failover. This should at least include switching the master server to another host in the local cluster. If possible, the DR cluster should be tested once per quarter.

• Disable any automatic operating system patching processes. The use of automatic patching will cause issues when all database servers automatically restart without coordination. See Section 5.15.3, “Performing Maintenance on an Entire Dataservice”.

• Regularly check for maintenance releases and upgrade your environment. Every version includes stability and usability fixes to ease the administrative process.

### 2.6. Prepare Hosts

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

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

For RHEL/CentOS-based distributions:

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

For Ubuntu-based distributions:

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

• Install the Continuent Puppet module.

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

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

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

### 2.6.1. Prepare MySQL Hosts

Use the Continuent Puppet module to install all prerequisites including MySQL. This will implement the prerequisites described in Section B.2, “Host Configuration” and Section B.3, “MySQL Database Setup”.

```shell
shell > puppet apply -e "class { 'tungsten' :
```
2.6.2. Prepare Connector Hosts

Use the Continuent Puppet module to install all common prerequisites. This will implement the prerequisites described in Section B.2, “Host Configuration”.

```
shell > puppet apply -e "class {'tungsten' :}"
```

2.6.3. Prepare Active Witness Hosts

Use the Continuent Puppet module to install all common prerequisites. This will implement the prerequisites described in Section B.2, “Host Configuration”.

```
shell > puppet apply -e "class {'tungsten' :}"
```

2.6.4. Deploy SSH Keys

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

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

  ```
  shell > puppet apply -e "class {'tungsten': sshPublicKey => "-----BEGIN RSA PRIVATE KEY-----
  MIIEogIBAAKCAQEAoxtELMB3x3zFz6fVX6g6Fg4Pq2b18TFg6J3CVmWXRxCu9oPG0g
  QfLCZCldSuy5Y7fGxGc5Z1eQyEKLlREh4W8hBHGJLqQeXs0rBn64AuMz
  aBfx7y JP3wHdFwFNu6uSeS013K0hWf7TFszvoxRkQGuHNcLlBnH0wh6C06hG
  oz2abah8b3D10JbhncqMq40bL6u6vR0vbdVx0ioF0g+gbyB45T14g608iFfC
  snmxZnXf5pxb4OL8kAXRcUS+o1g4a31La5GQYADQ102U096KHEK3jOjyOBjX7Y
  XT9wvKHP4KfIv18vYfY8c5K0h10na6QozGb0H3eTb Dulw6DMGqadd6T6c64p144atZ5hVG
  u2D10FytdsFw5h3u5u6n5gC6U+uvtxF1nXvPyVdhr+rF3MqV5VSmO4V4F
  ZZs1igvQRX7SNZK3MAaB/0dcg&N09HbF.redylyyxyolx1iy1vumq1d6qyA9IPh
  hox31Lj/zsRM7QD/L/vpCvJ3CvCvyJ3CMM1u9Y5y3vV9YK9Cjy7cYhih
  V5G5V1fc8Pvprz86k/jyb20lN6L40OxzJLyi5aCjYbPRr7b2BKSzf2fSPyrYTS
  cauvNBN1111Y11IKuTT3ZHy/URf7ln1vslzsdLJudc/UYUx90v65F+08BCCG0F15S
  Zbbg6AgCYEA5S4k90Hn5oD509PaQ5R8Xa/mPeGQy/0j/1addSxCOuI7uOV9uHuz
  xwh3K22h1=7QfzQa8Sl3jy6l2eS2IFqPQHO9A91v/xu37QgFj6n0m1k38j0Q6
  sngKSKipbCmTlU0t0yCv3HNWf8430xJ4k7U060etd0bLr/PKAMxXIEGyEA3cag
  YL3KUaC2z3TGP52NA9Cngv0gUfogx32El571l87iUJFyXRjX5jIu5u3b3Febasv0sFh
  #5n08hlahqJ31lmIzpggu5Qb88qO1m93pGuxLjvGnU0cFBKxhTbd0ObgCmCOJ3
  tytcscxTv1x5l2paAs4Y03L30obwqg4d8bMcgYfAzycjCu8cCPdcrcRg2a0v8bj
  n0xTAC4k6mNIPR9k/Q0bB7lTls8qKxZLCGGZ8K8960ryFCIFz33uIz3p69915L
  KNOq5KXNQgyv6eASb3k+jd23+yj3L2pauuAHH5hbrnax113vLs6ac8K티h3uL
  KNOg3XXLjgGk3BvZ/nQ8bg9yK3WP59Xp64Zjilq3w3kdfG6r3Pm8y3fYePZyMh
  JPPg+SGKLw66zq7jejir3rd0G6r1PcxF0ZvZC8/vQ0bJ/Eo08Br3c06G310c7c06
  V30023322Tjv0r/dBvHA9HYDurXkLr7aKPR9C96JwVXZ86dtQ11dGammaJyY
  8R3Fw4gmoA66Q0DOEF094b4N3H15iEHMD07QBxP3838ND312D6Q+p896055
  9C0j1eK1E02KTL8812+3Q135/F1K1q26zK0X060Fp9vPGqvG3mN7r8Xs876ssX3
  n2QFX7x3VSe3P19bgfC6nx5V7SwSVMj3v6aCnBwEqZ7h7tWq=
  -----END RSA PRIVATE KEY-----"
  sshPrivateCert => "AAAAB3NzeCyMQEEAAAAlAQABAD8dOQGm7WY9473VY9117QhBMc5XK79u7UkKksx
  h6g5f44KhS0cVkUw2nW7b3L18yf/lXvXzC7YVuaAsOS185bR/6f2xZwNj6s6Ef8F6
  sb4LoU0gWVd6Cmg
  Yvta/sQ7T84jOJdxZ/kP3syo6eC9uy2484zVxzaq4z5w7Cp+yjFQPP6QK33C95G9sZbYxYdJVRu06e0v5vI
  C5d18Xg7J06v3CvR3J6Cw4D549X6dbd6N2sL9yZT2SpwReuv2687yKu6PV1IgA995649Khb2wS6gCqC"
  }
```

- After unpacking the software package run the `tpm ssh-copy-cert` to output a set of commands that will setup the SSH certificate and authorized keys for a user. Run these commands as the `tungsten` system user before proceeding with deployment.

Shell> ./tools/tpm ssh-copy-cert

```

```
2.7. Deployment Security

Tungsten Clustering supports SSL, TLS and certificates for both communication and authentication for all components within the system. This security is disabled by default and includes:

- Authentication between command-line tools [cctrl], and between background services.
- SSL/TLS between command-line tools and background services.
- SSL/TLS between Tungsten Replicator and datasources.
- SSL/TLS between Tungsten Connector and datasources.
- File permissions and access by all components.

If you are using a single staging directory to handle your complete installation, tpm will automatically create the necessary certificates for you. If you fit in the below categories, you will need to use manually generated certificates.

- Installing heterogeneous replication using independent configurations
- MSMM, Cluster-Slave replication or anything using multiple Continuent packages
- Installing from multiple Staging Directories

Important

Installing from a staging host will automatically generate certificates and configuration for a secured installation. No further changes or actions are required.

For INI-based installations, there are additional steps required to copy the needed certificate files to all of the nodes.

2.7.1. Enabling Security

By default, security is disabled for the entire installation.

Security can be enabled by using the --disable-security-controls=false [451] option to the tpm command:

- --disable-security-controls=false [451]

This has the same effect as adding --file-protection-level=0027 [455], --rmi-ssl=true [454], --thl-ssl=true [454], --rmi-authentication=true [454], --jgroups-ssl=true [453].

Important

Installing from a staging host will automatically generate certificates and configuration for a secured installation. No further changes or actions are required.
2.7.1.1. Enabling Security using the Staging Method

Security can be enabled either during initial installation or via an update. For many reasons, it is much easier to enable SSL at install time. Both procedures follow below.

Enabling During Install

Security can be enabled at install time by using the \texttt{--disable-security-controls=false} option to the \texttt{tpm configure} command.

\begin{verbatim}
shell> tools/tpm configure defaults --disable-security-controls=false \\
[...the rest of the configuration options...]
shell> tools/tpm install
\end{verbatim}

\textbf{Important}

Installing from a staging host will automatically generate certificates and configuration for a secured installation. No further changes or actions are required.

Enabling Post-Installation

Security can be enabled after install time by using the \texttt{--disable-security-controls=false} option to the \texttt{tpm configure} command followed by a special invocation of the \texttt{tpm update} command.

\begin{verbatim}
shell> tools/tpm configure defaults --disable-security-controls=false
shell> tools/tpm update --replace-jgroups-certificate --replace-tls-certificate --replace-release
\end{verbatim}

\textbf{Warning}

This update will force all running processes to be restarted. Connectors MUST be done at the same time or they will no longer be able to communicate with the managers.

2.7.1.2. Enabling Security using the INI Method

Security can be enabled either during initial installation or via an update. For many reasons, it is much easier to enable SSL at install time. Both procedures follow below.

Enabling During Install

\begin{itemize}
  \item First, configure the \texttt{tungsten.ini} file as follows:
  \begin{verbatim}
  disable-security-controls=false [451]
  start-and-report=false [472]
  \end{verbatim}
  \item Next, do the fresh install on each node, which will generate new, different certificates on every node.
  \begin{verbatim}
  shell> tools/tpm install
  \end{verbatim}
  \item You must then select one of the nodes and copy that node's certificate files to all other nodes.
  \begin{verbatim}
  shell> for i in `seq 2 6`; do scp /opt/continuent/share/[jpt]* db$i:/opt/continuent/share/; done
  shell> for i in `seq 2 6`; do scp /opt/continuent/share/[jpt]* db$i:/opt/continuent/share/; done
  \end{verbatim}
  \item On all nodes:
  \begin{verbatim}
  shell> startall
  \end{verbatim}
\end{itemize}

Enabling Post-Installation

Security can be enabled after install time by updating the \texttt{tungsten.ini} file, followed by a special invocation of the \texttt{tpm update} command on all nodes.

\begin{itemize}
  \item First, configure the \texttt{tungsten.ini} file as follows:
  \begin{verbatim}
  disable-security-controls=false [451]
  start-and-report=false [472]
  \end{verbatim}
\end{itemize}
• Enable Maintenance mode on the cluster

```shell
cctrl -multi
cctrl> use world
cctrl> set policy maintenance
```

• Do the update on each node, which will generate new, different certificates on every node.

```
Warning
This update procedure will force all running Tungsten processes to be stopped. Connectors MUST be done at the same time or they will no longer be able to communicate with the Managers.
```

```
shell> stopall
tpm query staging
cd {staging_directory}
tools/tpm update --replace-jgroups-certificate --replace-tls-certificate --replace-release
```

• As with a fresh install, you must then select one of the nodes and copy that node's certificate files to all other nodes:

For example, to seed a 6-node composite cluster, login to db1 and copy both the main and backup files to the other five nodes:

```
shell> for i in `seq 2 6`; do scp /opt/continuent/share/*db${i}:/opt/continuent/share; done
shell> for i in `seq 2 6`; do scp /opt/continuent/share/*.db${i}:/opt/continuent/share; done
```

• On all nodes:

```
shell> startall
```

### 2.7.2. Disabling Security

There may be situations where security must be disabled for the entire installation.

Security can be disabled in the following ways during configuration with `tpm`:

- `--disable-security-controls=true`
  
  This has the same effect as adding `--file-protection-level=none`, `--rmi-ssl=false`, `--rmi-authentication=false`, `--jgroups-ssl=false`.

- `--file-protection-level=none`
  
  Disables file level protection, including ownership and file mode settings.

- `--rmi-ssl=false`
  
  Disables the use of SSL/TLS for communicating with services, this includes starting, stopping, or controlling individual services and operations, such as putting Tungsten Replicator online or offline.

- `--rmi-authentication=false`
  
  Disables the use of authentication when accessing and controlling services.

- `--thl-ssl=false`
  
  Disables the use of SSL/TLS for THL transmission between replicators.

- `--jgroups-ssl=false`
  
  Disables SSL/TLS for group communication within the cluster.

### 2.7.3. Creating Suitable Certificates

By default, `tpm` can automatically create suitable certificates and configuration for use in your deployment. To create the required certificates by hand, use the following steps:

• Generating a JGroups Certificate

Run this command to create the keystore in `/etc/tungsten`. You may use your own location, but the values for `-storepass` and `-keypass` must match.

```
shell> keytool -genseckey -alias jgroups 
    -validity 365
    -keyalg Blowfish -keysize 56 -keystore /etc/tungsten/jgroups.jceks
```
2.7.4. Installing from a Staging Host with Manually Generated Certificates

Follow the steps in Section 2.7.3, “Creating Suitable Certificates” to create JGroups and TLS certificates.

Update your configuration to specify these certificates and the keystore password:

```
shell> tools/tpm configure SERVICE
   --java-tls-keystore-path=/etc/tungsten/tls.jks
   --java-jgroups-keystore-path=/etc/tungsten/jgroups.jceks
   --java-keystore-password=mykeystorepass
```

2.7.5. Installing via INI File with Manually Generated Certificates

Follow the steps in Section 2.7.3, “Creating Suitable Certificates” to create JGroups and TLS certificates.

• Transfer the generated certificates to the same path on all hosts.

• Update your configuration to specify these certificates and the keystore password:

```
java-tls-keystore-path=/etc/tungsten/tls.jks
java-jgroups-keystore-path=/etc/tungsten/jgroups.jceks
java-keystore-password=mykeystorepass
```

2.7.6. Installing via INI File with CA-Signed Certificates

• This procedure will take a signed certificate from a known Certificate Authority and use it as the basis for all SSL operations within the cluster, not including Connector client-server SSL, which is configured separately. Please visit Section 2.7.13, “Configuring Connector SSL” for more information about configuring Connector SSL.

• The below example procedure assumes that you have an existing, installed and running cluster with security enabled by setting `disable-security-controls=false`.

Assume a 3-node cluster called `alpha` with member hosts `db1`, `db2` and `db3`.

Warning

In all examples below, because you are updating an existing secure installation, the password `tungsten` is required, do not change it.

• Select one node to create the proper set of certs, i.e. `db1`:

```
shell> su - tungsten
shell> mkdir /etc/tungsten/secure
shell> mkdir ~/certs
shell> cd ~/certs
```

• Copy the available files (CA cert, Intermediate cert [if needed], signed cert and signing key) into `~/certs/`, i.e.:

```
ca.crt.pem
int.crt.pem
signed.crt.pem
signing.key.pem
```

• Create a pkcs12 (.p12) version of the signed certificate:

```
shell> openssl pkcs12 -export -in ~/certs/signed.crt.pem -inkey ~/certs/signing.key.pem -out ~/certs/tungsten_sec.crt.p12 -name replserver
Enter Export Password: tungsten
Verifying - Enter Export Password: tungsten
```

• Create a pkcs12-based keystore [.jks] version of the signed certificate:
• Import the Certificate Authority’s certificate into the keystore:

```shell
keytool -import -alias careplserver -file ~/certs/ca.crt.pem -keypass tungsten -keystore /etc/tungsten/secure/tungsten_keystore.jks -storepass tungsten
```

Certificate was added to keystore

• Import the Certificate Authority’s intermediate certificate (if supplied) into the keystore:

```shell
keytool -import -alias interreplserver -file ~/certs/int.crt.pem -keypass tungsten -keystore /etc/tungsten/secure/tungsten_keystore.jks -storepass tungsten
```

Certificate was added to keystore

• Export the cert from the keystore into file `client.cer` for use in the next step to create the truststore:

```shell
keytool -export -alias replserver -file ~/certs/client.cer -keystore /etc/tungsten/secure/tungsten_keystore.jks
```

Certificate stored in file `/home/tungsten/certs/client.cer`

• Create the truststore:

```shell
keytool -import -trustcacerts -alias replserver -file ~/certs/client.cer -keystore /etc/tungsten/secure/tungsten_truststore.ts -storepass tungsten -noprompt
```

Certificate was added to keystore

• Create the rmi_jmx password store entry:

```shell
tpasswd -c tungsten tungsten -t rmi_jmx -p /etc/tungsten/secure/tungsten_truststore.ts -e -ts /etc/tungsten/secure/tungsten_truststore.ts -tsp tungsten
```

Using parameters:

```
-----------------
security.properties    = /opt/continuent/tungsten/cluster-home/../cluster-home/conf/security.properties
password.file.location    = /etc/tungsten/secure/passwords.store
encrypted.password    = true
truststore.location    = /etc/tungsten/secure/tungsten_truststore.ts
truststore.password    = *********
-----------------
```

Creating non existing file: `/etc/tungsten/secure/passwords.store`

User created successfuly: tungsten

• Create the tls password store entry:

```shell
tpasswd -c tungsten tungsten -t unknown -p /etc/tungsten/secure/passwords.store -e -ts /etc/tungsten/secure/tungsten_truststore.ts -tsp tungsten
```

Using parameters:

```
-----------------
security.properties    = /opt/continuent/tungsten/cluster-home/../cluster-home/conf/security.properties
password.file.location    = /etc/tungsten/secure/passwords.store
encrypted.password    = true
truststore.location    = /etc/tungsten/secure/tungsten_truststore.ts
truststore.password    = *********
-----------------
```

User created successfuly: tungsten

• List and verify the user for each security service password store entry, rmi_jmx and tls (which has a display tag of `unknown`):

```shell
tpasswd -l -p /etc/tungsten/secure/passwords.store -ts /etc/tungsten/secure/tungsten_truststore.ts
```

Using parameters:

```
-----------------
security.properties    = /opt/continuent/tungsten/cluster-home/../cluster-home/conf/security.properties
password.file.location    = /etc/tungsten/secure/passwords.store
encrypted.password    = true
truststore.location    = /etc/tungsten/secure/tungsten_truststore.ts
truststore.password    = *********
-----------------
```

Listing users by application type:

```
[unknown]
```

```
tungsten
```

```
[rmi_jmx]
```

```
[unknown]
```

```
tungsten
```

```
[rmi_jmx]
```
On host db1, transfer the generated certificates to the same path on all remaining hosts:

```
for host in `seq 2 3`; do rsync -av /etc/tungsten/secure/ db$host:/etc/tungsten/secure/; done
```

Edit the `/etc/tungsten/tungsten.ini` configuration file on all nodes and add:

```
[defaults]
... disable-security-controls=false
java-keystore-path=/etc/tungsten/secure/tungsten_keystore.jks
java-keystore-password=tungsten
java-truststore-path=/etc/tungsten/secure/tungsten_truststore.ts
java-truststore-password=tungsten
rmi-ssl=true
rmi-authentication=true
rmi-user=tungsten
java-passwordstore-path=/etc/tungsten/secure/passwords.store
```

On one node only, enable maintenance mode:

```
cctrl> set policy maintenance
```

On ALL nodes, stop the cluster software, execute the update, then start the cluster:

```
Warning
This procedure requires the complete restart of all layers of the Cluster, and will cause a brief downtime.
```

```
shell> tpm query staging
shell> cd (staging_dir)
shell> stopall
shell> tools/tpm update --replace-release
shell> startall
```

On one node only, enable automatic mode and check cluster status:

```
shell> cctrl
Tungsten Clustering 6.0.5 build 40
alpha: session established, encryption=true, authentication=true
```

```
cctrl> ls
COORDINATOR[db1:AUTOMATIC:ONLINE]
ROUTERS:
+---------------------------------------------------------------------------------+
|connector@db1[9871](ONLINE, created=0, active=0)                                 |
|connector@db2[27930](ONLINE, created=0, active=0)                                |
|connector@db3[23727](ONLINE, created=0, active=0)                                |
+---------------------------------------------------------------------------------+
```

```
DATASOURCES:
+---------------------------------------------------------------------------------+
|db1(master:ONLINE, progress=1, THL latency=0.656)                                |
|STATUS [OK] [2019/06/06 12:48:11 PM UTC]                                         |
|  MANAGER(state=ONLINE)                                                         |
|  REPLICATOR(role=master, state=ONLINE)                                         |
|  DATASERVER(state=ONLINE)                                                       |
|  CONNECTIONS(created=0, active=0)                                               |
+---------------------------------------------------------------------------------+
```

```
|db2(slave:ONLINE, progress=1, latency=9.858)                                     |
|STATUS [OK] [2019/06/06 12:48:11 PM UTC]                                         |
|  MANAGER(state=ONLINE)                                                         |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                             |
|  DATASERVER(state=ONLINE)                                                       |
|  CONNECTIONS(created=0, active=0)                                               |
+---------------------------------------------------------------------------------+
```

```
|db3(slave:ONLINE, progress=1, latency=19.235)                                    |
|STATUS [OK] [2019/06/06 12:48:11 PM UTC]                                         |
|  MANAGER(state=ONLINE)                                                         |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                             |
|  DATASERVER(state=ONLINE)                                                       |
|  CONNECTIONS(created=0, active=0)                                               |
+---------------------------------------------------------------------------------+
```

```
|  MANAGER(state=ONLINE)                                                         |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                             |
|  DATASERVER(state=ONLINE)                                                       |
|  CONNECTIONS(created=0, active=0)                                               |
+---------------------------------------------------------------------------------+
```

```
42
```
Deployment

2.7.7. Replacing the JGroups Certificate from a Staging Directory

If you meet the requirements to use an automatically generated certificate from the staging directory, the `tpm update` command can handle the certificate replacement. Simply add the `--replace-jgroups-certificate` option to your command. This will create errors if your staging configuration does not reflect the full list of hosts or if you limit the command to a specific host.

```bash
shell> tools/tpm update --replace-jgroups-certificate
```

If you do not meet these requirements, generate a new certificate and update it through the `tpm` command.

```bash
shell> tools/tpm configure SERVICE
    --java-jgroups-keystore-path=/etc/tungsten/jgroups.jceks
    --java-keystore-password=mykeystorepass
```

Then perform an update and replace the entire release directory:

```bash
shell> tools/tpm update --replace-release
```

2.7.8. Replacing the TLS Certificate from a Staging Directory

If you meet the requirements to use an automatically generated certificate from the staging directory, the `tpm update` command can handle the certificate replacement. Simply add the `--replace-tls-certificate` option to your command. This will create errors if your staging configuration does not reflect the full list of hosts or if you limit the command to a specific host.

```bash
shell> tools/tpm update --replace-tls-certificate
```

If you do not meet these requirements, generate a new certificate and update it through the `tpm` command.

```bash
shell> tools/tpm configure SERVICE
    --java-tls-keystore-path=/etc/tungsten/tls.jks
    --java-keystore-password=mykeystorepass
```

Then perform an update and replace the entire release directory:

```bash
shell> tools/tpm update --replace-release
```

2.7.9. Removing JGroups Encryption from a Staging Directory

Using the `tpm` update command, the jgroups encryption can be easily removed.

```bash
shell> tpm configure SERVICE
    --jgroups-ssl=false
```

Then perform an update and replace the entire release directory:

```bash
shell> tpm update --replace-release
```

2.7.10. Removing JGroups Encryption via INI File

To remove the JGroups encryption from a running cluster:

1. Put the cluster into `MAINTENANCE` mode
2. Update the INI file
   ```ini
   jgroups-ssl=false
   ```
3. Run `tpm update` and restart the manager
   ```bash
   shell> tpm update
   shell> manager restart
   ```
4. Check the cluster through `cctrl`
5. Put the cluster back into `AUTOMATIC` mode

2.7.11. Removing TLS Encryption from a Staging Directory

Using the `tpm update` command, the general Continuent service encryption can be easily removed.

```bash
shell> tpm configure SERVICE
    --thl-ssl=false
    --rmi-ssl=false
    --rmi-authentication=false
```
Then perform an update and replace the entire release directory:

```bash
shell> tpm update --replace-release
```

### 2.7.12. Removing TLS Encryption via INI File

To remove the TLS encryption from a running cluster:

1. Put the cluster into `MAINTENANCE` mode
2. Update the INI file
   ```
thl-ssl=false
rm-ssl=false
rm-authentication=false
   ```
3. Run `tpm` update and restart the manager
   ```
shell> tpm update
shell> manager restart
shell> replicator restart
   ```
4. Check the cluster through `cctrl`
5. Cycle through the connectors and restart them
   ```
shell> connector restart
   ```
6. Put the cluster back into `AUTOMATIC` mode

### 2.7.13. 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 2.7.14, “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.

**Note**

Please note that when operating in Bridge mode, the Connector is only involved in picking the correct server. In this situation the SSL configuration will be identical to the regular MySQL SSL setup, as explained in the MySQL documentation located here: [https://dev.mysql.com/doc/refman/8.0/en/using-encrypted-connections.html](https://dev.mysql.com/doc/refman/8.0/en/using-encrypted-connections.html)

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

```bash
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.

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

### 2.7.13.1. Setup Environment and Paths

Use these environment values on all Database & Connector nodes.
### 2.7.13.2. 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

```
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
```

2. Generate Server Cert

```
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
```

3. Generate Client Cert

```
shell> openssl req -newkey rsa:2048 -days 3600 -nodes -keyout $MYSQL_CERTS_PATH/client-key.pem -out $MYSQL_CERTS_PATH/client-req.pem
shell> openssl rsa -in $MYSQL_CERTS_PATH/client-key.pem -out $MYSQL_CERTS_PATH/client-key.pem
shell> openssl x509 -req -in $MYSQL_CERTS_PATH/client-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/client-cert.pem
```

4. Verify All Certificates

```
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 certificates)

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

6. Set proper ownership and permissions on ALL DB nodes

```
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 certificates you just created (add three lines to the `[mysqld]` stanza)

```
shell> vi /etc/my.cnf
[mysqld]
...  
port=13306
# add three lines for SSL support
ssl-ca=/etc/mysql/certs/ca-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 5.15.3, “Performing Maintenance on an Entire Dataservice” for more information.

   ctrl>r ls
   ctrl>r datasource db3 shun
   db3# service mysql restart
   ctrl>r recover
   ctrl>r datasource db2 shun
   db2# service mysql restart
   ctrl>r recover
   ctrl>r switch to db2
   ctrl>r datasource db1 shun
   db1# service mysql restart
   ctrl>r recover
   ctrl>r switch to db1
   ctrl>r 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.

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

10. Verify that MySQL is working with SSL

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

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

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

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

    c. Verify SSL:

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

       Important

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

2.7.13.3. Enable and Test SSL encryption from the Connector to the Database

1. Convert MySQL Client Cert to pkcs12 format

   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> 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> 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> keytool -import -alias mysqlServerCACert -file $MYSQL_CERTS_PATH/ca-cert.pem
   -keystore $CONN_CERTS_PATH/tungsten_connector_keystore.jks
   -storepass secret
   -noprompt
5. For INI-based deployments only, copy the certs to all Connector nodes (repeat as needed so that every Connector node has the same certificates)

```
shell> rsync -av $CONN_CERTS_PATH/ connectorHost:$CONN_CERTS_PATH/
```

6. Set proper ownership and permissions on ALL Connector nodes

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

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

See Section 6.6.1, “user.map File Format” for more information.

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

8. Update the Connector configuration to enable SSL

- Staging Method

```
shell> tpm query staging
shell> cd ${STAGING_DIR}
shell> tools/tpm configure {yourServiceName} \
-connector-ssl=true \
--java-connector-keystore-password=secret \
--java-connector-truststore-password=secret \
--java-connector-truststore-path=$CONN_CERTS_PATH/tungsten_connector_truststore.ts \
--java-connector-keystore-path=$CONN_CERTS_PATH/tungsten_connector_keystore.jks
```

- INI Method

```
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=$CONN_CERTS_PATH/tungsten_connector_truststore.ts
java-connector-keystore-path=$CONN_CERTS_PATH/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 “SSLIN=false SSLOUT=true”

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

SSLOUT 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;
```
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
--------------
mysql  Ver 14.14 Distrib 5.5.42-37.1, for Linux (x86_64) using readline 5.1
        Connection id:      70
        Current database:
        Current user:      app_user@app1
        SSL:         Not in use
        Current pager:      stdout
        Using outfile:      ''
        Using delimiter:   ;
        Server version:      5.5.42-37.1-log-tungsten Percona Server (GPL), Release 37.1, Revision 39acee8
        Protocol version:   10
        Connection:      app1 via TCP/IP
        Server characterset:   latin1
        Db     characterset:   latin1
        Client characterset:   latin1
        Conn.  characterset:   latin1
        TCP port:      3306
        Uptime:         2 hours 27 min 53 sec
        Threads:        4  Questions: 41474  Slow queries: 0  Opens: 47
        Flush tables: 2  Open tables: 10  Queries per second avg: 4.674
--------------

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.

2.7.13.4. 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.

mysql> tungsten connection status;
-----------------------------
<table>
<thead>
<tr>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>db1@east(master:ONLINE) STATUS(OK), QOS=RW_STRICT SSL.IN=true SSL.OUT=true</td>
</tr>
</tbody>
</table>
-------------------------------
1 row in set (0.00 sec)

3. Check the SSL status
### Note

Expecting "Cipher in use is \texttt{XXX-XXX-XXXXXX-XXX}"  

SSL is in use because the \texttt{mysql} client was invoked in SSL mode.  
The connection from the \texttt{mysql} client to the database is encrypted.

```
mysql> status

mysql  Ver 14.14 Distrib 5.5.42-37.1, for Linux (x86_64) using readline 5.1
Connection id:       68
Current database:
Current user:       ssl_user@app1
SSL:               Cipher in use is DHE-RSA-AES256-SHA
Current pager:      stdout
Using outfile:      ''
Using delimiter:    ;
Server version:     5.5.42-37.1-log-tungsten Percona Server (GPL), Release 37.1, Revision 39acee0
Protocol version:   10
Connection:         app1 via TCP/IP
Server characterset: latin1
Db     characterset:   latin1
Client characterset: latin1
Conn.  characterset:   latin1
TCP port:          3306
Uptime:            2 hours 33 min 32 sec
Threads:           4  Questions: 43065  Slow queries: 0  Opens: 47
Flush tables: 2  Open tables: 10  Queries per second avg: 4.674
```

### Important

If you are able to login to MySQL and see that the "tungsten connection status:" is "SSLIN=true SSLOUT=true", and the "status:" contains "Cipher in use is \texttt{XXX-XXX-XXXXXX-XXX}", then you have successfully configured SSL-encrypted communication between the Application/Client and MySQL through the Connector.

#### 2.7.14. Creating the Truststore and Keystore

The SSL configuration works through two separate files that define the server and client side of the encryption configuration. Because individual hosts within a Tungsten Clustering configuration are both servers (when acting as a master, or when providing status information), and clients (when reading remote THL and managing nodes remotely), both the server and client side of the configuration must be configured.

Configuration for all systems relies on two files, the truststore, which contains the server certificate information (the certificates it will accept from clients), and the keystore, which manages the client certificate information (the certificates that will be provided to servers). The truststore and keystore hold SSL certificate information, and are password protected.

The keystore and truststore operate by holding one or more certificates that will be used for encrypting communication. The following certificate options are available:

- Create your own server and client certificates
- Create your own server certificates, get the server certificate signed by a Certificate Authority (CA), and use a corresponding signed client certificate
- Use a server and client certificate already signed by a CA. Care should be taken with these certificates, as they are associated with specific domains and/or hosts, and may cause problems in a dynamic environment.

In a multi-node environment such as Tungsten Clustering, all the hosts in the dataservice can use the same keystore and truststore certificates. The \texttt{tpm} command will distribute these files along with the configuration when a new installation is deployed, or when updating an existing deployment.

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

Because the client and server components of the Tungsten Clustering configuration are the same, the same certificate can be used and add to both the keystore and truststore files.

The process is as follows:

1. Create the keystore and generate a certificate
2. Export the certificate
3. Import the certificate to the truststore

To start, use the supplied keytool to create a keystore and populate it with a certificate. The process asks for certain information. The alias is the name to use for the server and can be any identifier. When asked for the first and last name, use localhost, as this is used as the server identifier for the certificate. The other information should be entered accordingly.

Keystores (and truststores) also have their own passwords that are used to protect the store from updating the certificates. The password must be known as it is required in the configuration so that Tungsten Clustering can open the keystore and read the contents.

```
shell> keytool -genkey -alias replserver -keyalg RSA -keystore keystore.jks
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: localhost
What is the name of your organizational unit?
[Unknown]: My OU
What is the name of your organization?
[Unknown]: Continuent
What is the name of your City or Locality?
[Unknown]: Mountain View
What is the name of your State or Province?
[Unknown]: CA
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=localhost, OU=My OU, O=Continuent, L=Mountain View, ST=CA, C=US correct?
[no]: yes
Enter key password for <any>
(RETURN if same as keystore password):
```

The above process has created the keystore and the ‘server’ certificate, stored in the file `keystore.jks`.

Alternatively, you can create a new certificate in a keystore non-interactively by specifying the passwords and certificate contents on the command-line:

```
shell> keytool -genkey -alias replserver -keyalg RSA -keystore keystore.jks -dname "cn=localhost, ou=IT, o=Continuent, c=US" -storepass password -keypass password
```

Now you need to export the certificate so that it can be added to the truststore as the trusted certificate:

```
shell> keytool -export -alias replserver -file client.cer -keystore keystore.jks
Enter keystore password:
Certificate stored in file <client.cer>
```

This has created a certificate file in `client.cer` that can now be used to populate your truststore. When added the certificate to the truststore, it must be identified as a trusted certificate to be valid. The password for the truststore must be provided. It can be the same, or different, to the one for the keystore, but must be known so that it can be added to the Tungsten Clustering configuration.

```
shell> keytool -import -v -trustcacerts -alias replserver -file client.cer -keystore truststore.ts
Enter keystore password:
Re-enter new password:
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 {
        KeyIdentifier {
            0000: E7 D1 DB 0B 42 AC 61 84   D4 2E 9A F1 80 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`. 

50
A non-interactive version is available by using the `-noprompt` option and supplying the truststore name:

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

The two files, the keystore (`keystore.jks`), and truststore (`truststore.ts`), along with their corresponding passwords can be now be used with the `tpm` command.

### 2.7.14.2. 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
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):
```

#### Create a new signing request the certificate:

```shell
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 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
shell> keytool -import -alias replserver -file signedcert.pem -keypass password \
-keystore keystore.jks -storepass password
```

Now install the root CA certificate:

```shell
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 2.7.14.4, “Converting SSL Certificates for `keytool`” for more information.

And an intermediary certificate if you were sent one:

```shell
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
shell> keytool -export -alias replserver -file client.cer -keystore keystore.jks
Enter keystore password:
Certificate stored in file <client.cer>
```
The exported certificate and CA root and/or intermediary certificates must now be imported to the truststore:

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

• Self-Signing

If you have setup your own certificate authority, you can self-sign the request using `openssl`:

```
shell> openssl ca -in certrequest.pem -out certificate.pem
```

Convert the certificate to a plain PEM certificate:

```
shell> openssl x509 -in certificate.pem -out certificate.pem -outform PEM
```

Finally, for a self-signed certificate, you must combine the signed certificate with the CA certificate:

```
shell> cat certificate.pem cacert.pem > certfull.pem
```

This certificate can be imported into your keystore and truststore.

To import your signed certificate into your keystore:

```
shell> keytool -import -alias replserver -file certfull.pem -keypass password \
    -keystore keystore.jks -storepass password
```

Then export the certificate for use in your truststore:

```
shell> keytool -export -alias replserver -file client.cer -keystore keystore.jks
```

Enter keystore password:

Certificate stored in file <client.cer>

The same certificate must also be exported and added to the truststore:

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

This completes the setup of your truststore and keystore. The files created can be used in your `tpm` configuration.

### 2.7.14.3. 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 2.7.14.4, “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:
Deployment

```sh
deployment/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:

```sh
deployment/shell> keytool -export -alias replserver -file client.cer -keystore keystore.jks
Enter keystroke password:
Certificate stored in file <client.cer>
```

The exported certificate and CA root and/or intermediary certificates must now be imported to the truststore:

```sh
deployment/shell> keytool -import -trustcacerts -alias replserver -file client.cer \
    -keystore truststore.ts -storepass password -noprompt

deployment/shell> keytool -import -trustcacerts -alias replserver -file cacert.pem \
    -keystore truststore.ts -storepass password -noprompt

deployment/shell> keytool -import -trustcacerts -alias replserver -file intercert.pem \
    -keystore truststore.ts -storepass password -noprompt
```

2.7.14.4 Converting SSL Certificates for keytool

Some versions of the openssl toolkit generate certificates which are incompatible with the certificate mechanisms of third-party tools, even though the certificates themselves work fine with OpenSSL tools and libraries. This is due to a bug which affected certain releases of openssl 1.0.0 and later and the X.509 certificates that are created.

This problem only affects self-generated and/or self-signed certificates generated using the openssl command. Officially signed certificates from Thawte, VeriSign, or others should be compatible with keytool without conversion.

To get round this issue, the keys can be converted to a different format, and then imported into a keystore and truststore for use with Tungsten Clustering.

To convert a certificate, use openssl to convert the X.509 into PKCS12 format. You will be prompted to enter a password for the generated file which is required in the next step:

```sh
deployment/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:

```sh
deployment/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:

```sh
deployment/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

```sh
deployment/shell> keytool -importkeystore -srckeystore server.p12 -destkeystore truststore.ts -srcstoretype pkcs12
Enter destination keystore password:
Re-enter new password:
Enter source keystore password:
Entry for alias 1 successfully imported.
Import command completed:  1 entries successfully imported, 0 entries failed or cancelled
```

For official CA certificates, the generated certificate information should be valid for importing using keytool, and this file should not need conversion.
Chapter 3. Deployment: MySQL Topologies

Creating a Tungsten Clustering (for MySQL) Data Service using Tungsten Clustering combines a number of different components, systems, and functionality, to support a running database Data Service 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 Tungsten Clustering and the configuration, and optionally automatically starts the services on each host. tpm manages the entire deployment, configuration, and upgrade procedure.

- **tpm INI configuration** — tpm uses an INI to configure the service on the local host. The INI file must be created 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 Tungsten Clustering.

3.1. Deploying Standalone HA Clusters

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 datasources in the Data Service.

Figure 3.1. Topologies: Standalone HA Cluster

3.1.1. Prepare: Standalone HA Cluster

Before continuing with deployment you will need the following:

1. The name to use for the cluster.
2. The list of datasources in the cluster. These are the servers which will be running MySQL.
3. The list of servers that will run the connector.
4. The username and password of the MySQL replication user.
5. The username and password of the first application user. You may add more users after installation.

All servers must be prepared with the proper prerequisites. See Section 2.6, “Prepare Hosts” and Appendix B, Prerequisites for additional details.

3.1.2. Install: Standalone HA Cluster

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

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-clustering-6.1.3-43.tar.gz
```

2. Change to the Tungsten Clustering directory:

```
shell> cd tungsten-clustering-6.1.3-43
```

3. Run `tpm` to perform the installation, using either the staging method or the INI method. Review Section 9.1, “Comparing Staging and INI tpm Methods” for more details on these two methods.

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

Show Staging

Show INI

```
shell> ./tools/tpm configure defaults \
    --reset \
    --user=tungsten \
    --install-directory=/opt/continuent \
    --profile-script=~/.bash_profile \
    --replication-user=tungsten \
    --replication-password=password \
    --replication-port=13306 \
    --application-user=app_user \
    --application-password=secret \
    --application-port=3306 \
    --start-and-report=true
```

```
shell> ./tools/tpm configure alpha \
    --topology=clustered \
    --master=host1 \
    --members=host1,host2,host3 \
    --connectors=host4
```

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

```
[defaults]
user=tungsten
install-directory=/opt/continuent
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=password
replication-port=13306
application-user=app_user
application-password=secret
application-port=3306
start-and-report=true

[alpha]
topology=clustered
master=host1
members=host1,host2,host3
connectors=host4
```

Configuration group defaults

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

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

- `--reset` [470]

  reset [470]
For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- `--user=tungsten` [477]
  
  *System User*

- `--install-directory=/opt/continuent` [456]
  
  *Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.*

- `--profile-script=~/.bash_profile` [467]
  
  *Append commands to include env.sh in this profile script*

- `--replication-user=tungsten` [470]
  
  *For databases that required authentication, the username to use when connecting to the database using the corresponding connection method [native, JDBC, etc.].*

  - `--replication-password=password` [469]
  
    *The password to be used when connecting to the database using the corresponding `--replication-user` [470].*

  - `--replication-port=13306` [469]
  
    *The network port used to connect to the database server. The default port used depends on the database being configured.*

- `--application-user=app_user` [437]
  
  *Database username for the connector*

- `--application-password=secret` [437]
  
  *Database password for the connector*

  - `--application-port=3306` [437]
  
    *Port for the connector to listen on*

- `--start-and-report=true` [472]
  
  *Start the services and report out the status after configuration*

**Configuration group: `alpha`**

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

- `--topology=clustered` [476]
topology=clustered

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

- --master=host1
- master=host1

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

- --members=host1,host2,host3
- members=host1,host2,host3

Hostnames for the dataservice members

- --connectors=host4
- connectors=host4

Hostnames for the dataservice connectors

Run tpm to install the software with the configuration.

```shell
dir ./tools/tpm install
```

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

4. Initialize your PATH and environment.

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

**Important**

Do not include start-and-report if you are taking over for MySQL native replication. See Section 3.10.1, "Migrating from MySQL Native Replication 'In-Place'" for next steps after completing installation.

### 3.1.3. Best Practices: Standalone HA Cluster

Follow the guidelines in Section 2.5, "Best Practices".

### 3.2. Deploying Composite Master/Slave Clustering

Tungsten Clustering supports the creation of composite clusters. This includes multiple master/slave dataservices tied together. One of the dataservices is identified as the master and all other dataservices replicate from it.
3.2.1. Prepare: Composite Master/Slave Cluster

Before continuing with deployment you will need the following:

1. The cluster name for each Master/Slave Cluster and a Composite cluster name to group them.
2. The list of datasources in each cluster. These are the servers which will be running MySQL.
3. The list of servers that will run the connector. Each connector will be associated with a preferred cluster but will have access to the master regardless of location.
4. The username and password of the MySQL replication user.
5. The username and password of the first application user. You may add more users after installation.

All servers must be prepared with the proper prerequisites. See Section 2.6, "Prepare Hosts" and Appendix B, Prerequisites for additional details.

3.2.2. Install: Composite Master/Slave Cluster

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

   
   ```shell
   cd /opt/continuent/software
   tar zxf tungsten-clustering-6.1.3-43.tar.gz
   ```

2. Change to the Tungsten Clustering directory:

   ```shell
   cd tungsten-clustering-6.1.3-43
   ```

3. Run `tpm` to perform the installation. This method assumes you are using the Section 9.3, "tpm Staging Configuration" method:

   Click the link below to switch examples between Staging and INI methods
   
   Show Staging
Deployment: MySQL Topologies

Show INI

```
shell> ./tools/tpm configure defaults
    --reset
    --user=tungsten
    --install-directory=/opt/continuent
    --profile-script=~/.bash_profile
    --replication-user=tungsten
    --replication-password=secret
    --replication-port=13306
    --application-user=app_user
    --application-password=secret
    --application-port=3306
    --start-and-report

shell> ./tools/tpm configure alpha
    --topology=clustered
    --master=host1.alpha
    --members=host1.alpha,host2.alpha,host3.alpha
    --connectors=host1.alpha,host2.alpha,host3.alpha

shell> ./tools/tpm configure beta
    --topology=clustered
    --relay=host1.beta
    --members=host1.beta,host2.beta,host3.beta
    --connectors=host1.beta,host2.beta,host3.beta
    --relay-source=alpha

shell> ./tools/tpm configure gamma
    --composite-datasources=alpha,beta

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

```
[defaults]
user=tungsten
install-directory=/opt/continuent
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=secret
replication-port=13306
application-user=app_user
application-password=secret
application-port=3306
start-and-report

[alpha]
topology=clustered
master=host1.alpha
members=host1.alpha,host2.alpha,host3.alpha
connectors=host1.alpha,host2.alpha,host3.alpha

[beta]
topology=clustered
relay=host1.beta
members=host1.beta,host2.beta,host3.beta
connectors=host1.beta,host2.beta,host3.beta
relay-source=alpha

[gamma]
composite-datasources=alpha,beta
```

Configuration group defaults

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

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

-   **--reset** [470]

```
    reset [470]
```

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

-   **--user=tungsten** [477]

```
    user=tungsten [477]
```

System User
Deployment: MySQL Topologies

- `--install-directory=/opt/continuent`
  
  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--profile-script=~/.bash_profile`
  
  Append commands to include env.sh in this profile script

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

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

- `--replication-port=13306`
  
  The network port used to connect to the database server. The default port used depends on the database being configured.

- `--application-user=app_user`
  
  Database username for the connector

- `--application-password=secret`
  
  Database password for the connector

- `--application-port=3306`
  
  Port for the connector to listen on

- `--start-and-report`
  
  Start the services and report out the status after configuration

Configuration group `alpha`

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

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

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

- `--master=host1.alpha`
  
  master=host1.alpha
The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- \texttt{--members=host1.alpha,host2.alpha,host3.alpha}\ [460]

  Hostnames for the dataservice members

- \texttt{--connectors=host1.alpha,host2.alpha,host3.alpha}\ [444]

  Hostnames for the dataservice connectors

Configuration group \textit{beta}

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

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

- \texttt{--topology=clustered}\ [476]

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

- \texttt{--relay=host1.beta}\ [459]

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

- \texttt{--members=host1.beta,host2.beta,host3.beta}\ [460]

  Hostnames for the dataservice members

- \texttt{--connectors=host1.beta,host2.beta,host3.beta}\ [444]

  Hostnames for the dataservice connectors

- \texttt{--relay-source=alpha}\ [469]

  Dataservice name to use as a relay source

Configuration group \textit{gamma}

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

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

- \texttt{--composite-datasources=alpha,beta}\ [441]

  Data services that should be added to this composite data service

Run \texttt{tpm} to install the software with the configuration.

```
shell $ tools/tpm install
```

During the startup and installation, \texttt{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.
shell > source /opt/continuent/share/env.sh

The Composite Master/Slave Cluster should be installed and ready to use.

3.2.3. Best Practices: Composite Master/Slave Cluster

Follow the guidelines in Section 2.5, “Best Practices”.

3.3. Deploying Multisite/Multimaster Clustering

Warning

From v6 of Tungsten Clustering the Multimaster topology has been overhauled. Whilst the Multisite/Multimaster topology is still valid and supported, it is recommended that you use the newer Composite Multimaster topology from v6 onwards.

Please refer to Section 3.4, “Deploying Composite Multimaster Clustering”

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 Tungsten Clustering uses the Tungsten Replicator System of Record (SOR) service, which enables multimaster operation between the two sites.

The configuration is in two separate parts:

• Tungsten Clustering 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 3.3, “Topologies: Multisite/Multimaster Clusters”.
The service can be described as follows:

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

- **Tungsten Clustering 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 Tungsten Clustering 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 Tungsten Clustering and replicated to the west Tungsten Clustering dataservice. The configuration allows for changes in the Tungsten Clustering 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 Tungsten Clustering 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 Tungsten Clustering and replicated to the east Tungsten Clustering dataservice. The configuration allows for changes in the Tungsten Clustering 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 for Multisite/Multimaster deployments are Tungsten Clustering 5.4.x and Tungsten Replicator 5.4.x

3.3.1. Prepare: Multisite/Multimaster Clusters

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:

```shell
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. Update your configuration file to explicitly use row-based replication by adding the following to your my.cnf file:

```shell
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. Tungsten Clustering 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 C.3.1, “Triggers”.

3.3.2. Install: Multisite/Multimaster Clusters

**Note**

The procedures in this section are designed for the Multisite/Multimaster topology ONLY. Do NOT use these procedures for Composite Multimaster Clustering using v6 onwards.

For version 6.x Composite Multimaster Clustering Clustering, please refer to Section 3.4, “Deploying Composite Multimaster Clustering”

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

1. Install the Tungsten Clustering and Tungsten Replicator packages or download the tarballs, and unpack them:

```shell
shell> cd /opt/continuent/software
shell> tar zxf tungsten-clustering-6.1.3-43.tar.gz
shell> tar zxf tungsten-replicator-6.1.3-43.tar.gz
```

2. Change to the Tungsten Clustering directory:

```shell
shell> cd tungsten-clustering-6.1.3-43
```

3. Run tpm to configure the installation. This method assumes you are using the Section 9.3, “tpm Staging Configuration” method:

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

For ini install, the ini file contains all the configuration for both the cluster deployment and the replicator deployment.

For a staging install, you first use the cluster configuration show below and then configure the replicator as a separate process. These additional steps are outlined below

Show Staging

Show INI

```shell
shell> ./tools/tpm configure defaults
   --reset
   --user=tungsten
   --install-directory/opt/continuent
   --replication-user=tungsten
```
```
deployment-password=secret 
--replication-port=3306 
--profile-script=~/bashrc 
--application-user=app_user 
--application-password=secret 
--skip-validation-check=MySQLPermissionsCheck 
--start-and-report
```

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

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

```
shell> vi /etc/tungsten/tungsten.ini
```n
```
[defaults]
user=tungsten
install-directory=/opt/continuent
replication-user=tungsten
replication-password=secret
replication-port=3306
profile-script=~/bashrc
application-user=app_user
application-password=secret
skip-validation-check=MySQLPermissionsCheck
start-and-report
```

```
[defaults.replicator]
home-directory=/opt/replicator
rmi-port=10002
executable-prefix=mm
```

```
[east]
topology=clustered
connectors=east1,east2,east3
master=east1
members=east1,east2,east3
```

```
[west]
topology=clustered
connectors=west1,west2,west3
master=west1
members=west1,west2,west3
```

```
[east_west]
topology=cluster-slave
master-data-service=east
slave-data-service=west
thl-port=2113
```

```
[west_east]
topology=cluster-slave
master-data-service=west
slave-data-service=east
thl-port=2115
```

Configuration group **defaults**

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

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

- **--reset [470]**

  reset [470]

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

- **--user=tungsten [477]**

  user=tungsten [477]

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

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

• **--replication-user=tungsten**

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

• **--replication-password=secret**

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

• **--replication-port=3306**

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

• **--profile-script=~/.bashrc**

Append commands to include env.sh in this profile script

• **--application-user=app_user**

Database username for the connector

• **--application-password=secret**

Database password for the connector

• **--skip-validation-check=MySQLPermissionsCheck**

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

  **Warning**

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

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

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

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

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

• **--start-and-report**

It
Deployment: MySQL Topologies

Start the services and report out the status after configuration

Configuration group defaults.replicator

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.

- **--home-directory=/opt/replicator**
  
  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- **--rmi-port=10002**
  
  Replication RMI listen port

- **--executable-prefix=mm**
  
  When enabled, the supplied prefix is added to each command alias that is generated for a given installation. This enables multiple installations to co-exist and be accessible through a unique alias. For example, if the executable prefix is configured as `east`, then an alias for the installation to `trepctl` will be created as `east_trepctl`.

  Alias information for executable prefix data is stored within the `$CONTINUENT_ROOT/share/aliases.sh` file for each installation.

Configuration group east

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

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

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

- **--connectors=east1,east2,east3**
  
  Hostnames for the dataservice connectors

- **--master=east1**
  
  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- **--members=east1,east2,east3**
  
  Hostnames for the dataservice members

Configuration group west

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

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

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

- `--connectors=west1,west2,west3` [444]
  
  Connectors = west1, west2, west3 [444]

  Hostnames for the dataservice connectors:

  - `--master=west1` [459]
    
    Master = west1 [459]

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

  - `--members=west1,west2,west3` [460]
    
    Members = west1, west2, west3 [460]

    Hostnames for the dataservice members

Configuration group `east_west`

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

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

- `--topology=cluster-slave` [476]
  
  Topology = cluster-slave [476]

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

- `--master-dataservice=east` [469]
  
  Master-dataservice = east [469]

  Dataservice name to use as a relay source.

- `--slave-dataservice=west` [474]
  
  Slave-dataservice = west [474]

  Dataservice to use to determine the value of host configuration.

- `--thl-port=2113` [476]
  
  Thl-port = 2113 [476]

  Port to use for THL Operations.

Configuration group `west_east`

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

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

- `--topology=cluster-slave` [476]
  
  Topology = cluster-slave [476]

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

- `--master-dataservice=west` [469]
  
  Master-dataservice = west [469]

  Dataservice name to use as a relay source.

- `--slave-dataservice=east` [474]
  
  Slave-dataservice = east [474]
Dataservice to use to determine the value of host configuration

- `--thl-port=2115`

Port to use for THL Operations

Run `tpm` to install the software with the configuration.

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

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

4. Change to the Tungsten Replicator directory:

```
shell> cd tungsten-replicator-6.1.3-43
```

5. Run `tpm` to configure the installation. This method assumes you are using the Section 9.3, “tpm Staging Configuration” method:

6. If you are running a staging install, first configure the replicator using the following example, if configuring using an ini file, skip straight to the install step below

```
shell> ./tools/tpm configure defaults \
    --reset \n    --user=tungsten \n    --install-directory=/opt/replicator \n    --replication-user=tungsten \n    --replication-password=secret \n    --profile-script=/bashrc \n    --application-user=app_user \n    --application-password=secret \n    --skip-validation-check=MySQLPermissionsCheck \n    --rmi-port=10002 \n    --executable-prefix=mm \n    --thl-port=2113 \n    --start-and-report
```

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

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

```
shell> ./tools/tpm configure east_west \n    --topology=cluster-slave \n    --master-dataservice=east \n    --slave-dataservice=west \n    --thl-port=2113
```

```
shell> ./tools/tpm configure west_east \n    --topology=cluster-slave \n    --master-dataservice=west \n    --slave-dataservice=east \n    --thl-port=2115
```

7. Run `tpm` to install the software with the configuration.

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

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

8. Initialize your `PATH` and environment.

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

The MSMM clustering should be installed and ready to use.
3.3.3. Best Practices: Multisite/Multimaster Clusters

Note
The procedures in this section are designed for the Multisite/Multimaster topology ONLY. Do NOT use these procedures for Composite Multimaster Clustering using v6 onwards.

For version 6.x Composite Multimaster Clustering, please refer to Section 3.4, "Deploying Composite Multimaster Clustering"

Note
In addition to this information, follow the guidelines in Section 2.5, "Best Practices".

- Running a Multisite/Multimaster service uses many different components to keep data updated on all servers. Monitoring the dataservice is divided into monitoring the two different clusters. Be mindful when using commands that you have the correct path. You should either use the full path to the command under `/opt/continuent` and `/opt/replicator`, or use the aliases created by setting the `--executable-prefix=m-` option. Calling `trepctl` would become `mm_trepctl`.

- Configure your database servers with distinct `auto_increment_increment` and `auto_increment_offset` settings. Each location that may accept writes should have a unique offset value.

Using `cctrl` gives you the dataservice status individually for the east and west dataservice. For example, the east dataservice is shown below:

```
Continuent Tungsten 6.1.3 build 43
east: session established
[LOGICAL] /east >

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=8.722)                 |
| [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=1.141)                 |
| [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 Tungsten Clustering service east, and a Tungsten Replicator service east that reads data from the west Tungsten Clustering service. A corresponding west Tungsten Clustering and west Tungsten Replicator service.

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

- When data is inserted on east, the sequence number of the east Tungsten Clustering service and east Tungsten Replicator service [on west{1,2,3}] should be compared.
• When data is inserted on the master within the east Tungsten Clustering, use `cctrl` to determine the cluster status. Sequence numbers within the Tungsten Clustering east should match, and latency between hosts in the Tungsten Clustering service are relative to each other.

• When data is inserted on west, the sequence number of the west Tungsten Clustering service and west Tungsten Replicator service (on east(1,2,3)) should be compared.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Tungsten Clustering Service Seqno</th>
<th>Tungsten Replicator Service Seqno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert/update data on east</td>
<td>east Seqno</td>
<td>west Seqno</td>
</tr>
<tr>
<td>Insert/update data on west</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 5.3, "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 Tungsten Clustering 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>100.161</td>
</tr>
<tr>
<td>east3</td>
<td>east</td>
<td>master</td>
<td>ONLINE</td>
<td>44</td>
<td>0.697</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>west1</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>53</td>
<td>119.934</td>
</tr>
<tr>
<td>west2</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>53</td>
<td>181.128</td>
</tr>
<tr>
<td>west3</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>53</td>
<td>204.790</td>
</tr>
<tr>
<td>west1</td>
<td>west</td>
<td>master</td>
<td>ONLINE</td>
<td>294327</td>
<td>0.285</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>east1</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>294327</td>
<td>0.879</td>
</tr>
<tr>
<td>east2</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>294327</td>
<td>0.567</td>
</tr>
<tr>
<td>east3</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>2494327</td>
<td>1.046</td>
</tr>
<tr>
<td>west3</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>231595</td>
<td>22.895</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 Tungsten Clustering service:

```
shell> trepctl status
```

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

```
shell> mm_trepctl -service west status
```

### 3.3.4. Configuring Startup on Boot

**Note**

The procedures in this section are designed for the Multisite/Multimaster topology ONLY. Do NOT use these procedures for Composite Multimaster Clustering using v6 onwards.

For version 6.x Composite Multimaster Clustering, please refer to Section 3.4, "Deploying Composite Multimaster Clustering".

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

• For the Tungsten Clustering service, use Section 4.3, "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 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):

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

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

```bash
APP_NAME="mmreplicator"
```

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

On Debian/Ubuntu:

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

On RedHat/CentOS:

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

3.3.5. Resetting a single dataservice

**Note**

The procedures in this section are designed for the Multisite/Multimaster topology ONLY. Do NOT use these procedures for Composite Multimaster Clustering using v6 onwards.

For version 6.x Composite Multimaster Clustering, please refer to Section 3.4, “Deploying Composite Multimaster Clustering”

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 Tungsten Clustering will not attempt to automatically recover the service.

```bash
cctrl [east]> set policy maintenance
```

2. On the `east`, failed, Tungsten Clustering service, put each Tungsten Connector offline:

```bash
cctrl [east]> router * offline
```

3. Reset the failed Tungsten Replicator service on all servers connected to the failed Tungsten Clustering service. For example, on `west{1,2,3}` reset the `east` Tungsten Replicator service:

```bash
shell west1> /opt/replicator/tungsten-replicator/bin/trepctl -service east offline
shell west1> /opt/replicator/tungsten-replicator/bin/trepctl -service east reset -all -y
```

4. Reset the Tungsten Clustering service on each server within the failed region (`east{1,2,3}`):

```bash
shell east1> /opt/continuent/tungsten-replicator/bin/replicator stop
shell east1> /opt/continuent/tungsten-replicator/bin/tpm reset east
shell east1> /opt/continuent/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:

```bash
shell east1> /opt/continuent/tungsten-replicator/scripts/tungsten_provision_slave \
    --direct --source=west1
```

6. Place all the Tungsten Replicator services on `west{1,2,3}` back online:

```bash
shell west1> /opt/replicator/tungsten-replicator/bin/trepctl -service east online
```

7. On the `east`, failed, Tungsten Clustering service, put each Tungsten Connector online:

```bash
cctrl [east]> router * online
```

8. Set the policy back to automatic:

```bash
cctrl> set policy automatic
```
3.3.6. Resetting all dataservices

**Note**

The procedures in this section are designed for the Multisite/Multimaster topology ONLY. Do NOT use these procedures for Composite Multimaster Clustering using v6 onwards.

For version 6.x Composite Multimaster Clustering, please refer to Section 3.4, “Deploying Composite Multimaster Clustering”

To reset all of the dataservices and restart the Tungsten Clustering 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
shell> /opt/continuent/tungsten/tools/tpm reset
shell> /opt/replicator/tungsten-tungsten-replicator/bin/replicator start
```

3.3.7. Provisioning during live operations

**Note**

The procedures in this section are designed for the Multisite/Multimaster topology ONLY. Do NOT use these procedures for Composite Multimaster Clustering using v6 onwards.

For version 6.x Composite Multimaster Clustering, please refer to Section 3.4, “Deploying Composite Multimaster Clustering”

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 5.10, “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/replicator/tungsten-tungsten-replicator/tools/tpm reset
      shell> /opt/continuent/tungsten-tungsten-replicator/scripts/tungsten_provision_slave --direct --source=west1
      ```

  4. Check that Tungsten Clustering 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
      ```

3.3.8. Adding a new Cluster/Dataservice

**Note**

The procedures in this section are designed for the Multisite/Multimaster topology ONLY. Do NOT use these procedures for Composite Multimaster Clustering using v6 onwards.

For version 6.x Composite Multimaster Clustering, please refer to Section 3.4, “Deploying Composite Multimaster Clustering”
To add an entirely new cluster (dataservice) to the mesh, follow the below simple procedure.

**Note**

There is no need to set the Replicator starting points, and no downtime/maintenance window is required!

1. Choose a cluster to take a node backup from:
   - Choose a cluster and slave node to take a backup from.
   - Enable maintenance mode for the cluster:
     ```shell
cctrl
   cctrl> set policy maintenance
   
   ```
   - Shun the selected slave node and stop both local and cross-site replicator services:
     ```shell
   cctrl
cctrl> datasource {slave_hostname_here} shun
   slave shell>
   trepctl offline
   slave shell>
   mm_trepctl offline
   slave shell>
   mm_replicator stop
   
   ```
   - Take a backup of the shunned node, then copy to/restore on all nodes in the new cluster.
   - Recover the slave node and put cluster back into automatic mode:
     ```shell
   slave shell>
   replicator start
   slave shell>
   trepctl online
   slave shell>
   mm_replicator start
   slave shell>
   mm_trepctl online
   cctrl
cctrl> datasource {slave_hostname_here} online
   cctrl>
   set policy automatic
   
   ```

2. On ALL nodes in all three (3) clusters, ensure the `/etc/tungsten/tungsten.ini` has all three clusters defined and all the correct cross-site combinations.

3. Install the Tungsten Clustering software on new cluster nodes to create a single standalone cluster and check the `cctrl` command to be sure the new cluster is fully online.

4. Install the Tungsten Replicator software on all new cluster nodes and start it.
   Replication will now be flowing INTO the new cluster from the original two.

5. On the original two clusters, run `tools/tpm update` from the cross-site replicator staging software path:
   ```shell
   mm_tpm query staging
cd {replicator_staging_directory}
tools/tpm update --replace-release
   mm_trepctl online
   mm_trepctl services
   
   ```
   Check the output from the `mm_trepctl services` command output above to confirm the new service appears and is online.

**Note**

There is no need to set the cross-site replicators at a starting position because:
- Replicator feeds from the new cluster to the old clusters start at seqno 0.
- The tungsten_olda and tungsten_oldb database schemas will contain the correct starting points for the INBOUND feed into the new cluster, so when the cross-site replicators are started and brought online they will read from the tracking table and carry on correctly from the stored position.

### 3.3.9. Enabling SSL for Replicators Only

**Note**

The procedures in this section are designed for the Multisite/Multimaster topology ONLY. Do NOT use these procedures for Composite Multimaster Clustering using v6 onwards.

For version 6.x Composite Multimaster Clustering, please refer to Section 3.4, “Deploying Composite Multi-master Clustering”
It is possible to enable secure communications for just the Replicator layer in a MSMM topology. This would include both the Cluster Replicators and the Cross-Site Replicators because they cannot be SSL-enabled independently.

1. Create a certificate and load it into a java keystore, and then load it into a truststore and place all files into the `/etc/tungsten/` directory. For detailed instructions, see Section 2.7.14.1, “Creating Your Own Client and Server Certificates”

2. Update `/etc/tungsten/tungsten.ini` to include these additional lines in the both the `defaults` section and the `defaults.replicator` section:

   ```
   [defaults]
   ...
   java-keystore-path=/etc/tungsten/keystore.jks
   java-keystore-password=secret
   java-truststore-path=/etc/tungsten/truststore.ts
   java-truststore-password=secret
   thl-ssl=true

   [defaults.replicator]
   ...
   java-keystore-path=/etc/tungsten/keystore.jks
   java-keystore-password=secret
   java-truststore-path=/etc/tungsten/truststore.ts
   java-truststore-password=secret
   thl-ssl=true
   ```

3. Put all clusters into maintenance mode.

   ```
   cctrl>
   set policy maintenance
   ```

4. On all hosts, update the cluster configuration:

   ```
   tpm query staging
   cd {cluster_staging_directory}
   tools/tpm update
trepctl online
trepctl status | grep thl
   ```

   On all hosts, update the cross-site replicator configuration:

   ```
   mm_tpm query staging
   cd {replicator_staging_directory}
   tools/tpm update
   mm_trepctl online
   mm_trepctl status | grep thl
   ```

   **Important**
   Please note that all replication will effectively be down until all nodes/services are SSL-enabled and online.

5. Once all the updates are done and the Replicators are back up and running, use the various commands to check that secure communications have been enabled.

   Each datasource will show `[SSL]` when enabled:

   ```
   cctrl>
   ls
   ```

   ```
   DATA SOURCES:
   ------------------------------
   | [SSL] | [OK] [2018/04/19 11:18:21 PM UTC] |
   | -------- | -------- | -------- | -------- | -------- |
   | db1(master:ONLINE, progress=208950063, THL latency=0.920) | MANAGER(state=ONLINE) | REPLICATOR(role=master, state=ONLINE) | DATASERVER(state=ONLINE) | CONNECTIONS(created=1, active=0) |
   | [SSL] | [OK] [2018/04/19 11:18:21 PM UTC] |
   | -------- | -------- | -------- | -------- | -------- |
   | db2(slave:ONLINE, progress=208950061, latency=0.939) | MANAGER(state=ONLINE) | REPLICATOR(role=slave, master=db1, state=ONLINE) | DATASERVER(state=ONLINE) | CONNECTIONS(created=8, active=0) |
   | [SSL] | [OK] [2018/04/19 11:18:21 PM UTC] |
   | -------- | -------- | -------- | -------- | -------- |
   | db3(slave:ONLINE, progress=208950063, latency=0.939) | MANAGER(state=ONLINE) | REPLICATOR(role=slave, master=db2, state=ONLINE) | DATASERVER(state=ONLINE) | CONNECTIONS(created=1, active=0) |
   ```
Both the local cluster replicator status command `trepctl status` and the cross-site replicator status command `mm_trepctl status` will show `thls` instead of `thl` in the values for `masterConnectUri`, `masterListenUri` and `pipelineSource`.

```
shell> trepctl status | grep thl
masterConnectUri : thls://db1:2112/
masterListenUri  : thls://db5:2112/
pipelineSource   : thls://db1:2112/
```

### 3.3.10. Dataserver maintenance

**Note**

The procedures in this section are designed for the Multisite/Multimaster topology ONLY. Do NOT use these procedures for Composite Multimaster Clustering using v6 onwards.

For version 6.x Composite Multimaster Clustering, please refer to Section 3.4, "Deploying Composite Multimaster Clustering”.

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 5.15, “Performing Database or OS Maintenance”, except that you must also restart the corresponding Tungsten Replicator service after the main Tungsten Clustering 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 Tungsten Clustering 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
```

### 3.3.10.1. Fixing Replication Errors

In the event of a replication fault, the standard `cctrl`, `trepctl` and other utility commands in Chapter 8, 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:

```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.
3.4. Deploying Composite Multimaster Clustering

A Composite Multimaster Cluster 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 Tungsten Clustering uses two services within each node; one provides the replication within the cluster, and the second provides replication from the remote cluster. Both are managed by the Tungsten Manager.

**Note**

Multimaster Clusters were previously referred to as Multisite/Multimaster clusters. The name has been updated to reflect the nature of these clusters as part of an overall multimaster deployment using clusters, where the individual clusters could be in the same or different locations.

Whilst the older Multisite/Multimaster topology is still valid and supported, it is recommended that this newer Composite Multimaster topology is adopted from version 6 of Tungsten Clustering onwards. For details on the older topology, see Section 3.3, “Deploying Multisite/Multimaster Clustering”.

The configuration is handled with a single configuration and deployment that configures the core cluster services and additional cross-cluster services.

A sample display of how this operates is provided in Figure 3.4, “Topologies: Composite Multimaster Clusters”.

Figure 3.4. Topologies: Composite Multimaster Clusters

The service can be described as follows:

- **Tungsten Clustering Service: east**
  
  Replicates data between east1, east2 and east3 [not shown].

- **Tungsten Clustering Service: west**
  
  Replicates data between west1, west2 and west3 [not shown].
• Tungsten Clustering Service: west_from_east

Defines the replication of data within east as a replicator service using a secondary service within the cluster deployed Tungsten Replicator. This service reads from all the hosts within the Tungsten Clustering service east and writes to west1, west2, and west3.

• Tungsten Replicator Service: east_from_west

Defines the replication of data within west as a replicator service using a secondary service within the cluster deployed Tungsten Replicator. This service reads from all the hosts within the Tungsten Clustering service west and writes to east1, east2, and east3.

3.4.1. Prepare: Composite Multimaster Clusters

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. 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. Tungsten Clustering 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 C.3.1, “Triggers”.

3.4.2. Install: Composite Multimaster Clusters

Deployment of composite multimaster clusters is only supported using the INI method of deployment.

Configuration and deployment of the cluster works as follows:

• Creates two basic master/slave clusters.

• Creates a composite service that includes the master/slave clusters within the definition.

The resulting configuration within the example builds the following deployment:

• One cluster, east, with three hosts.

• One cluster, west, with three hosts.

• All six hosts in the two clusters will have a manager, replicator and connector installed.

• Each replicator has two replication services, one service that replicates the data within the cluster. The second service, replicates data from the other cluster to this host.

Creating the full topology requires a single install step, this creates the Tungsten Clustering cluster dataservices, and creates the Composite dataservices on different network ports to allow for the cross-cluster replication to operate.

1. Create the combined configuration file /etc/tungsten/tungsten.ini on all cluster hosts:

```
[defaults]
user=tungsten
install-directory=/opt/continuent
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=secret
replication-port=13306
application-user=app_user
application-password=secret
application-port=3306
start-and-report

[east]
topology=clustered
master=east1
members=east1,east2,east3
```
Deployment: MySQL Topologies

connectors=east1,east2,east3

case west
  when topology=clustered
    master=west1
    members=west1,west2,west3
    connectors=west1,west2,west3
end

case usa
  when topology=composite-multimaster
    composite-datasources=east,west
end

Configuration group defaults

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

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

• user=tungsten [477]
  System User

• install-directory=/opt/continuent [456]
  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

• profile-script=~/.bash_profile [467]
  Append commands to include env.sh in this profile script

• replication-user=tungsten [470]
  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method [native, JDBC, etc.].

• replication-password=secret [469]
  The password to be used when connecting to the database using the corresponding --replication-user.

• replication-port=13306 [469]
  The network port used to connect to the database server. The default port used depends on the database being configured.

• application-user=app_user [437]
  Database username for the connector

• application-password=secret [437]
  Database password for the connector

• application-port=3306 [437]
  Port for the connector to listen on

• start-and-report [472]
  Start the services and report out the status after configuration

Configuration group east

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

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

• topology=clustered [476]
  Replication topology for the dataservice Valid values are star,cluster-slave,master-slave,fan-in,clustered,cluster-alias,all-masters,direct

• master=east1 [459]
  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.
Deployment: MySQL Topologies

- **members=east1,east2,east3**
  Hostnames for the dataservice members

- **connectors=east1,east2,east3**
  Hostnames for the dataservice connectors

Configuration group **west**

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

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

- **master=west1**
  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- **members=west1,west2,west3**
  Hostnames for the dataservice members

- **connectors=west1,west2,west3**
  Hostnames for the dataservice connectors

Configuration group **usa**

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

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

- **composite-datasources=east,west**
  Data services that should be added to this composite data service

The configuration above defines two clusters, **east** and **west**, which are both part of a composite multimaster cluster service, **usa**. Configuration can be divided up into the four sections shown, as follows:

**Warning**

Service names should not contain the keyword `from` within a composite multimaster deployment. This keyword is used (with the underscore separator, for example, `east_from_west`) to denote cross-site replicators within the cluster. To avoid confusion, avoid using `from` so that it is easy to distinguish between replication pipelines.

When configuring this service, **tpm** will automatically imply the following into the configuration:

- A parent composite service, **usa** in this example, with child services as listed, **east** and **west**.

- Replication services between each child service, using the service name `a_from_b`, for example, `east_from_west` and `west_from_east`.

  More child services will create more automatic replication services. For example, with three clusters, **alpha**, **beta**, and **gamma**, **tpm** would configure `alpha_from_beta` and `alpha_from_gamma` on the alpha cluster, `beta_from_alpha` and `beta_from_gamma` on the beta cluster, and so on.

- For each additional service, the port number is automatically configured from the base port number for the first service. For example, using the default port 2112, the `east_from_west` service would have THL port 2113.

2. Execute the installation on each host within the entire composite cluster. For example, on all six hosts provided in the sample configuration above.

a. Install the Tungsten Clustering package `.rpm`, or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
```
b. Change to the Tungsten Clustering staging directory:

```
shell> cd tungsten-clustering
```

c. Run `tpm` to install the Clustering software:

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

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

3. Initialize your `PATH` and environment:

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

The Composite Multimaster clustering should be installed and ready to use.

### 3.4.3. Best Practices: Composite Multimaster Clusters

**Note**

In addition to this information, follow the guidelines in Section 2.5, “Best Practices”.

- Running a Composite Multimaster service uses many different components to keep data updated on all servers. Monitoring the dataser-vice is divided into monitoring the two different clusters and each cluster sub-service cluster responsible for replication to/from remote clusters.
- Configure your database servers with distinct `auto_increment_increment` and `auto_increment_offset` settings. Each location that may accept writes should have a unique offset value.

Using `cctrl` gives you the dataservice status. By default, `cctrl` will connect you to the custer associated with the node that you issue the command from. To start at the top level, issue `cctrl -multi` instead.

At the top level, the composite cluster output shows the composite service, composite cluster members and replication services:

```
Tungsten Clustering 6.1.3 build 43
east: session established, encryption=false, authentication=false
[LOGICAL] / > ls
usa
east
east_from_west
west
```

To examine the overall composite cluster status, change to the composite cluster and use `ls`:

```
[LOGICAL] /usa > ls
COORDINATOR[west3:AUTOMATIC:ONLINE]
east:COORDINATOR[east3:AUTOMATIC:ONLINE]
west:COORDINATOR[west3:AUTOMATIC:ONLINE]
```

```
ROUTERS:
```

```
|connector@east1[10583](ONLINE, created=0, active=0) | |
|connector@east2[10548](ONLINE, created=0, active=0) | |
|connector@east3[10540](ONLINE, created=0, active=0) | |
|connector@west1[10589](ONLINE, created=0, active=0) | |
|connector@west2[10541](ONLINE, created=0, active=0) | |
|connector@west3[10547](ONLINE, created=0, active=0) | |
```

```
DATASOURCES:
```

```
|east(composite master:ONLINE, global progress=1, max latency=3.489) | |
|STATUS [OK] [2019/12/24 10:21:08 AM UTC] | |
```

```
|east_from_west(relay:ONLINE, progress=1, max latency=3.489) | |
```

```
|west(composite master:ONLINE, global progress=1, max latency=0.989) | |
|STATUS [OK] [2019/12/24 10:21:08 AM UTC] | |
```

```
|west_from_east(relay:ONLINE, progress=1, max latency=0.983) | |
```

```
Deployment: MySQL Topologies

For each cluster within the composite cluster, four lines of information are provided:

- **east(composite master:ONLINE, global progress=1, max latency=3.489)**
  
  This line indicates:
  
  - The name and type of the composite cluster, and whether the master in the cluster is online.
  - The global progress. This is a counter that combines the local progress of the cluster, and the replication of data from this cluster to the remote clusters in the composite to this cluster. For example, if data is inserted into **west**
  - The maximum latency within the cluster.

- **[STATUS [OK] [2019/12/24 10:21:08 AM UTC]]**
  
  The status and date within the master of the cluster.

- **east(master:ONLINE, progress=1, max latency=1.483)**
  
  The status and progress of the cluster.

- **east_from_west(relay:ONLINE, progress=1, max latency=3.489)**
  
  The status and progress of remote replication from the cluster.

The **global progress** and the **progress** work together to provide an indication of the overall replication status within the composite cluster:

- Inserting data into the master on **east** will:
  - Increment the **progress** within the **east** cluster.
  - Increment the **global progress** within the **east** cluster.

- Inserting data into the master on **west** will:
  - Increment the **progress** within the **west** cluster.
  - Increment the **global progress** within the **west** cluster.

Looking at the individual cluster shows only the cluster status, not the cross-cluster status:

```
[LOGICAL] /east > ls
COORDINATOR[east1:AUTOMATIC:ONLINE]

ROUTERS:
| connector@east1[10583](ONLINE, created=0, active=0) |
| connector@east2[10548](ONLINE, created=0, active=0) |
| connector@east3[10540](ONLINE, created=0, active=0) |
| connector@west1[10589](ONLINE, created=0, active=0) |
| connector@west2[10541](ONLINE, created=0, active=0) |
| connector@west3[10547](ONLINE, created=0, active=0) |

DATASOURCES:
| east1(master:ONLINE, progress=1, THL latency=0.765) |
| STATUS [OK] [2019/12/24 10:21:12 AM UTC] |
| MANAGER(state=ONLINE) |
| REPLICATOR(role=master, state=ONLINE) |
| DATASERVER(state=ONLINE) |
| CONNECTIONS(created=0, active=0) |

| east2(slave:ONLINE, progress=1, latency=0.826) |
| STATUS [OK] [2019/12/24 10:21:13 AM UTC] |
| MANAGER(state=ONLINE) |
| REPLICATOR(role=slave, master=east1, state=ONLINE) |
| DATASERVER(state=ONLINE) |
| CONNECTIONS(created=0, active=0) |

| east3(slave:ONLINE, progress=1, latency=0.842) |
| STATUS [OK] [2019/12/24 10:21:13 AM UTC] |
| MANAGER(state=ONLINE) |
| REPLICATOR(role=slave, master=east1, state=ONLINE) |
| DATASERVER(state=ONLINE) |
| CONNECTIONS(created=0, active=0) |
```

82
Within each cluster, cctrl can be used to monitor the current status. For more information on checking the status and controlling operations, see Section 5.3, “Checking Dataservice Status”.

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
  host  | servicename   | role   | state  | appliedlastseqno | appliedlatency |
-------|---------------|--------|--------|------------------|---------------|
  east1 | east          | master | ONLINE | 5                 | 8.440         |
  east2 | east          | slave  | ONLINE | 5                 | 8.538         |
  east3 | east          | slave  | ONLINE | 5                 | 8.517         |
  east1 | east_from_west| relay  | ONLINE | 23                | 6.074         |
  east2 | east_from_west| slave  | ONLINE | 23                | 8.131         |
  east3 | east_from_west| slave  | ONLINE | 23                | 8.111         |
  west1 | west          | master | ONLINE | 23                | 8.021         |
  west2 | west          | slave  | ONLINE | 23                | 8.059         |
  west3 | west          | slave  | ONLINE | 23                | 8.089         |
  east1 | east_from_east| relay  | ONLINE | 5                 | 8.563         |
  east2 | east_from_east| slave  | ONLINE | 5                 | 8.592         |
  east3 | east_from_east| slave  | ONLINE | 5                 | 8.592         |
```

In the above example, it can be seen that the west services have a 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 servicename. For example, on east1 to check the status of the replicator within the Tungsten Clustering service, use the trepctl services command to get the status of both the local and cross-cluster services:

```
shell> trepctl status
Processing services command...
NAME      VALUE
appliedLastSeqno: 6
appliedLatency : 0.43
role        : master
serviceName : east
serviceType : local
started     : true
state       : ONLINE
NAME      VALUE
appliedLastSeqno: 4
appliedLatency : 1837.999
role        : relay
serviceName : east_from_west
serviceType : local
started     : true
state       : ONLINE
Finished services command...
```

To get a more detailed status, you must explicitly specify the service

```
shell> trepctl -service east_from_west status
```

### 3.4.4. Configuring Startup on Boot

For the Tungsten Clustering service, use Section 4.3, “Configuring Startup on Boot”.

### 3.4.5. 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 Tungsten Clustering will not attempt to automatically recover the service.

```
cctrl [east]> set policy maintenance
```
2. On the east, failed, Tungsten Clustering service, put each Tungsten Connector offline:
   
   ```shell
cctrl [east]> router * offline
   ```

3. Reset the local failed service on all servers connected to the remote failed service. For example, on west{1,2,3} reset the west_from_east service:
   
   ```shell
shell west> treectl -service west_from_east offline
shell west> treectl -service west_from_east reset -all -y
   ```

4. Reset the local service on each server within the failed region [east{1,2,3}]:
   
   ```shell
shell east> treectl -service east offline
shell east> treectl -service east reset -all -y
   ```

5. Restore a backup on each host [east{1,2,3}] in the failed east service from a host in the west service:
   
   ```shell
shell east> tungsten_provision_slave
   --direct --source=west1
   ```

6. Place all the services on west{1,2,3} back online:
   
   ```shell
shell west> treectl -service west_from_east online
   ```

7. On the east, failed, Tungsten Clustering service, put each Tungsten Connector online:
   
   ```shell
cctrl [east]> router * online
   ```

8. Set the policy back to automatic:
   
   ```shell
cctrl> set policy automatic
   ```

### 3.4.6. Resetting all dataservices

To reset all of the dataservices and restart the Tungsten Clustering services:

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

```shell
replicator stop
shell> tpm reset
shell> replicator start
```

### 3.4.7. 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 5.15, “Performing Database or OS Maintenance”, except that you must also restart the corresponding Tungsten Replicator service after the main Tungsten Clustering 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 Tungsten Clustering will not attempt to automatically recover the service.
   
   ```shell
cctrl [east]> set policy maintenance
   ```

2. Shun the first slave datasource so that maintenance can be performed on the host.
   
   ```shell
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:
   
   ```shell
cctrl [east]> service host/mysql restart
   ```

5. Bring the host back into the dataservice:
   
   ```shell
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:
   
   ```shell
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:
   
   ```shell
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

3.4.7.1. Fixing Replication Errors

In the event of a replication fault, the standard cctrl, trepctl and other utility commands in Chapter 8, 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:

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.

3.4.7.1.1. Recovering Cross Site Services

In a Multimaster topology, a switch or a failover not only promotes a slave to be a new master, but also will require the ability to reconfigure
cross-site communications. This process therefore assumes that cross-site communication is online and working. In some situations, it may
be possible that cross-site communication is down, or for some reason cross-site replication is in an OFFLINE:ERROR (in [Tungsten Replicator 6.1
Manual]) state - for example a DDL or DML statement that worked in the local cluster may have failed to apply in the remote.

If a switch or failover occurs and the process is unable to reconfigure the cross-site replicators, the local switch will still succeed, however the
associated cross-site services will be placed into a SHUNNED(SUBSERVICE-SWITCH-FAILED) state.

The guide explains how to recover from this situation.

- The examples are based on a 2-cluster topology, named NYC and LONDON and the composite dataservice named GLOBAL.
- The cluster is configured with the following dataservers:
  - NYC : db1 (master), db2 (slave), db3 (slave)
  - LONDON: db4 (master), db5 (slave), db6 (slave)
- The cross site replicators in both clusters are in an OFFLINE:ERROR (in [Tungsten Replicator 6.1 Manual]) state due to failing DDL
- A switch was then issued, promoting db3 as the new master in NYC and db5 as the new master in LONDON

When the cluster enters a state where the cross-site services are in an error, output from cctrl will look like the following:
In the above example, you can see that all services are in the **SHUNNED(SUBSERVICE-SWITCH-FAILED)** state, and partial reconfiguration has happened.

The Replicators for db4 and db6 should be slaves of db5. db5 has correctly configured to the new master in nyc, db3. The actual state of the cluster in each scenario maybe different depending upon the cause of the loss of cross-site communication. Using the steps below, apply the necessary actions that relate to your own cluster state, if in any doubt always contact Continuent Support for assistance.

1. The first step is to ensure the initial replication errors have been resolved and that the replicators are in an online state, the steps to resolve the replicators will depend on the reason for the error, for further guidance on resolving these issues, see Chapter 5, Operations Guide.

2. From one node, connect into cctrl at the expert level:

   ```
   shell> cctrl -expert -multi
   ```

3. Next, connect to the cross-site subservice, in this example, london_from_nyc

   ```
   cctrl> use london_from_nyc
   ```

4. Next, place the service into Maintenance Mode

   ```
   cctrl> set policy maintenance
   ```

5. Enable override of commands issued

   ```
   cctrl> set force true
   ```

6. Bring the relay datasource online

   ```
   cctrl> datasource db5 online
   ```

7. If you need to change the source for the relay replicator to the correct, new, master in the remote cluster, take the replicator offline. If the relay source is correct, then move on to step 10

   ```
   cctrl> replicator db5 offline
   ```

8. Change the source of the relay replicator

   ```
   cctrl> replicator db5 relay nyc/db3
   ```

9. Bring the replicator online

   ```
   cctrl> replicator db5 online
   ```

10. For each datasource that requires the replicator altering, issue the following commands:

    ```
    cctrl> replicator datasource offline
    cctrl> replicator datasource slave db5
    cctrl> replicator datasource online
    ```

    For example:

    ```
    cctrl> replicator db4 offline
    cctrl> replicator db4 slave db5
    cctrl> replicator db4 online
    ```

11. Once all replicators are using the correct source, we can then bring the cluster back

    ```
    cctrl> cluster welcome
    ```

12. Some of the datasources may still be in the SHUNNED state, so for each of those, you can then issue the following

    ```
    cctrl> datasource datasource online
    ```

    For example:

    ```
    cctrl> datasource db4 online
    ```

13. Once all nodes are online, we can then return the cluster to automatic
Deployment: MySQL Topologies

3.4.8. Adding a Cluster to a Composite Multimaster Topology

This procedure explains how to add additional clusters to an existing v6.x MultiMaster configuration.

The example in this procedure adds a new 3-node cluster consisting of nodes db7, db8 and db9 within a service called Tokyo. The existing cluster contains two dataservices, NYC and London, made up of nodes db1, db2, db3 and db4, db5, db6 respectively.

3.4.8.1. Pre-Requisites

Ensure the new nodes have all the necessary pre-requisites in place, specifically paying attention to the following:
- MySQL auto_increment parameters set appropriately on existing and new clusters
- All new nodes have full connectivity to the existing nodes and the hosts file contains correct hostnames
- All existing nodes have full connectivity to the new nodes and hosts file contains correct hostnames

3.4.8.2. Backup and Restore

We need to provision all the new nodes in the new cluster with a backup taken from one node in any of the existing clusters. In this example we are using db6 in the London dataservice as the source for the backup.

1. Shun and stop the services on the node used for the backup

   ```shell
   db6-shell> cctrl
cctrl> datasource db6 shun
cctrl> replicator db6 offline
cctrl> exit
db6-shell> stopall
   ```

2. Next, use whichever method you wish to copy the mysql datafiles from db6 to all the nodes in the new cluster (scp, rsync, xtrabackup etc). Ensure ALL database files are copied.

3. Once backup copied across, restart the services on db6

   ```shell
   db6-shell> sudo service mysqld start
db6-shell> startall
db6-shell> cctrl
cctrl> datasource db6 recover
cctrl> exit
   ```

4. Ensure all files copied to the target nodes have the correct file ownership

5. Start mysql on the new nodes

3.4.8.3. Update Existing Configuration

Next we need to change the configuration on the existing hosts to include the configuration of the new cluster.

You need to add a new service block that includes the new nodes and append the new service to the composite-datasource parameter in the composite dataservice, all within `/etc/tungsten/tungsten.ini`

Example of a new service block and composite-datasource change added to existing hosts configuration:

```ini
[tokyo]
topology=clustered
master=db7
members=db7,db8,db9
connectors=db7,db8,db9

[global]
topology=composite-multi-master
composite-datasources=nyc,london,tokyo
```

3.4.8.4. New Host Configuration

To avoid any differences in configuration, once the changes have been made to the tungsten.ini on the existing hosts, copy this file from one of the nodes to all the nodes in the new cluster.
Ensure `start-and-report` is `false` or not set in the config.

3.4.8.5. Install on new nodes

On the 3 new nodes, validate the software:

```
shell> cd /opt/continuent/software/tungsten-cluster-tungsten-clustering-6.1.3-43
shell> tools/tpm validate
```

This may produce Warnings that the tracking schemas for the existing cluster already exist - this is OK and they can be ignored. Assuming no other unexpected errors are reported, then go ahead and install the software:

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

3.4.8.6. Update existing nodes

Before we start the new cluster, we now need to update the existing clusters

1. Put entire cluster into MAINTENANCE

```
shell> cctrl
   cctrl> use {composite-dataservice}
   cctrl> set policy maintenance
   cctrl> ls
   COORDINATOR[db3:MAINTENANCE:ONLINE]
      london:COORDINATOR[db4:MAINTENANCE:ONLINE]
      nyc:COORDINATOR[db3:MAINTENANCE:ONLINE]
   cctrl> exit
```

2. Update the software on each node. This needs to be executed from the software staging directory using the replace-release option as this will ensure the new cross-site dataservices are setup correctly. Update the Master first followed by the slaves, cluster by cluster:

```
shell> cd /opt/continuent/software/tungsten-cluster-tungsten-clustering-6.1.3-43
shell> tools/tpm update --replace-release
```

3.4.8.7. Start the new cluster

On all the nodes in the new cluster, start the software:

```
shell> startall
```

3.4.8.8. Validate and check

Using `cctrl`, check that the new cluster appears and that all services are correctly showing online, it may take a few moments for the cluster to settle down and start everything

```
shell> cctrl
   cctrl> use {composite-dataservice}
   cctrl> ls
   cctrl> exit
```

Check the output of `trepctl` and ensure all replicators are online and new cross-site services appear in the pre-existing clusters

```
shell> trepctl -service {service} status
shell> trepctl services
```

Place entire cluster back into AUTOMATIC

```
shell> cctrl
   cctrl> use {composite-dataservice}
   cctrl> set policy automatic
   cctrl> ls
   COORDINATOR[db2:AUTOMATIC:ONLINE]
      london:COORDINATOR[db5:AUTOMATIC:ONLINE]
      nyc:COORDINATOR[db2:AUTOMATIC:ONLINE]
   cctrl> exit
```

3.5. 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 Tungsten Clustering on only the connector host. Failure to configure in this way, and `tpm` will install a full Tungsten Clustering service across all the implied members of the cluster.

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

   ```
   shell> cd /opt/continuent/software
   shell> tar zxf tungsten-clustering-6.1.3-43.tar.gz
   ```

2. Change to the Tungsten Clustering directory:

   ```
   shell> cd tungsten-clustering-6.1.3-43
   ```

3. Run `tpm` to perform the installation, using either the staging method or the INI method. Review Section 9.1, “Comparing Staging and INI `tpm` Methods” for more details on these two methods.

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

   Show Staging

   Show INI

   ```
   shell> ./tools/tpm configure defaults
   --reset
   --user=tungsten
   --profile-script=~/.bashrc
   --application-user=app_user
   --application-password=secret
   --application-port=3306
   --replication-port=13306
   --install-directory=/opt/continuent
   ```

   ```
   shell> ./tools/tpm configure alpha
   --connectors=connectorhost1
   --master=host1
   --members=host1,host2,host3
   ```

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

   **Configuration group defaults**

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

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

   - `--reset [470]`
     
     reset [470]
     
     For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

   - `--user=tungsten [477]`
     
     user=tungsten [477]
     
     System User

   - `--profile-script=~/.bashrc [467]`
     
     ```
     [defaults]
     user=tungsten
     profile-script=~/.bashrc
     application-user=app_user
     application-password=secret
     application-port=3306
     replication-port=13306
     install-directory=/opt/continuent
     ```

     ```
     [alpha]
     connectors=connectorhost1
     master=host1
     members=host1,host2,host3
     ```
Deployment: MySQL Topologies

profile-script=~/.bashrc [467]

Append commands to include env.sh in this profile script

• --application-user=app_user [437]
  application-user=app_user [437]

Database username for the connector

• --application-password=secret [437]
  application-password=secret [437]

Database password for the connector

• --application-port=3306 [437]
  application-port=3306 [437]

Port for the connector to listen on

• --replication-port=13306 [469]
  replication-port=13306 [469]

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

• --install-directory=/opt/continuent [456]
  install-directory=/opt/continuent [456]

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

Configuration group alpha

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

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

• --connectors=connectorhost1 [444]
  connectors=connectorhost1 [444]

Hostnames for the dataservice connectors

• --master=host1 [459]
  master=host1 [459]

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

• --members=host1,host2,host3 [460]
  members=host1,host2,host3 [460]

Hostnames for the dataservice members

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 Tungsten Clustering configuration. The values above are identical to those used in Section 3.1, “Deploying Standalone HA Clusters” deployment.

4. Run `tpm` to install the software with the configuration.

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

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

5. Initialize your `PATH` and environment.
6. Start the connector service:

   shell> connector 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:
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>connector@connector2[16019](ONLINE, created=0, active=0)</td>
</tr>
<tr>
<td>connector@host1[18450](ONLINE, created=19638, active=0)</td>
</tr>
<tr>
<td>connector@host2[1995](ONLINE, created=0, active=0)</td>
</tr>
<tr>
<td>connector@host3[8895](ONLINE, created=0, active=0)</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
</tr>
</tbody>
</table>

- The active status of the connector can be monitored using `cctrl` as normal.

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

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

  shell> mysql -utungsten -p -hconnector -P3306

- The `user.map` authorization file must be created and managed separately on standalone connectors. For more information, see Section 6.6, “User Authentication”

### 3.6. Deploying Additional Datasources, Managers, or Connectors

#### 3.6.1. Adding Datasources to an Existing Deployment

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

2. Update the configuration using `tpm`, adding the new host to the list of `--members [460]`, `--hosts [455]`, and `--connectors [444]`, if applicable.

   If using the staging method of deployment, you can use `+=`, which appends the host to the existing deployment as shown in the example below. Click the link to switch between staging and ini type deployment examples.

   **Show Staging**

   **Show INI**

   ```bash
   shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
   shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten
   shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1
   shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
   shell> ssh {STAGING_USER}@{STAGING_HOST}
   shell> cd {STAGING_DIRECTORY}
   shell> ./tools/tpm configure alpha \
   --members=host4 \
   --hosts=host4 \
   --connectors=host4 \
   Run the `tpm` command to update the software with the Staging-based configuration:

   shell> ./tools/tpm update --no-connectors
   ```
For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

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

```
[alpha]
...
```

```
members=host1,host2,host3,host4
```

```
hosts=host1,host2,host3,host4
```

```
connectors=host1,host2,host3,host4
```

Run the `tpm` command to update the software with the INI-based configuration:

```
shell> tpm query staging
```

```
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
```

```
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
shell> cd (STAGING_DIRECTORY)
```

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

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

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

4. 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 slave 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_2019-01-17_17-27_96
```

```
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)                          |
+----------------------------------------------------------------------------+
```

```
DATASOURCES:
```

```
+----------------------------------------------------------------------------+
|host1(master:ONLINE, progress=219, THL latency=1.047)                       |
|STATUS [OK] [2018/12/13 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] [2018/12/13 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] [2018/12/13 04:16:18 PM GMT]                                    |
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=host1, state=ONLINE)                        |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
```

```
+----------------------------------------------------------------------------+
```

```
[92]```
Deployment: MySQL Topologies

<table>
<thead>
<tr>
<th>MANAGER(state=ONLINE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPLICATOR(role=slave, master=host1, state=ONLINE)</td>
</tr>
<tr>
<td>DATASERVER(state=ONLINE)</td>
</tr>
<tr>
<td>CONNECTIONS(created=0, active=0)</td>
</tr>
</tbody>
</table>

If the host has not come up, or the progress does not match the master, check Section 5.6, “Datasource Recovery Steps” for more information on determining the exact status and what steps to take to enable the host operation.

### 3.6.2. 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 B, 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.

Update the configuration using `tpm`, adding the new host to the list of members:

- If using the staging method of deployment, you can use `+=`, which appends the host to the existing deployment as shown in the example below. Click the link to switch between staging and ini type deployment examples.

Show Staging

Shell> `tpm query staging`

Tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.1-41

Shell> `echo The staging USER is 'tpm query staging| cut -d: -f1 | cut -d@ -f1`

The staging USER is tungsten

Shell> `echo The staging HOST is 'tpm query staging| cut -d: -f1 | cut -d@ -f2`

The staging HOST is db1

Shell> `echo The staging DIRECTORY is 'tpm query staging| cut -d: -f2`

The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.1-41

Shell> `ssh {STAGING_USER}@{STAGING_HOST} cd {STAGING_DIRECTORY}`

Shell> `/tools/tpm configure alpha \\n  --members+=host4 \n  --witnesses=host4 \n  --enable-active-witnesses=true`

Shell> `/tools/tpm update --no-connectors`

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

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

[alpha]
...
  Members=host1,host2,host3,host4
  Witnesses=host4
  enable-active-witnesses=true

Run the `tpm` command to update the software with the INI-based configuration:

Shell> `tpm query staging`

Tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.1-41

Shell> `echo The staging DIRECTORY is 'tpm query staging| cut -d: -f2`
For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

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

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] [2018/12/05 04:27:47 PM GMT] |
| MANAGER(state=ONLINE) |
| REPLICATOR(role=slave, master=host3, state=ONLINE) |
| DATASERVER(state=ONLINE) |
| CONNECTIONS(created=0, active=0) |
```

```
|host2(slave:ONLINE, progress=8946, latency=0.334) |
|STATUS [OK] [2018/12/05 04:06:59 PM GMT] |
| MANAGER(state=ONLINE) |
| REPLICATOR(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] [2018/11/20 05:39:14 PM GMT] |
| MANAGER(state=ONLINE) |
| REPLICATOR(role=master, state=ONLINE) |
| DATASERVER(state=ONLINE) |
| CONNECTIONS(created=0, active=0) |
```

```
WITNESSES:
|host4(witness:ONLINE) |
| MANAGER(state=ONLINE) |
```

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

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

To add passive witness to an existing installation, use `tpm` to update the active configuration.

```
Passive Witness hosts are deprecated as of v6.1.0
Use Active Witness hosts instead
Passive Witness support will be removed in v6.2.0
```

Note
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”
### 3.6.4. 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 B, Prerequisites have been followed.

2. Update the configuration using `tpm`, adding the new host to the list of connectors [444]

   If using the staging method of deployment, you can use `+=`, which appends the host to the existing deployment as shown in the example below. Click the link to switch between staging and ini type deployment examples.
Deployment: MySQL Topologies

The staging USER is tungsten

```
shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1
```

```
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
shell> ssh {STAGING_USER}@{STAGING_HOST}
```

```
shell> cd {STAGING_DIRECTORY}
```

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

Run the `tpm` command to update the software with the Staging-based configuration:

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

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, "Configuration Changes from a Staging Directory".

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

```
[alpha]
...
connectors=host4
```

Run the `tpm` command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
shell> cd {STAGING_DIRECTORY}
```

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

For information about making updates when using an INI file, please see Section 9.4.4, "Configuration Changes with an INI file".

3. 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)      |
+---------------------------------------------------------+
```

3.6.5. Adding a remote Composite Cluster

Adding an entire new cluster provides significant level of availability and capacity. The new cluster nodes that form the cluster will be fully aware of the original cluster(s) and communicate with existing managers and datasources within the cluster.

The following steps guide you through updating the configuration to include the new hosts and services you are adding.

1. On the new host(s), ensure the Appendix B, Prerequisites have been followed.

2. Let’s assume that we have a composite cluster dataservice called global with two clusters, east and west, with three nodes each.

In this worked example, we show how to add an additional cluster service called north with three new nodes.

3. Set the cluster to maintenance mode using `cctrl`

```
shell> cctrl
```
4. Using the following as an example, update the configuration to include the new cluster and update the additional composite service block.

Click the link to switch between staging or ini examples

Show Staging

Show INI

```sh
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41

shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten

shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41

shell> ssh `{STAGING_USER}@{STAGING_HOST}`

shell> cd `{STAGING_DIRECTORY}`
```

Run the `tpm` command to update the software with the Staging-based configuration:

```sh
shell> ./tools/tpm configure north 
    --connectors=db7,db8,db9 
    --relay-source=east 
    --relay=db7 
    --slaves=db8,db9 
    --topology=clustered
```

Run the `tpm` command to update the software with the INI-based configuration:

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

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

6. On every node in the original clusters, make sure all replicators are online:

```sh
shell> trepctl online; trepctl services
```

7. Update the old cluster via cctrl to recognize the new cluster (i.e. north):

```sh
shell> cctrl -multi
```
8. Go to the relay (master) node of the new cluster (i.e. db7) and provision it from a slave of the original cluster (i.e. db2):

```
tungsten@db7:~  $ tungsten_provision_slave --source db2
```

```
tungsten@db7:~  $ tungsten_provision_slave --source db2
```

```
NOTE  >> Put north replication service offline
NOTE  >> Create a backup of db2 in /opt/continuent/backups/provision_xtrabackup_2019-01-07_13-53_58
NOTE  >> db2 >> Run innobackupex sending the output to db7:/opt/continuent/backups/provision_xtrabackup_2019-01-07_13-53_58
NOTE  >> db2 >> Prepare the files for MySQL to run
NOTE  >> Stop MySQL and empty all data directories
NOTE  >> Stop the MySQL service
NOTE  >> Transfer data files to the MySQL data directory
NOTE  >> Start the MySQL service
NOTE  >> Backup and restore complete
NOTE  >> Put the north replication service online
NOTE  >> Clear THL and relay logs for the north replication service

9. Go to a slave node of the new cluster (i.e. db8) and provision it from the relay node of the new cluster (i.e. db7):

```
tungsten@db8:~  $ tungsten_provision_slave --source db7
```

```
tungsten@db8:~  $ tungsten_provision_slave --source db7
```

```
NOTE  >> Put north replication service offline
NOTE  >> Create a backup of db7 in /opt/continuent/backups/provision_xtrabackup_2019-01-07_13-54_71
NOTE  >> db7 >> Run innobackupex sending the output to db8:/opt/continuent/backups/provision_xtrabackup_2019-01-07_13-54_71
NOTE  >> db7 >> Prepare the files for MySQL to run
NOTE  >> Stop MySQL and empty all data directories
NOTE  >> Stop the MySQL service
NOTE  >> Transfer data files to the MySQL data directory
NOTE  >> Start the MySQL service
NOTE  >> Backup and restore complete
10. Go to a slave node (i.e. db9) of the new cluster and provision it from the newly-provisioned slave node of the new cluster (i.e. db8):

```shell
tungsten_provision_slave --source db8
```

NOTE >> Put the north replication service online

NOTE >> Clear THL and relay logs for the north replication service

11. Set the composite cluster to automatic mode using `cctrl`:

```shell
cctrl -multi
```

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

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

### 3.6.6. Converting from a single cluster to a composite cluster

There are two possible scenarios for converting from a single standalone cluster to a composite cluster. The two following sections will guide you through examples of each of these.

#### 3.6.6.1. Convert and add new nodes as a new service

The following steps guide you through updating the configuration to include the new hosts as a new service and convert to a Composite Cluster.

For the purpose of this worked example, we have a single cluster dataservice called `east` with three nodes, defined as db1, db2 and db3 with db1 as the master.

Our goal is to create a new cluster dataservice called `west` with three nodes, defined as db4, db5 and db6 with db4 as the relay.

We will configure a new composite dataservice called `global`.

1. On the new host(s), ensure the Appendix B, Prerequisites have been followed.

   If configuring via the Staging Installation method, skip straight to Step 4:

2. On the new host(s), ensure the `tungsten.ini` contains the correct service blocks for both the existing cluster and the new cluster.

3. On the new host(s), install the proper version of clustering software, ensuring that the version being installed matches the version currently installed on the existing hosts.

   **Important**
   
   Ensure `--start-and-report [472]` is set to `false` in the configuration for the new hosts.

4. Set the existing cluster to maintenance mode using `cctrl`:

```shell
cctrl -multi
```

5. Add the definition for the new slave cluster service `west` and composite service `global` to the existing configuration on the existing host(s):

   Click the link to switch between staging or ini examples

   Show Staging

   Show INI

   ```shell
tpm query staging
   ```
Deployment: MySQL Topologies

```
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
```
```
shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten

```
shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1

```
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41

```
shell> ssh {STAGING_USER}@{STAGING_HOST}
```
```
shell> cd {STAGING_DIRECTORY}
```
```
shell> ./tools/tpm configure west 
    --connectors=db4,db5,db6 
    --relay-source=east 
    --relay=db4 
    --slaves=db5,db6 
    --topology=clustered
```
```
shell> ./tools/tpm configure global 
    --composite-datasources=east,west
```

Run the `tpm` command to update the software with the Staging-based configuration:
```
shell> ./tools/tpm update --no-connectors --replace-release
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.
```
shell> vi /etc/tungsten/tungsten.ini
```
```
[west]
...
connectors=db4,db5,db6
relay-source=east
relay=db4
slaves=db5,db6
topology=clustered
```
```
[global]
...
composite-datasources=east,west
```

Run the `tpm` command to update the software with the INI-based configuration:
```
shell> ./tools/tpm update --no-connectors --replace-release
```

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.
```
Note

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

Using the `--replace-release` option ensures the metadata files for the cluster are correctly rebuilt. This parameter MUST be supplied.
```

6. On every node in the original cluster, make sure all replicators are online:
```
shell> trepctl online; trepctl services
```

7. On all the new hosts in the new cluster, start the manager processes only
```
shell> manager start
```

8. From the original cluster, use cctrl to check that the new dataservice and composite dataservice have been created, and place the new dataservice into maintenance mode
```
```
9. Start the replicators in the new cluster ensuring they start as OFFLINE:

```
shell> replicator start offline
```

10. Go to the relay (master) node of the new cluster (i.e. db4) and provision it from a slave of the original cluster (i.e. db2):

```
db4-shell> prov-sl.sh -s db2 -m xtrabackup
source = db2
method = xtrabackup
parallel threads = 4
port = 22
help =
Topology = COMPOSITE
```

11. Go to each slave node of the new cluster and provision from the relay node of the new cluster (i.e. db4):

```
db5-shell> prov-sl.sh -s db4
source = db4
method = mysqldump
parallel threads = 4
port = 22
help =
Topology = COMPOSITE
```

```
db6-shell> prov-sl.sh -s db4
source = db4
method = mysqldump
parallel threads = 4
port = 22
help =
Topology = COMPOSITE
```

12. Bring the replicators in the new cluster online:

```
shell> trepctl online
```
13. From a node in the original cluster (e.g. db1), using cctrl, set the composite cluster online and return both clusters to automatic:

```shell
shell> cctrl -multi
[LOGICAL] / > use global
[LOGICAL] / > datasource west online
[LOGICAL] / > set policy automatic
```

14. Start the connectors associated with the new cluster hosts in west:

```shell
shell> connector start
```

15. If --no-connectors was issued during the update, then during a period when it is safe, restart the connectors associated with the original cluster:

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

### 3.6.6.2 Convert and move nodes to a new service

This method of conversion is a little more complicated and the only safe way to accomplish this would require downtime for the replication on all nodes.

To achieve this without downtime to your applications, it is recommended that all application activity be isolated to the master host only. Following the conversion, all activity will then be replicated to the slave nodes.

Our example starting cluster has 5 nodes (1 master and 4 slaves) and uses service name `alpha`. Our target cluster will have 6 nodes (3 per cluster) in 2 member clusters `alpha_east` and `alpha_west` in composite service `alpha`.

This means that we will reuse the existing service name `alpha` as the name of the new composite service, and create two new service names, one for each cluster (`alpha_east` and `alpha_west`).

To convert the above configuration, follow the steps below:

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

2. Ensure the cluster is in MAINTENANCE mode. This will prevent the managers from performing any unexpected recovery or failovers during the process.

```shell
shell> cctrl> set policy maintenance
```

3. Next, you must stop all services on all existing nodes.

```shell
shell> stopall
```

4. If configuring via the INI Installation Method, update tungsten.ini on all original 5 nodes, then copy the file to the new node.

You will need to create two new services for each cluster, and change the original service stanza to represent the composite service. An example of how the complete configuration would look is below. Click the link to switch between ini and staging configurations.

**Show Staging**

**Show INI**

```ini
shell> ./tools/tpm configure defaults \
   --reset \
   --user=tungsten \
   --install-directory=/opt/continuent \
   --profile-script=~/.bash_profile \
   --replication-user=tungsten \
   --replication-password=secret \
   --replication-port=13306 \
   --application-user=app_user \
   --application-password=secret \
   --application-port=3306
```

```ini
shell> ./tools/tpm configure alpha_east \
   --topology=clustered \
   --master=db1 \
   --members=db1,db2,db3 \
   --connectors=db1,db2,db3
```

```ini
shell> ./tools/tpm configure alpha_west \
   --topology=clustered \
   --relay=db4 \
   --members=db4,db5,db6 \
   --connectors=db4,db5,db6
```
Deployment: MySQL Topologies

```
shell> ./tools/tpm configure alpha
    --composite-datasources=alpha_east,alpha_west

shell> vi /etc/tungsten/tungsten.ini

[defaults]
user=tungsten
install-directory=/opt/continuent
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=secret
replication-port=13306
application-user=app_user
application-password=secret
application-port=3306

[alpha_east]
topology=clustered
master=db1
members=db1,db2,db3
connectors=db1,db2,db3

[alpha_west]
topology=clustered
relay=db4
members=db4,db5,db6
connectors=db4,db5,db6
relay-source=alpha_east

[alpha]
composite-datasources=alpha_east,alpha_west
```

5. Using your preferred backup/restore method, take a backup of the MySQL database on one of the original nodes and restore this to the new node.

If preferred, this step can be skipped, and the provision of the new node completed via the use of the supplied provisioning scripts, explained in Step 10 below.

6. Invoke the conversion using the `tpm` command from the software extraction directory.

   If installation configured via the INI method, this command should be run on all 5 original nodes. If configured via Staging method, this command should be run on the staging host only.

```
shell> tpm query staging
shell> cd {software_staging_dir_from_tpm_query}
shell> ./tools/tpm update --replace-release --force
shell> rm /opt/continuent/tungsten/cluster-home/conf/cluster/*/datasource/*
```

**Note**

The use of the `--force` option is required to force the override of the old properties.

7. Only if installation configured via the INI method, then proceed to install the software using the `tpm` command from the software extraction directory on the new node:

```
shell> cd {software_staging_dir}
shell> ./tools/tpm install
```

**Note**

Ensure you install the same version of software on the new node that matches exactly, the version on the existing 5 nodes.

8. Start all services on all existing nodes.

```
shell> startall
```

9. Bring the clusters back into AUTOMATIC mode:

```
shell> cctrl -multi
cctrl> use alpha
cctrl> set policy automatic
cctrl> exit
```

10. If you skipped the backup/restore step above, you now need to provision the database on the new node. To do this, use the prov-sl.sh script to provision the database from one of the existing nodes, for example db5.
3.7. 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.

Additionally, using a replicator that writes data into an existing data service can be used when migrating from an existing service into a new Tungsten Clustering service. For more information on initially provisioning the data for this type of operation, see Section 5.12.2, “Migrating from MySQL Native Replication Using a New Service”.

![Figure 3.5. Topologies: Replicating into a Dataservice](image)

In order to configure this deployment, there are two steps:

1. Create a new replicator on the source server that extracts the data.
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 5.3.0 or later.
- Hosts on both the replicator and cluster must be able to communicate with each other.
- The replication user on the source host must have the `RELOAD`, `REPLICATION SLAVE`, and `REPLICATION CLIENT GRANT` privileges.
- Replicator must be able to connect as the `tungsten` user to the databases within the cluster.
- 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`:

```text
# Add the tungsten user to the user.map
```

Install the Tungsten Replicator package (see Section 2.3.2, “Using the RPM and DEB package files”), or download the compressed tarball and unpack it on `host1`:

```bash
shell> cd /opt/replicator/software
shell> tar xzvf tungsten-replicator-6.1.3-43.tar.gz
```

Change to the Tungsten Replicator staging directory:

```bash
shell> cd tungsten-replicator-6.1.3-43
```

Configure the replicator on `host1`
First we configure the defaults and a cluster alias that points to the masters and slaves within the current Tungsten Clustering service that you are replicating from:

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

Show Staging

Show INI

```
shell> ./tools/tpm configure alpha \
     --master=host1 \
     --install-directory=/opt/continuent \
     --replication-user=tungsten \
     --replication-password=password \
     --enable-batch-service=true
```

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

Configuration group alpha

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

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

- **--master=host1** [459]
  
  master=host1 [459]

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

- **--install-directory=/opt/continuent** [456]
  
  install-directory=/opt/continuent [456]

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

- **--replication-user=tungsten** [470]
  
  replication-user=tungsten [470]

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

- **--replication-password=password** [469]
  
  replication-password=password [469]

  The password to be used when connecting to the database using the corresponding --replication-user [470].

- **--enable-batch-service=true** [in [Tungsten Replicator 6.1 Manual]]
  
  enable-batch-service=true [in [Tungsten Replicator 6.1 Manual]]

  This option enables batch mode for a service, which ensures that replication services that are writing to a target database using batch mode in heterogeneous deployments [for example Hadoop, Amazon Redshift or Vertica]. Setting this option enables the following settings on each host:

  - On a Master
    - mysql-use-bytes-for-string [464] is set to false.
    - colnames filter is enabled [in the binlog-to-q stage to add column names to the THL information.
    - pkey filter is enabled [in the binlog-to-q and q-to-dbms stage], with the addPkeyToInserts and addColumnsToDeletes filter options set to true. This ensures that rows have the right primary key information.
• `enumtoString` filter is enabled [in the `q-to-thl` stage], to translate `ENUM` values to their string equivalents.

• `settoString` filter is enabled [in the `q-to-thl` stage], to translate `SET` values to their string equivalents.

On a Slave

• `mysql-use-bytes-for-string` is set to true.

• `pkey` filter is enabled [in the `q-to-dbms` stage].

This creates a configuration that specifies that the topology should read directly from the source host, `host3`, writing directly to `host1`. An alternative THL port is provided to ensure that the THL listener is not operating on the same network port as the original.

Now install the service, which will create the replicator reading direct from `host3` into `host1`:

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

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

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

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

Use `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 -service beta status
Processing status command...
NAME                      VALUE
appliedLastEventId        : mysql-bin.000026:00000000000000001311:1252
appliedLastSeqno          : 5
appliedLatency            : 0.748
channel                  : 1
clusterName              : beta
currentTimeMillis         : 1390410611881
dataServerHost           : host1
databaseHost             : host3
extensions               : 
host                     : host3
latestEpochNumber         : 1
masterConnectURI          : thl://host3:2112/
masterListenURI          : thl://host1:2113/
maximumStoredSeqno       : 5
minimumStoredSeqno       : 0
offlineRequests           : NONE
pendingError             : NONE
pendingErrorId           : NONE
pendingErrorSeqno        : -1
pendingExceptionMessage  : NONE
pipelineSource           : jdbc:mysql:thin://host3:13306/
relativeLatency          : 8408.881
resourcePrecedence       : 99
rmiPort                  : 10000
role                     : master
segType                  : java.lang.Long
serviceName              : beta
serviceType              : local
simpleServiceName        : beta
```

106
3.8. Replicating Data Out of a Cluster

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

**Figure 3.6. Topologies: Replicating Data Out of a Cluster**

![Diagram](image)

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

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

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

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

1. Identify the cluster to replicate from. You will need the master, slaves and THL port (if specified). Use `tpm reverse` from a cluster member to find the correct values.

2. If you are replicating to a non-MySQL server. Update the configuration of the cluster to include `--enable-heterogeneous-service=true` prior to beginning. The same option must be included when installing the Tungsten Replicator.

3. Identify all servers that will replicate from the cluster. If there is more than one, a relay server should be identified to replicate from the cluster and provide THL data to other servers.

4. Prepare each server according to the prerequisites for the DBMS platform it is serving. If you are working with multiple DBMS platforms; treat each platform as a different cluster-slave during deployment.
5. Make sure the THL port for the cluster is open between all servers.

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

1. Install the Tungsten Replicator package or download the Tungsten Replicator tarball, and unpack it:
   ```shell
   cd /opt/continuent/software
tar zxf tungsten-replicator-6.1.3-43.tar.gz
   ```

2. Change to the unpackaged directory:
   ```shell
cd tungsten-clustering-6.1.3-43
   ```

3. Configure the replicator
   
   Click the link below to switch examples between Staging and INI methods
   
   Show Staging
   
   Show INI
   
   ```shell
   ./tools/tpm configure defaults
   --install-directory=/opt/continuent
   --profile-script=~/.bash_profile
   --replication-password=secret
   --replication-port=13306
   --replication-user=tungsten
   --user=tungsten
   ```

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

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

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

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

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

   - `--install-directory=/opt/continuent` [456]
     
     **install-directory=/opt/continuent** [456]
     
     Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

   - `--profile-script=~/.bash_profile` [467]
     
     **profile-script=~/.bash_profile** [467]
Deployment: MySQL Topologies

Append commands to include env.sh in this profile script

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

- `--replication-port=13306`
  
  The network port used to connect to the database server. The default port used depends on the database being configured.

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

- `--user=tungsten`
  
  System User

Configuration group alpha

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

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

- `--master=host1`
  
  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- `--slaves=host2,host3`
  
  What are the slaves for this dataservice?

- `--thl-port=2112`
  
  Port to use for THL Operations

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

Configuration group beta

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

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

- `--relay=host6`
  
  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- `--relay-source=alpha`
  
  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.
Deployment: MySQL Topologies

```
relay-source=alpha
```

Dataservice name to use as a relay source

- topology=cluster-slave

```
topology=cluster-slave
```

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

**Important**
If you are replicating to a non-MySQL server. Include the `--enable-heterogeneous-service=true` option in the above command.

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

**Note**
Do not include `start-and-report=true` if you are taking over for MySQL native replication. See Section 5.12.1, “Migrating from MySQL Native Replication 'In-Place’” for next steps after completing installation.

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

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

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

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

The cluster should be installed and ready to use.

### 3.9. Replicating from a Cluster to a Datawarehouse

You can replicate data from an existing cluster to a datawarehouse such as Hadoop or Vertica. A replication slave node handles the datawarehouse loading by obtaining THL from the cluster. The configuration of the cluster needs to be changed to be compatible with the required target applier format.

The cluster-slave deployment works by configuring the cluster replication service in heterogeneous mode, and then replicating out to the slave that writes into the datawarehouse by using a cluster alias. This ensures that changes to the cluster topology [i.e. master switches during a failover or maintenance] still allow replication to continue effectively to your chosen datawarehouse.

The datawarehouse may be installed and running on the same host as the replicator, “Onboard”, or on a different host entirely, “Offboard”.

---

110
Below is a summary of the steps needed to configure the cluster-slave topology, with links to the actual procedures included:

1. Install or update a cluster, configured to operate in heterogeneous mode.

   In our example, the cluster configuration file `/etc/tungsten/tungsten.ini` would contain two stanzas:
   - `[defaults]` - contains configuration values used by all services.
   - `[alpha]` - contains cluster configuration parameters, and will use `topology=clustered` to indicate to the `tpm` command that nodes listed in this stanza are to be acted upon during installation and update operations.

   For more details about installing the source cluster, please see Section 3.9.2, “Replicating from a Cluster to a Datawarehouse - Configuring the Cluster Nodes”.

2. Potentially seed the initial data. For more information about various ways to provision the initial data into the target warehouse, please see Section 3.10, “Migrating and Seeding Data”.

3. Install the slave replicator:

   In our example, the slave configuration file `/etc/tungsten/tungsten.ini` would contain three stanzas:
• **[defaults]** - contains configuration values used by all services.

• **[alpha]** - contains the list of cluster nodes for use by the slave applier service as a source list. This stanza will use `topology=clus-ter-alias [476]` to ensure that no installation or update action will ever be taken on the listed nodes by the `tpm` command.

• **[omega]** - defines a replicator slave applier service that uses `topology=cluster-slave [476]`. This service will extract THL from the cluster nodes defined in the relay source cluster-alias definition `[alpha]` and write the events into your chosen datawarehouse.

For more details about installing the slave replicator, please see Section 3.9.3, “Replicating from a Cluster to a Datawarehouse - Configuring the Cluster-Slave”.

### 3.9.1. Replicating from a Cluster to a Datawarehouse - Prerequisites

There are the prerequisite requirements for cluster-slave operations:

• The Tungsten Clustering and Tungsten Replicator must be version 5.2.0 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.

### 3.9.2. Replicating from a Cluster to a Datawarehouse - Configuring the Cluster Nodes

There are the steps to configure a cluster to act as the source for a cluster-slave replicator writing into a datawarehouse:

• Enable MySQL ROW-based Binary Logging

  All MySQL databases running in clusters replicating to non-MySQL targets must operate in ROW-based replication mode to prevent data drift.

  This is required because replication to the datawarehouse environment must send the raw-data, rather than the statements which cannot be applied directly to a target datawarehouse.

  You must configure the *my.cnf* file to enable ROW-based binary logging:

  ```
  binlog-format = ROW
  ```

  ROW-based binary logging can also be enabled without restarting the MySQL server:

  ```
  mysql> select @@global.binlog_format;
  +-------------------------+
  | @@global.binlog_format  |
  +-------------------------+
  | MIXED                   |
  +-------------------------+
  mysql> SET GLOBAL binlog_format = 'ROW';
  Query OK, 0 rows affected (0.00 sec)
  mysql> select @@global.binlog_format;
  +-------------------------+
  | @@global.binlog_format  |
  +-------------------------+
  | ROW                     |
  +-------------------------+
  ```

• Enable and Configure the Extractor Filters

  Heterogeneous mode should be enabled within the cluster.

  The extractor filters and two associated properties add the column names and primary key details to the THL. This is required so that the information can be replicated into the datawarehouse correctly.

  For example, on every cluster node the lines below would be added to the `/etc/tungsten/tungsten.ini` file, then `tpm update` would be executed:

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

  For staging deployments, prepend two hyphens to each line and include on the command line.
3.9.3. Replicating from a Cluster to a Datawarehouse - Configuring the Cluster-Slave

Configure the replicator that will act as a slave, reading information from the cluster and then applying that data into the chosen datawarehouse. Multiple example targets are shown.

This node may be located either on a separate host (for example when replicating to Amazon Redshift), or on the same node as the target datawarehouse service (i.e. HP Vertica or Hadoop).

On the following pages are the steps to configure a cluster-slave target replicator writing into a datawarehouse for both staging and INI methods of installation.

Figure 3.8. Topologies: Replication from a Cluster to an Offboard Datawarehouse

3.9.3.1. Replicating Data from a Cluster to a Datawarehouse (Staging Use Case)

The following Staging-method procedure will install the Tungsten Replicator software onto target node host6, extracting from a cluster consisting of three (3) nodes (host1, host2 and host3) and applying into the target datawarehouse via host6.

Important

If you are replicating to a MySQL-specific target, please see Section 3.8, “Replicating Data Out of a Cluster” for more information.
1. On your staging server, go to the software directory.

   ```shell
   cd /opt/continuent/software
   ```

2. Download the latest Tungsten Replicator version.

3. Unpack the release package

   ```shell
   tar xvzf tungsten-replicator-6.0.3-599.tar.gz
   ```

4. Change to the unpackaged directory:

   ```shell
   cd tungsten-replicator-6.0.3-599
   ```

5. Execute the `tpm` command to configure defaults for the installation.

   ```shell
   ./tools/tpm configure defaults
   --install-directory=/opt/replicator
   --profile-script=~/.bashrc
   --replication-password=secret
   --replication-port=13306
   --replication-user=tungsten
   --start-and-report=true
   --mysql-allow-intensive-checks=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 defaults`
     This runs the `tpm` command. `configure defaults` indicates that we are setting options which will apply to all dataservices.

   - `--install-directory=/opt/replicator [456]`
     The installation directory of the Tungsten service. This is where the service will be installed on each server in your dataservice.

   - `--profile-script=~/.bashrc [467]`
     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 Tungsten Clustering™ are easier to use.

   - `--user=tungsten [477]`
     The operating system user name that you have created for the Tungsten service, `tungsten`.

   - `--replication-user=tungsten [470]`
     The user name that will be used to apply replication changes to the database on slaves.

   - `--replication-password=password [469]`
     The password that will be used to apply replication changes to the database on slaves.

   - `--replication-port=13306 [469]`
     Set the port number to use when connecting to the MySQL server.

   - `--start-and-report [472]`
     Tells `tpm` to startup the service, and report the current configuration and status.

6. Configure a cluster alias that points to the masters and slaves within the current Tungsten Clustering service that you are replicating from:

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

   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 alpha`
This runs the **tpm** command. **configure** indicates that we are creating a new dataservice, and **alpha** is the name of the dataservice being created.

This definition is for a dataservice alias, not an actual dataservice because **--topology=cluster-alias** has been specified. This alias is used in the cluster-slave section to define the source hosts for replication.

- **--master=host1**
  Specifies the hostname of the default master in the cluster.

- **--slaves=host2,host3**
  Specifies the name of any other servers in the cluster that may be replicated from.

- **--thl-port=2112**
  The THL port for the cluster. The default value is 2112 but any other value must be specified.

- **--topology=cluster-alias**
  Define this as a cluster dataservice alias so **tpm** does not try to install cluster software to the hosts.

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

7. On the replication slave node, copy the `convertstringfrommysql.json` filter configuration sample file into the `/opt/replicator/share` directory then edit it to suit:

```
cp /opt/replicator/tungsten/tungsten-replicator/support/filters-config/convertstringfrommysql.json /opt/replicator/share/
vi /opt/replicator/share/convertstringfrommysql.json
```

Once the `convertstringfrommysql` JSON configuration file has been edited, update the `/etc/tungsten/tungsten.ini` file to add and configure any addition options needed for the specific datawarehouse you are using.

8. Create the configuration that will replicate from cluster dataservice **alpha** into the database on the host specified by **--relay=host6**:

```
shell>./tools/tpm configure omega \
  --relay=host6 \
  --relay-source=alpha \
  --repl-svc-remote-filters=convertstringfrommysql \
  --property=replicator.filter.convertstringfrommysql.definitionsFile=/opt/replicator/share/convertstringfrommysql.json \
  --topology=cluster-slave
```

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 omega**
  This runs the **tpm** command. **configure** indicates that we are creating a new replication service, and **omega** is the unique service name for the replication stream from the cluster.

- **--relay=host6**
  Specifies the hostname of the destination database into which data will be replicated.

- **--relay-source=alpha**
  Specifies the name of the source cluster dataservice alias (defined above) that will be used to read events to be replicated.

- **--topology=cluster-slave**
  Read source replication data from any host in the **alpha** dataservice.

9. Now finish configuring the **omega** dataservice with the options specific to the datawarehouse target in use.

- **AWS RedShift Target**

```
shell>./tools/tpm configure omega \
  --batch-enabled=true \
  --batch-load-template=redshift \
  --enable-heterogeneous-slave=true
```

---

115
Deployment: MySQL Topologies

```bash
--datasource-type=redshift \
--replication-host=REDSHIFT_ENDPOINT_FQDN_HERE \
--replication-user=REDSHIFT_PASSWORD_HERE \
--replication-password=REDSHIFT_PASSWORD_HERE \
--redshift-dbname=REDSHIFT_DB_NAME_HERE \
--svc-applier-filters=dropstatementdata \
--svc-applier-block-commit-interval=10s \
--svc-applier-block-commit-size=5
```

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.

  - **--topology=cluster-slave** [476]

    Configure the topology as a cluster-slave. This will configure the individual replicator as a slave of all the nodes in the cluster, as defined in the previous configuration of the cluster topology.

  - **--relay** [459]

    Configure the node as the relay for the cluster which will replicate data into the datawarehouse.

  - **--enable-heterogeneous-slave** [453]

    Configures the slave to correctly process the incoming data so that it can be written to the datawarehouse. This includes correcting the processing of text data types and configuring the appropriate filters.

  - **--replication-host** [469]

    The target host for writing data. In the case of Redshift, this is the fully qualified hostname of the Redshift host.

  - **--replication-user** [470]

    The user within the Redshift service that will be used to write data into the database.

  - **--replication-password=password** [469]

    The password for the user within the Redshift service that will be used to write data into the database.

  - **--datasource-type=redshift** [449]

    Set the datasource type to be used when storing information about the replication state.

  - **--batch-enabled=true** [439]

    Enable the batch service, this configures the JavaScript batch engine and CSV writing semantics to generate the data to be applied into a datawarehouse.

  - **--batch-load-template=redshift** [440]

    The batch load template to be used. Since we are replicating into Redshift, the *redshift* template is used.

  - **--redshift-dbname=dev** [468]

    The name of the database within the Redshift service where the data will be written.

Please see Install Amazon Redshift Applier [in [Tungsten Replicator 6.1 Manual]] for more information.

- **Vertica Target**

```bash
shell> ./tools/tpm configure omega \
--batch-enabled=true \
--batch-load-template=vertica6 \
--batch-load-language=js \
--datasource-type=vertica \
--disable-relay-logs=true \
--enable-heterogeneous-service=true \
--replication-user=dbadmin \
--replication-password=VERTICA_DB_PASSWORD_HERE \
--replication-host=VERTICA_HOST_NAME_HERE \
--replication-port=5433 \
```
Deployment: MySQL Topologies

```
--svc-applier-block-commit-interval=5s \
--svc-applier-block-commit-size=500 \
--vertica-dbname=VERTICA_DB_NAME_HERE
```

Please see Install Vertica Applier [in [Tungsten Replicator 6.1 Manual]] for more information.

- Elasticsearch Target - Native Applier

```
shell> ./tools/tpm configure omega \
--datasource-type=elasticsearch \
--replication-host=ELASTICSEARCH_HOST_NAME_HERE \
--replication-password=null \
--replication-port=9200 \
--replication-user=root 
```

Please see Install Elasticsearch Applier [in [Tungsten Replicator 6.1 Manual]] for more information.

- For additional targets, please see the full list at Deploying Appliers [in [Tungsten Replicator 6.1 Manual]], or click on some of the targets below:
  - Deploying the Oracle Applier [in [Tungsten Replicator 6.1 Manual]]
  - Deploying the MySQL Applier [in [Tungsten Replicator 6.1 Manual]]
  - Deploying the Vertica Applier [in [Tungsten Replicator 6.1 Manual]]

10. Once the configuration has been completed, you can perform the installation to set up the Tungsten Replicator services using the `tpm` command run from the staging directory:

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

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

The cluster-slave replicator should now be installed and ready to use.

### 3.9.3.2. Replicating Data from a Cluster to a Datawarehouse (INI Use Case)

The following INI-based procedure will install the Tungsten Replicator software onto target node host6, extracting from a cluster consisting of three (3) nodes [host1, host2 and host3] and applying into the target datawarehouse via host6.

**Important**

If you are replicating to a MySQL-specific target, please see Deploying the MySQL Applier [in [Tungsten Replicator 6.1 Manual]] for more information.

1. On the replication slave node, copy the `convertstringfrommysql.json` filter configuration sample file into the `/opt/replicator/share` directory then edit it to suit:

```
cp /opt/replicator/tungsten/tungsten-replicator/support/filters-config/convertstringfrommysql.json /opt/replicator/share/
vi /opt/replicator/share/convertstringfrommysql.json
```

Once the `convertstringfrommysql` JSON configuration file has been edited, update the `/etc/tungsten/tungsten.ini` file to add and configure any addition options needed for the specific datawarehouse you are using.

2. Create the configuration file `/etc/tungsten/tungsten.ini` on the destination DBMS host, i.e. **host6**:

```
[defaults]
user=tungsten
install-directory=/opt/replicator
replication-user=tungsten
replication-password=secret
replication-port=3306
profile-script=~/.bashrc
mysql-allow-intensive-checks=true
start-and-report=true

[alpha]
topology=cluster-alias
master=host1
members=host1,host2,host3
thl-port=2112

[omega]
topology=cluster-slave
relay=host6
relay-source=alpha
repl-svc-remote-filters=convertstringfrommysql
property=replicator.filter.convertstringfrommysql.definitionsFile=/opt/replicator/share/convertstringfrommysql.json
```
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.

- **[defaults]**

  - defaults indicates that we are setting options which will apply to all cluster dataservices.

- **user=tungsten [477]**

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

- **install-directory=/opt/replicator [456]**

  The installation directory of the Tungsten Replicator service. This is where the replicator software will be installed on the destination DBMS server.

- **replication-user=tungsten [470]**

  The MySQL user name to use when connecting to the MySQL database.

- **replication-password=secret [469]**

  The MySQL password for the user that will connect to the MySQL database.

- **replication-port=3306 [469]**

  The TCP/IP port on the destination DBMS server that is listening for connections.

- **start-and-report=true [472]**

  Tells tpm to startup the service, and report the current configuration and status.

- **profile-script=~/.bashrc [467]**

  Tells tpm to add PATH information to the specified script to initialize the Tungsten Replicator environment.

- **[alpha]**

  - alpha is the name and identity of the source cluster alias being created.

  This definition is for a dataservice alias, not an actual dataservice because topology=cluster-alias [476] has been specified. This alias is used in the cluster-slave section to define the source hosts for replication.

- **topology=cluster-alias [476]**

  Define this as a cluster dataservice alias so tpm does not try to install cluster software to the hosts.

- **members=host1,host2,host3 [460]**

  A comma separated list of all the hosts that are part of this cluster dataservice.

- **master=host1 [459]**

  The hostname of the server that is the current cluster master MySQL server.

- **thl-port=2112 [476]**

  The THL port for the cluster. The default value is 2112 but any other value must be specified.

- **[omega]**

  - omega is the unique service name for the replication stream from the cluster.

  This replication service will extract data from cluster dataservice alpha and apply into the database on the DBMS server specified by relay=host6 [459].

- **topology=cluster-slave [476]**

  Tells tpm this is a cluster-slave replication service which will have a list of all source cluster nodes available.
The hostname of the destination DBMS server.

- `relay-source=alpha [469]`

Specifies the name of the source cluster dataservice alias (defined above) that will be used to read events to be replicated.

### Important

The `cluster-alias` name [i.e. `alpha`] MUST be the same as the cluster dataservice name that you are replicating from.

### Note

Do not include `start-and-report=true [472]` if you are taking over for MySQL native replication. See Section 5.12.1, “Migrating from MySQL Native Replication ‘In-Place’” for next steps after completing installation.

3. Now finish configuring the `omega` dataservice with the options specific to the datawarehouse target in use.

Append the appropriate code snippet below to the bottom of the existing `[omega]` stanza:

- **AWS RedShift Target - Offboard Batch Applier**

```plaintext
batch-enabled=true
dataservice-type=redshift_enable-heterogeneous-slave=true
replication-host=REDSHIFT_ENDPOINT_FQDN_HERE
replication-password=REDSHIFT_PASSWORD_HERE
replication-user=REDSHIFT_PASSWORD_HERE
redshift-dbname=REDSHIFT_DB_NAME_HERE
svc-applier-block-commit-interval=10s
svc-applier-block-commit-size=5
```

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

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

- **--topology=cluster-slave [476]**

  Configure the topology as a cluster-slave. This will configure the individual replicator as a slave of all the nodes in the cluster, as defined in the previous configuration of the cluster topology.

- **--relay [459]**

  Configure the node as the relay for the cluster which will replicate data into the datawarehouse.

- **--enable-heterogeneous-slave=true [453]**

  Configures the slave to correctly process the incoming data so that it can be written to the datawarehouse. This includes correcting the processing of text data types and configuring the appropriate filters.

- **--replication-host [469]**

  The target host for writing data. In the case of Redshift, this is the fully qualified hostname of the Redshift host.

- **--replication-user [470]**

  The user within the Redshift service that will be used to write data into the database.

- **--replication-password=password [469]**

  The password for the user within the Redshift service that will be used to write data into the database.

- **--datasource-type=redshift [449]**

  Set the datasource type to be used when storing information about the replication state.

- **--batch-enabled=true [439]**

  Enable the batch service, this configures the JavaScript batch engine and CSV writing semantics to generate the data to be applied into a datawarehouse.
- `--batch-load-template=redshift` [440]
  The batch load template to be used. Since we are replicating into Redshift, the `redshift` template is used.

- `--redshift-dbname=dev` [468]
  The name of the database within the Redshift service where the data will be written.

Please see Install Amazon Redshift Applier [in Tungsten Replicator 6.1 Manual] for more information.

- Vertica Target - Onboard/Offboard Batch Applier

```
batch-enabled=true
batch-load-template=vertica
batch-load-language=js
datasource-type=vertica
disable-relay-logs=true
enable-heterogeneous-service=true
replication-user=dbadmin
replication-password=VERTICA_DB_PASSWORD_HERE
replication-host=VERTICA_HOST_NAME_HERE
replication-port=5433
csv-applier-block-commit-interval=5s
csv-applier-block-commit-size=500
vertica-dbname=VERTICA_DB_NAME_HERE
```


- Elasticsearch Target - Onboard/Offboard Native Applier

```
datasource-type=elasticsearch
replication-host=ELASTICSEARCH_HOST_NAME_HERE
replication-password=null
replication-port=9200
replication-user=root
```


- For additional targets, please see the full list at Deploying Appliers [in Tungsten Replicator 6.1 Manual], or click on some of the targets below:
  - Deploying the Oracle Applier [in Tungsten Replicator 6.1 Manual]
  - Deploying the MySQL Applier [in Tungsten Replicator 6.1 Manual]
  - Deploying the Vertica Applier [in Tungsten Replicator 6.1 Manual]

4. Download and install the latest Tungsten Replicator package (.rpm), or download the compressed tarball and unpack it on host6:

```
shell> cd /opt/continuent/software
shell> tar xzvf tungsten-replicator-6.0.3-599.tar.gz
```

5. Change to the Tungsten Replicator staging directory:

```
shell> cd tungsten-replicator-6.0.3-599
```

6. Run tpm to install the Tungsten Replicator software with the INI-based configuration:

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

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

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

The cluster-slave replicator should now be installed and ready to use.

3.10. Migrating and Seeding Data

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

If you are migrating an existing MySQL native replication deployment to use Tungsten Clustering the configuration of the Tungsten Clustering replication must be updated to match the status of the slave.
1. Deploy Tungsten Clustering using the model or system appropriate according to Chapter 2, Deployment. Ensure that the Tungsten Clustering is not started automatically by excluding the `--start` or `--start-and-report` 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
   shell> trepctl -service alpha reset -all -y
   ```

4. On the master
   Login and start Tungsten Clustering services and put the Tungsten Replicator online:
   ```
   shell> startall
   shell> trepctl online
   ```

5. On the master
   Put the cluster into maintenance mode using `cctrl` to prevent Tungsten Clustering 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
   Last_Error: Error executing row event: 'Table 'tungsten_alpha.heartbeat' doesn't exist'
   Exec_Master_Log_Pos: 181268
   ```

   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 Tungsten Clustering 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
   shell> 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.
   ```
   shell> trepctl online -from-event 000025:181268
   ```

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

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 Tungsten Clustering services:

```shell
trepctl online
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
trepctl online -from-event 000025:188249
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, Tungsten Clustering should be operating as the replication service for your MySQL servers. Use the information in Chapter 5, Operations Guide to monitor and administer the service.

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

When running an existing MySQL native replication service that needs to be migrated to a Tungsten Clustering service, one solution is to create the new Tungsten Clustering 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 3.9, "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`.

**Figure 3.9. 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;
```
Deployment: MySQL Topologies

Obtain the current slave position within the master binary log:

```
mysql> SHOW SLAVE STATUS|
  ...  
    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 Tungsten Clustering 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 3.7, “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.

### 3.10.3. 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 -hhost3 -utungsten -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 -hhost3 -utungsten -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
```

### 3.10.4. Seeding Data through `tungsten_provision_thl`

The `tungsten_provision_thl` command creates a temporary replicator (with identical configuration), and MySQL server (using MySQL Sandbox) through which a `mysqldump` creates a the THL in ROW-based format suitable for seeding any target database, but which is particularly suited to generating target data in the correct format for heterogeneous replication deployments.
Chapter 4. Deployment: Advanced

The following sections provide guidance and instructions for creating advanced deployments, including configuration automatic startup and shutdown during boot procedures, upgrades, downgrades, and removal of Tungsten Clustering.

4.1. 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.

4.1.1. 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.

4.1.2. Enabling Parallel Apply During Install

Parallel apply is enabled using the `svc-parallelization-type` and `channels` options of `tpm`. The parallelization type defaults to `none` which is to say that parallel apply is disabled. You should set it to `disk`. The `channels` option sets the number of channels (i.e., threads) you propose to use for applying data. Here is a code example of a MySQL Applier installation with parallel apply enabled. The slave will apply transactions using 30 channels.

```
show staging

Show INI

shell> ./tools/tpm configure defaults \
    --reset \ 
    --install-directory=/opt/continuent \ 
    --user=tungsten \ 
    --mysql-allow-intensive-checks=true \ 
    --profile-script=~/.bash_profile \ 
    --application-port=3306 \ 
    --application-user=app_user \ 
    --application-password=secret \ 
    --replication-port=13306 \ 
    --replication-user=tungsten \ 
    --replication-password=secret \ 
    --svc-parallelization-type=disk \ 
    --channels=10 \ 
    --start-and-report=true

shell> ./tools/tpm configure alpha \ 
    --master=host1 \ 
    --members=host1,host2,host3 \ 
    --connectors=host1,host2,host3 \ 
    --topology=clustered

shell> vi /etc/tungsten/tungsten.ini
```
Deployment: Advanced

**Configuration group defaults**

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

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

- **--reset [470]**
  - reset [470]
  
  For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- **--install-directory=/opt/continuent [456]**
  - install-directory=/opt/continuent [456]
  
  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- **--user=tungsten [477]**
  - user=tungsten [477]
  
  System User

- **--mysql-allow-intensive-checks=true [462]**
  - mysql-allow-intensive-checks=true [462]
  
  For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility. This includes checking each table definition individually for any unsupported data types.

- **--profile-script=~/.bash_profile [467]**
  - profile-script=~/.bash_profile [467]
  
  Append commands to include env.sh in this profile script

- **--application-port=3306 [437]**
  - application-port=3306 [437]
  
  Port for the connector to listen on

- **--application-user=app_user [437]**
  - application-user=app_user [437]
  
  Database username for the connector

- **--application-password=secret [437]**
  - application-password=secret [437]
  
  Database password for the connector
• --replication-port=13306

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

• --replication-user=tungsten

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

• --replication-password=secret

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

• --svc-parallelization-type=disk

Method for implementing parallel apply

• --channels=10

Number of replication channels to use for services

• --start-and-report=true

Start the services and report out the status after configuration

Configuration group alpha

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

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

• --master=host1

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

• --members=host1,host2,host3

Hostnames for the dataservice members

• --connectors=host1,host2,host3

Hostnames for the dataservice connectors

• --topology=clustered

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

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

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

• buffer-size — Sets the replicator block commit size, which is the number of transactions to commit at once on slaves. Values up to 100 are normally fine.
• native-slave-takeover [464] — Used to allow Tungsten to take over from native MySQL replication and parallelize it. See here for more.

You can check the number of active channels on a slave by looking at the "channels" property once the replicator restarts.

```shell
slave shell> trepctl -service alpha status | grep channels
channels               : 10
```

**Important**

The channel count for a Master will ALWAYS be 1 because extraction is single-threaded:

```shell
master shell> trepctl -service alpha status | grep channels
channels               : 1
```

**Warning**

Enabling parallel apply will dramatically increase the number of connections to the database server.

Typically the calculation on a slave would be: Connections = Channel_Count x Service_Count x 2, so for a 4-way Composite Multimaster topology with 30 channels there would be 30 x 4 x 2 = 240 connections required for the replicator alone, not counting application traffic.

You may display the currently used number of connections in MySQL:

```sql
mysql> SHOW STATUS LIKE 'max_used_connections';
```

<table>
<thead>
<tr>
<th>Variable_name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max_used_connections</td>
<td>190</td>
</tr>
</tbody>
</table>

Below are suggestions for how to change the maximum connections setting in MySQL both for the running instance as well as at startup:

```sql
mysql> SET GLOBAL max_connections = 512;
```

```sql
mysql> SHOW VARIABLES LIKE 'max_connections';
```

<table>
<thead>
<tr>
<th>Variable_name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_connections</td>
<td>512</td>
</tr>
</tbody>
</table>

```shell
shell> vi /etc/my.cnf
#max_connections = 151
max_connections = 512
```

### 4.1.3. 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.

4.1.4. Parallel Replication and Offline Operation

4.1.4.1. Clean Offline Operation

When you issue a trepctl offline command, Tungsten Replicator will bring all channels to the same point in the log and then go offline. This is known as going offline cleanly. When a slave has been taken offline cleanly the following are true:

- The `trep_commit_seqno` table contains a single row
- The `trep_shard_channel` table is empty

When parallel replication is not enabled, you can take the replicator offline by stopping the replicator process. There is no need to issue a `trepctl offline` command first.

4.1.4.2. Tuning the Time to Go Offline Cleanly

Putting a replicator offline may take a while if the slowest and fastest channels are far apart, i.e., if one channel gets far ahead of another. The separation between channels is controlled by the `maxOfflineInterval` parameter, which defaults to 5 seconds. This sets the allowable distance between commit timestamps processed on different channels. You can adjust this value at installation or later. The following example shows how to change it after installation. This can be done at any time and does not require the replicator to go offline cleanly.

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

Show Staging

Show INI

Run the `tpm` command to update the software with the Staging-based configuration:

```
shell> ./tools/tpm query staging
```

Run the `tpm` command to update the software with the INI-based configuration:

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

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

Run the `tpm` command to update the software with the INI-based configuration:
For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

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 downside is that going offline cleanly can become quite slow.

4.1.4.3. Unclean Offline

If you need to take a replicator offline quickly, you can either stop the replicator process or issue the following command:

d 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.

4.1.5. Adjusting Parallel Replication After Installation

4.1.5.1. How to Enable Parallel Apply After Installation

Warning

Be sure to place the cluster into MAINTENANCE mode first so the Manager does not attempt to automatically bring the replicator online.

cctrl> set policy maintenance

To enable parallel replication after installation, take the replicator offline cleanly using the following command:

d shell> trepctl offline

Modify the configuration to add two parameters:

Show Staging

Show INI

d shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.3-41
shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten
shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.3-41
shell> ssh {STAGING_USER}@{STAGING_HOST}
shell> cd {STAGING_DIRECTORY}

Run the tpm command to update the software with the Staging-based configuration:

d shell> /tools/tpm update

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

[defaults]
...
svc-parallelization-type=disk
channels=10
Run the `tpm` command to update the software with the INI-based configuration:

```bash
shell> tpm query staging
```

tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41

```bash
shell> echo The staging DIRECTORY is `tpm query staging | cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
```

```bash
shell> cd (STAGING_DIRECTORY)
```

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

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

**Note**

*You make use an actual data service name in place of the keyword `defaults`.*

Signal the changes by a complete restart of the Replicator process:

```bash
shell> replicator restart
```

**Warning**

Be sure to place the cluster into AUTOMATIC mode as soon as all replicators are updated and back online.

```bash
cctrl> set policy automatic
```

You can check the number of active channels on a slave by looking at the “channels” property once the replicator restarts.

```bash
slave shell> trepctl -service alpha status | grep channels
channels               : 10
```

**Important**

The channel count for a Master will ALWAYS be 1 because extraction is single-threaded:

```bash
master shell> trepctl -service alpha status | grep channels
channels               : 1
```

**Warning**

Enabling parallel apply will dramatically increase the number of connections to the database server.

Typically the calculation on a slave would be: Connections = Channel_Count x Service_Count x 2, so for a 4-way Composite Multimaster topology with 30 channels there would be 30 x 4 x 2 = 240 connections required for the replicator alone, not counting application traffic.

You may display the currently used number of connections in MySQL:

```sql
mysql> SHOW STATUS LIKE 'max_used_connections';
+----------------------+-------+
<table>
<thead>
<tr>
<th>Variable_name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max_used_connections</td>
<td>190</td>
</tr>
</tbody>
</table>
+----------------------+-------+
1 row in set (0.00 sec)
```

Below are suggestions for how to change the maximum connections setting in MySQL both for the running instance as well as at startup:

```sql
mysql> SET GLOBAL max_connections = 512;
```

```sql
mysql> SHOW VARIABLES LIKE 'max_connections';
+-----------------+-------+
<table>
<thead>
<tr>
<th>Variable_name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_connections</td>
<td>512</td>
</tr>
</tbody>
</table>
+-----------------+-------+
1 row in set (0.00 sec)
```

```bash
shell> vi /etc/my.cnf
```

```bash
#max_connections = 151
max_connections = 512
```

### 4.1.5.2. How to Change Channels Safely

To change the number of channels you must take the replicator offline cleanly using the following command:
Deployment: Advanced

```
shell> trepctl offline
```

This command brings all channels up the same transaction in the log, then goes offline. If you look in the `trep_commit_seqno` table, you will notice only a single row, which shows that updates to the slave have been completely serialized to a single point. At this point you may safely reconfigure the number of channels on the replicator, for example using the following command:

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

Show Staging

Show INI

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.3-41
```

```
shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten
```

```
shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1
```

```
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.3-41
```

```
shell> ssh {STAGING_USER}@{STAGING_HOST}
```

```
shell> cd {STAGING_DIRECTORY}
```

```
shell> ./tools/tpm configure alpha \
   --channels=5
```

Run the `tpm` command to update the software with the Staging-based configuration:

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

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

```
[alpha]
...
channels=5
```

Run the `tpm` command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.3-41
```

```
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.3-41
```

```
shell> cd (STAGING_DIRECTORY)
```

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

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

You can check the number of active channels on a slave by looking at the “channels” property once the replicator restarts.

If you attempt to reconfigure channels without going offline cleanly, Tungsten Replicator will signal an error when you attempt to go online with the new channel configuration. The cure is to revert to the previous number of channels, go online, and then go offline cleanly. Note that attempting to clean up the `trep_commit_seqno` and `trep_shard_channel` tables manually can result in your slaves becoming inconsistent and requiring full resynchronization. You should only do such cleanup under direction from Continuent support.

```
Warning
```

Failing to follow the channel reconfiguration procedure carefully may result in your slaves becoming inconsistent or failing. The cure is usually full resynchronization, so it is best to avoid this if possible.

4.1.5.3. How to Disable Parallel Replication Safely

The following steps describe how to gracefully disable parallel apply replication.

Replication Graceful Offline [critical first step]

To disable parallel apply, you must first take the replicator offline cleanly using the following command:

```
shell> trepctl offline
```
This command brings all channels up the same transaction in the log, then goes offline. If you look in the `trep_commit_seqno` table, you will notice only a single row, which shows that updates to the slave have been completely serialized to a single point. At this point you may safely disable parallel apply on the replicator, for example using the following command:

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

Show Staging

Show INI

```
shell> tpm query staging
staging@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41

shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten

shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41

shell> ssh {STAGING_USER}@{STAGING_HOST}

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm configure alpha 
--svc-parallelization-type=none 
--channels=1

Run the `tpm` command to update the software with the Staging-based configuration:

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

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

```
[alpha]
  svc-parallelization-type=none
  channels=1
```

Run the `tpm` command to update the software with the INI-based configuration:

```
shell> tpm query staging
staging@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

Verification

You can check the number of active channels on a slave by looking at the "channels" property once the replicator restarts.

```
shell> trepctl -service alpha status| grep channels
channels : 1
```

Notes and Warnings

If you attempt to reconfigure channels without going offline cleanly, Tungsten Replicator will signal an error when you attempt to go online with the new channel configuration. The cure is to revert to the previous number of channels, go online, and then go offline cleanly. Note that attempting to clean up the `trep_commit_seqno` and `trep_shard_channel` tables manually can result in your slaves becoming inconsistent and requiring full resynchronization. You should only do such cleanup under direction from Continuent support.

```
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.
```

4.1.5.4. How to Switch Parallel Queue Types Safely

As with channels you should only change the parallel queue type after the replicator has gone offline cleanly. The following example shows how to update the parallel queue type after installation:
4.1.6. Monitoring Parallel Replication

Basic monitoring of a parallel deployment can be performed using the techniques in Chapter 5, Operations Guide. Specific operations for parallel replication are provided in the following sections.

4.1.6.1. Useful Commands for Parallel Monitoring Replication

The replicator has several helpful commands for tracking replication performance:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trepctl status</td>
<td>Shows basic variables including overall latency of slave and number of apply channels</td>
</tr>
<tr>
<td>trepctl status -name shards</td>
<td>Shows the number of transactions for each shard</td>
</tr>
<tr>
<td>trepctl status -name stores</td>
<td>Shows the configuration and internal counters for stores between tasks</td>
</tr>
<tr>
<td>trepctl status -name tasks</td>
<td>Shows the number of transactions [events] and latency for each independent task in the replicator pipeline</td>
</tr>
</tbody>
</table>

4.1.6.2. Parallel Replication and Applied Latency On Slaves

The `trepctl status` appliedLastSeqno parameter shows the sequence number of the last transaction committed. Here is an example from a slave with 5 channels enabled.

```
shell> trepctl status
```

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

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

Show Staging

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
shell> echo The staging USER is 'tpm query staging| cut -d: -f1 | cut -d@ -f1'
The staging USER is tungsten
shell> echo The staging HOST is 'tpm query staging| cut -d: -f1 | cut -d@ -f2'
The staging HOST is db1
shell> echo The staging DIRECTORY is 'tpm query staging| cut -d: -f2'
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
shell> ssh (STAGING_USER)@ (STAGING_HOST)
shell> cd (STAGING_DIRECTORY)
shell> ./.tools/tpm configure alpha \\n   --svc-parallelization-type=disk \\n   --channels=5
```

Run the `tpm` command to update the software with the Staging-based configuration:

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

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

```
[alpha]
   svc-parallelization-type=disk
   channels=5
```

Run the `tpm` command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
shell> echo The staging DIRECTORy is 'tpm query staging| cut -d: -f2'
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
shell> cd (STAGING_DIRECTORY)
shell> ./.tools/tpm update
```

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”. 

```
```
Deployment: Advanced

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:

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

**Show Staging**

**Show INI**

```
shell> tpm query staging
```

```
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1` The staging USER is tungsten
```

```
shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2` The staging HOST is db1
```

```
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2` The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
shell> ssh {STAGING_USER}@{STAGING_HOST}
```

```
shell> cd {STAGING_DIRECTORY}
```

```
shell> ./tools/tpm configure alpha \
   --property=replicator.store.parallel-queue.syncInterval=500
```

Run the `tpm` command to update the software with the Staging-based configuration:

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

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, "Configuration Changes from a Staging Directory".

```
[alpha]
```

```
... property=replicator.store.parallel-queue.syncInterval=500
```

Run the `tpm` command to update the software with the INI-based configuration:

```
shell> tpm query staging
```

```
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2` The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
shell> cd (STAGING_DIRECTORY)
```

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

For information about making updates when using an INI file, please see Section 9.4.4, "Configuration Changes with an INI file".

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.

### 4.1.6.3. Relative Latency

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

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

In this example the last transaction had a latency of .571 seconds from the time it committed on the master and committed 8.944 seconds ago. If relative latency increases significantly in a busy system, it may be a sign that replication is stalled. This is a good parameter to check in monitoring scripts.

### 4.1.6.4. Serialization Count

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

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

```shell
shell> trepctl status -name stores
Processing status command (stores)...
... NAME VALUE
... ---- ----- 
... criticalPartition : -1
... discardCount : 0
... estimatedOfflineInterval : 0.0
... eventCount : 1512
... headSeqno : 78022
... 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.

### 4.1.6.5. Maximum Offline Interval

The maximum offline interval (`maxOfflineInterval`) parameter controls the “distance” between the fastest and slowest channels when parallel apply is enabled. The replicator measures distance using the seconds between commit times of the last transaction processed on each channel. This time is roughly equivalent to the amount of time a replicator will require to go offline cleanly.

You can change the `maxOfflineInterval` as shown in the following example, the value is defined in seconds.

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

Show INI

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten
shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
shell> cd $(STAGING_USER)@$(STAGING_HOST)
shell> ./tools/tpm configure alpha \
    --property=replicator.store.parallel-queue.maxOfflineInterval=30
```

Run the `tpm` command to update the software with the Staging-based configuration:

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

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

```
[alpha]
...
property=replicator.store.parallel-queue.maxOfflineInterval=30
```

Run the `tpm` command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
shell> cd $(STAGING_DIRECTORY)
shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

You can view the configured value as well as the estimate current value using the `trepctl status -name stores` command, as shown in yet another example:

```
shell> trepctl status -name stores
Processing status command (stores)...
NAME         VALUE
...          ...
estimatedOfflineInterval: 1.3
...          
maxOfflineInterval : 30
...          
Finished status command (stores)...
```

4.1.6.6. Workload Distribution

Parallel apply works best when transactions are distributed evenly across shards and those shards are distributed evenly across available channels. You can monitor the distribution of transactions over shards using the `trepctl status -name shards` command. This command lists transaction counts for all shards, as shown in the following example.

```
shell> trepctl status -name shards
Processing status command (shards)...
...          ...
NAME         VALUE
...          ...
appliedLastEventId: mysql-bin.000211:0000000020095076:0
appliedLastSeqno  : 78023
appliedLatency    : 0.255
eventCount        : 3523
shardId           : cust1
stage             : q-to-dbms
...          
Finished status command (shards)...
```
If one or more shards have a very large `eventCount` value compared to the others, this is a sign that your transaction workload is poorly distributed across shards.

The listing of shards also offers a useful trick for finding serialized transactions. Shards that Tungsten Replicator cannot safely parallelize are assigned the dummy shard ID `#UNKNOWN`. Look for this shard to find the count of serialized transactions. The `appliedLastSeqno` for this shard gives the sequence number of the most recent serialized transaction. As the following example shows, you can then list the contents of the transaction to see why it serialized. In this case, the transaction affected tables in different schemas.

```
shell> trepctl status -name shards
Processing status command (shards)...
NAME      VALUE
----      -----.....
appliedLastEventId: mysql-bin.800211:8000008820095529:0
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.800211:8000008820095529:0
- 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.800211:8000008820095076:0
appliedLastSeqno  : 78023
appliedLatency    : 0.248
applyTime         : 0.003
averageBlockSize  : 2.520
cancelled         : false
currentLastEventId: mysql-bin.800211:8000008820095076:0
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.

### 4.1.7. Controlling Assignment of Shards to Channels

Tungsten Replicator by default assigns channels using a round robin algorithm that assigns each new shard to the next available channel. The current shard assignments are tracked in table `trep_shard_channel` in the Tungsten catalog schema for the replication service.

For example, if you have 2 channels enabled and Tungsten processes three different shards, you might end up with a shard assignment like the following:

```
foo => channel 0
```
Deployment: Advanced

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>.
db1=0
db2=0
db3=0

db4=1

db5=2

db6=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

# Comm-separated list of shards that require critical section to run.
# A 'critical section' means that these events are single-threaded to
# ensure that all dependencies are met.
#*(critical)=common1,common2

# Method for channel hash assignments. Allowed values are round-robin and
# string-hash.
(hash-method)=round-robin
```

You can update the shard.list file to do three types of custom overrides.

1. Change the hashing method for channel assignments. Round-robin uses the trep_shard_channel table. The string-hash method just hashes the shard name.

2. Assign shards to explicit channels. Add lines of the form shard=channel to the file as shown by the commented-out entries.

3. Define critical shards. These are shards that must be processed in serial fashion. For example if you have a sharded application that has a single global shard with reference information, you can declare the global shard to be critical. This helps avoid applications seeing out of order information.

Changes to shard.list must be made with care. The same cautions apply here as for changing the number of channels or the parallelization type. For subscription customers we strongly recommend conferring with Continuent Support before making changes.

4.1.8. Disk vs. Memory Parallel Queues

Channels receive transactions through a special type of queue, known as a parallel queue. Tungsten offers two implementations of parallel queues, which vary in their performance as well as the requirements they may place on hosts that operate parallel apply. You choose the type of queue to enable using the --svc-parallelization-type option.

**Warning**

Do not change the parallel queue type without setting the replicator offline cleanly. See the procedure later in this page for more information.

**Disk Parallel Queue** [disk option]

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

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

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

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

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

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

Show Staging

Show INI

Run the tpm command to update the software with the Staging-based configuration:

Run the tpm command to update the software with the INI-based configuration:

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, "Configuration Changes from a Staging Directory".

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.

### 4.2. Starting and Stopping Tungsten Clustering

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

```
shell> stopall
Stopping Tungsten Connector...
Stopped Tungsten Connector.
Stopping Tungsten Replicator Service...
Stopped Tungsten Replicator Service.
Stopping Tungsten Manager Service...
```
4.2.1. Restarting the Replicator Service

**Warning**

Restarting a running replicator temporarily stops and restarts replication. Either set `MAINTENANCE` mode within `cctrl` [see Section 5.15, “Performing Database or OS Maintenance” or shun the datasource before restarting the replicator (Section 5.3.5.1, “Shunning a Datasource”).

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

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

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

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

4.2.2. Restarting the Connector Service

**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:

```sh
shell> connector stop
Stopping Tungsten Connector Service...
Stopped Tungsten Connector Service.
```

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

```sh
shell> connector start
Starting Tungsten Connector Service...
Waiting for Tungsten Connector Service......
running: PID:12338
```

If the cluster was configured with `auto-enable=false [437]` then you will need to put each node online individually.

4.2.3. Restarting the Manager Service

The manager service is designed to monitor the status and operation of the each of the datasources 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:
   ```sh
   [LOGICAL:EXPERT]/dsone > set policy maintenance
   ```

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

### 4.2.4. Restarting the Multisite/Multimaster Replicator Service

**Warning**

Restarting a running replicator temporarily stops and restarts replication. When using Multisite/Multimaster, restarting the additional replicator will stop replication between sites.

These instructions assume you have installed the additional replicator with the `--executable-prefix=mm` option. If not, you should go to `/opt/replicator/tungsten/tungsten-replicator/bin` and run the `replicator` command directly.

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

```
$ mm_replicator stop
Stopping Tungsten Replicator Service...
Stopped Tungsten Replicator Service.
```

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

```
$ mm_replicator start
Starting Tungsten Replicator Service...
```

### 4.3. Configuring Startup on Boot

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

```
$ 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/rc6.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/rc6.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/rc6.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:

```
$ sudo /opt/continuent/tungsten/cluster-home/bin/undeployall
```

### 4.3.1. Configuring Multisite/Multimaster Replicator Startup on Boot

Because there is an additional Tungsten Replicator running, each must be individually configured to startup on boot:

- For the Tungsten Clustering service, use Section 4.3, “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 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`):

```
$ sudo ln -s /opt/replicator/tungsten/tungsten-replicator/bin/replicator /etc/init.d/mmreplicator
```
2. Stop the Tungsten Replicator process. Failure to do this will cause issues because the service will no longer recognize the existing PID file and report it is not running.

   ```
   $ /etc/init.d/mmreplicator stop
   ```

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

   ```
   APP_NAME="mmreplicator"
   ```

4. Start the Tungsten Replicator process.

   ```
   $ /etc/init.d/mmreplicator start
   ```

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

   On Debian/Ubuntu:
   ```
   $ sudo update-rc.d mmreplicator defaults
   ```
   On RedHat/CentOS:
   ```
   $ sudo chkconfig --add mmreplicator
   ```

### 4.4. Upgrading Tungsten Clustering

To upgrade an existing installation on Tungsten Clustering, 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.

#### 4.4.1. Upgrading using the Staging Method [with ssh Access]

**Note**

- For INI file upgrades, see Section 4.4.2, "Upgrading when using INI-based configuration, or without ssh Access"

**Warning**

- Before performing and upgrade, please ensure that you have checked the Appendix B, Prerequisites, as software and system requirements may have changed between versions and releases.

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:

   ```
   $ tar zxf tungsten-clustering-6.1.3-43.tar.gz
   ```

3. Change to the extracted directory:

   ```
   $ cd tungsten-clustering6.1.3-43
   ```

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`:

     ```
     $ ./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 Tungsten Clustering 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
```

5. Check that the update configuration matches what you expect by using `tpm reverse`:

```
shell> /tools/tpm reverse
```

6. Run the upgrade process:

```
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 Tungsten Clustering 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:

```
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
Tungsten Clustering (For MySQL) 6.1.3 build 43
connect to 'dsone@host1'
dsone: session established
[LOGICAL]/dsone > ls
COORDINATOR[host3:AUTOMATIC:ONLINE]
```
The update process should now be complete. The current version can be confirmed by starting `cctrl`.

4.4.2. 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.

**Warning**

Before performing and upgrade, please ensure that you have checked the Appendix B, Prerequisites, as software and system requirements may have changed between versions and releases.

**Important**

Application traffic to the nodes will be disconnected when the connector restarts. Use the `--no-connectors` `tpm` option when you upgrade to prevent the connectors from restarting until later when you want them to.

4.4.2.1. Upgrading Using the No-Switch Method

To use the no-switch method of upgrading:

1. Place the cluster into maintenance mode
2. Upgrade the slaves in the dataservice. Be sure to shun and welcome each slave.
3. Upgrade the master node

**Important**

Replication traffic to the slaves will be delayed while the replicator restarts. The delays will increase if there are a large number of stored events in the THL. Old THL may be removed to decrease the delay. Do NOT delete THL that has not been received on all slave nodes or events will be lost.

4. Upgrade the connectors in the dataservice one-by-one

**Important**

Application traffic to the nodes will be disconnected when the connector restarts.

5. Place the cluster into automatic mode

4.4.2.2. Upgrading a Single Host using `tpm`

**Note**

For more information on performing maintenance across a cluster, see Section 5.15.3, “Performing Maintenance on an Entire Dataservice”.

...
To upgrade a single host using the `tpm` command:

1. Download the release package.

2. Unpack the release package:

   ```shell
   tar zxf tungsten-clustering-6.1.3-43.tar.gz
   ```

3. Change to the extracted directory:

   ```shell
cd tungsten-clustering-6.1.3-43
   ```

4. Execute `tpm update`, specifying the installation directory. This will update only this host:

   ```shell
   ./tools/tpm update --directory=/opt/continuent
   ```

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

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

### 4.4.3. Upgrade: From v4 or v5 to v6 Composite Multimaster Clusters

These steps are specifically for the safe and successful upgrade of an existing v4.0.7 or later, and v5.0.0 MSMM topology, to the new Multi-master topology.

**Warning**

It is very important to follow all the below steps and ensure full backups are taken when instructed. These steps can be destructive and without proper care and attention, data loss, data corruption or a split-brain scenario can happen.

**Warning**

Parallel apply MUST be disabled before starting your upgrade. You may re-enable it once the upgrade has been fully completed. See Section 4.1.5.3, “How to Disable Parallel Replication Safely” and Section 4.1.2, “Enabling Parallel Apply During Install” for more information.

**Note**

The examples in this section are based on three clusters named ‘nyc’, ‘london’ and ‘tokyo’

If you do not have exactly three clusters, please adjust this procedure to match your environment.

Click here for a video of the upgrade procedure, showing the full process from start to finish...

If you are currently installed using a staging-based installation, you must convert to an INI based installed, since INI based installation is the only option supported for the composite multi master deployments. For notes on how to perform the staging to INI file conversion using the `translatetoini.pl` script, please visit Section 9.4.6, "Using the `translatetoini.pl` Script".

Click here for a video of the INI conversion procedure, showing the full process from start to finish...
4.4.3.1. Supported Upgrade Paths

The only method to upgrade an existing MSMM installation is via the ini method, therefore customers currently deployed via Staging will need to convert to an ini deployment first.

<table>
<thead>
<tr>
<th>Path</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>ini, in place</td>
<td>Yes</td>
</tr>
<tr>
<td>ini, with master switch</td>
<td>No</td>
</tr>
<tr>
<td>Staging</td>
<td>No</td>
</tr>
<tr>
<td>Staging, with --no-connectors</td>
<td>No</td>
</tr>
</tbody>
</table>

4.4.3.2. Upgrade Prerequisites

**Warning**

Parallel apply MUST be disabled before starting your upgrade. You may re-enable it once the upgrade has been fully completed. See Section 4.1.5.3, “How to Disable Parallel Replication Safely” and Section 4.1.2, “Enabling Parallel Apply During Install” for more information.

- Obtain the latest v6 Tungsten Clustering software build and place it within `/opt/continuent/software`
- Extract the package
- The examples below refer to the `tungsten_prep_upgrade` script, this can be located in the extracted software package within the `tools` directory.

4.4.3.3. Upgrade Step 1: Backups

Take a full and complete backup of one node - this can be a slave, and preferably should be either performed by:

- Percona xtrabackup whilst database is open
- Manual backup of all datafiles after stopping the database instance

4.4.3.4. Upgrade Step 2: Stop the Cross-site Replicators

**Important**

Typically the cross-site replicators will be installed within `/opt/replicator`, if you have installed this in a different location you will need to pass this to the script in the examples using the `--path` option

1. The following commands tell the replicators to go offline at a specific point, in this case when they receive an explicit heartbeat. This is to ensure that all the replicators stop at the same sequence number and binary log position. The replicators will NOT be offline until the explicit heartbeat has been issued a bit later in this step.

   - On every nyc node:
     ```
     shell> ./tungsten_prep_upgrade -o
     or:
     shell> ./tungsten_prep_upgrade --service london --offline
     shell> ./tungsten_prep_upgrade --service tokyo --offline
     ```

   - On every london node:
     ```
     shell> ./tungsten_prep_upgrade -o
     or:
     shell> ./tungsten_prep_upgrade --service nyc --offline
     shell> ./tungsten_prep_upgrade --service tokyo --offline
     ```

   - On every tokyo node:
     ```
     shell> ./tungsten_prep_upgrade -o
     or:
     shell> ./tungsten_prep_upgrade --service london --offline
     shell> ./tungsten_prep_upgrade --service tokyo --offline
     ```

2. Next, on the master hosts within each cluster we issue the heartbeat, execute the following using the cluster-specific `trepctl`, typically in `/opt/continuent`:

   ```
   shell> trepctl heartbeat -name offline_for_upg
   ```
Ensure that every cross-site replicator on every node is now in the `OFFLINE:NORMAL` state:

```
shell> mmtrepctl status
or-
shell> mmtrepctl --service {servicename} status
```

3. Capture the position of the cross-site replicators on all nodes in all clusters.

The service name provided should be the name of the remote service(s) for this cluster, so for example in the `london` cluster you get the positions for `nyc` and `tokyo`, and in `nyc` you get the position for `london` and `tokyo`, etc.

- **On every `london` node:**
  ```
  shell> ./tungsten_prep_upgrade -g
  or-
  shell> ./tungsten_prep_upgrade --service nyc --get
  (NOTE: saves to ~/position-nyc-YYYYMMDDHHMMSS.txt)
  shell> ./tungsten_prep_upgrade --service tokyo --get
  (NOTE: saves to ~/position-tokyo-YYYYMMDDHHMMSS.txt)
  ```

- **On every `nyc` node:**
  ```
  shell> ./tungsten_prep_upgrade -g
  or-
  shell> ./tungsten_prep_upgrade --service london --get
  (NOTE: saves to ~/position-london-YYYYMMDDHHMMSS.txt)
  shell> ./tungsten_prep_upgrade --service tokyo --get
  (NOTE: saves to ~/position-tokyo-YYYYMMDDHHMMSS.txt)
  ```

- **On every `tokyo` node:**
  ```
  shell> ./tungsten_prep_upgrade -g
  or-
  shell> ./tungsten_prep_upgrade --service london --get
  (NOTE: saves to ~/position-london-YYYYMMDDHHMMSS.txt)
  shell> ./tungsten_prep_upgrade --service nyc --get
  (NOTE: saves to ~/position-nyc-YYYYMMDDHHMMSS.txt)
  ```

4. Finally, to complete this step, stop the replicators on all nodes:

```
shell> ./tungsten_prep_upgrade --stop
```

### 4.4.3.5. Upgrade Step 3: Export the `tungsten_*` Databases

On every node in each cluster, export the tracking schema for the cross-site replicator

Similar to the above step 2 when you captured the cross-site position, the same applies here, in `london` you export/backup `nyc` and `tokyo`, and in `nyc` you export/backup `london` and `tokyo`, and finally in `tokyo` you export/backup `nyc` and `london`.

- **On every `london` node:**
  ```
  shell> ./tungsten_prep_upgrade -d --alldb
  or-
  shell> ./tungsten_prep_upgrade --service nyc --dump
  shell> ./tungsten_prep_upgrade --service tokyo --dump
  ```

- **On every `nyc` node:**
  ```
  shell> ./tungsten_prep_upgrade -d --alldb
  or-
  shell> ./tungsten_prep_upgrade --service london --dump
  shell> ./tungsten_prep_upgrade --service tokyo --dump
  ```

- **On every `tokyo` node:**
  ```
  shell> ./tungsten_prep_upgrade -d --alldb
  or-
  shell> ./tungsten_prep_upgrade --service london --dump
  shell> ./tungsten_prep_upgrade --service nyc --dump
  ```

### 4.4.3.6. Upgrade Step 4: Uninstall the Cross-site Replicators

To uninstall the cross-site replicators, execute the following on every node:

```
shell> cd (replicator software path)
shell> tools/tpm uninstall --i-am-sure
```
4.4.3.7. Upgrade Step 5: Reload the tracking schema

We DO NOT want the reloading of this schema to appear in the binary logs on the Master, therefore the reload needs to be performed on each node individually:

- On every london node:
  ```shell
  ./tungsten_prep_upgrade -s nyc -u tungsten -w secret -r
  ```
  ```shell
  ./tungsten_prep_upgrade -s tokyo -u tungsten -w secret -r
  ```
  ```or```
  ```shell
  ./tungsten_prep_upgrade --service nyc --user tungsten --password secret --restore
  ```
  ```shell
  ./tungsten_prep_upgrade --service tokyo --user tungsten --password secret --restore
  ```

- On every tokyo node:
  ```shell
  ./tungsten_prep_upgrade -s london -u tungsten -w secret -r
  ```
  ```shell
  ./tungsten_prep_upgrade -s nyc -u tungsten -w secret -r
  ```
  ```or```
  ```shell
  ./tungsten_prep_upgrade --service london --user tungsten --password secret --restore
  ```
  ```shell
  ./tungsten_prep_upgrade --service nyc --user tungsten --password secret --restore
  ```

- On every nyc node:
  ```shell
  ./tungsten_prep_upgrade -s london -u tungsten -w secret -r
  ```
  ```shell
  ./tungsten_prep_upgrade -s tokyo -u tungsten -w secret -r
  ```
  ```or```
  ```shell
  ./tungsten_prep_upgrade --service london --user tungsten --password secret --restore
  ```
  ```shell
  ./tungsten_prep_upgrade --service tokyo --user tungsten --password secret --restore
  ```

4.4.3.8. Upgrade Step 6: Update Configuration

Update `/etc/tungsten/tungsten.ini` to a valid v6 config. An example of a valid config is as follows:

```ini
[defaults]
user=tungsten
home-directory=/opt/continuent
application-user=app_user
application-password=secret
application-port=3306
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=secret
mysql-allow-intensive-checks=true
skip-validation-check=THLSchemaChangeCheck
start-and-report=true

[nyc]
topology=clustered
master=db1
slaves=db2,db3
connectors=db1,db2,db3

[london]
topology=clustered
master=db4
slaves=db5,db6
connectors=db4,db5,db6

[tokyo]
topology=clustered
master=db7
slaves=db8,db9
connectors=db7,db8,db9

[global]
topology=composite-multi-master
composite-datasources=nyc,london,tokyo
```

4.4.3.9. Upgrade Step 7: Enter Maintenance Mode

Enable Maintenance mode on all clusters using the `cctrl` command:

```shell
shell> cctrl
shell> cctrl set policy maintenance
```

4.4.3.10. Upgrade Step 8: Stop Managers

Stop the manager process on all nodes:
4.4.3.11. Upgrade Step 9: Install the Software

Validate and install new release on all nodes:

```
shell> cd /opt/continent/software/tungsten-clustering-6.0.0-442
shell> tools/tpm validate-update
```

If validation shows no errors, run the install:

```
shell> tools/tpm update --replace-release
```

**Important**

If you had start-and-report=false you may need to restart manager services

**Warning**

Until all nodes have been updated, the output from cctrl may shows services in an OFFLINE, STOPPED, or UNKNOWN state. This is to be expected until all the new v6 managers are online

4.4.3.12. Upgrade Step 10: Start Managers

After the installation is complete on all nodes, start the manager services:

```
shell> manager start
```

4.4.3.13. Upgrade Step 11: Return to Automatic Mode

Return all clusters to Automatic mode using the cctrl command:

```
shell> cctrl
cctrl> set policy automatic
```

4.4.3.14. Upgrade Step 12: Validate

1. Identify the cross-site service name(s):

```
shell> trepctl services
```

In our example, the local cluster service will one of london, nyc or tokyo depending on the node you are on. The cross site replication services would be:

- **(within the london cluster)**
  - london_from_nyc
  - london_from_tokyo

- **(within the nyc cluster)**
  - nyc_from_london
  - nyc_from_tokyo

- **(within the tokyo cluster)**
  - tokyo_from_london
  - tokyo_from_nyc

2. Upon installation, the new cross-site replicators will come online, it is possible that they may be in an **OFFLINE:ERROR** state due to a change in Epoch numbers, check this on the master in each cluster by looking at the output from the trepctl command.

Check each service as needed based on the status seen above:

```
shell> trepctl -service london_from_nyc status
shell> trepctl -service london_from_tokyo status
```

3. If the replicator is in an error state due to an epoch difference, you will see an error similar to the following:

```
pendingErrorSeqno : -1
pendingExceptionMessage: Client handshake failure: Client response validation failed: log epoch numbers do not match: master source
```
The above error is due to the epoch numbers changing as a result of the replicators being restarted, and the new replicators being installed.

To resolve, simply force the replicator online as follows:

```
shell> trepctl -service london_from_nyc online -force
shell> trepctl -service london_from_tokyo online -force
or:
shell> trepctl -service nyc_from_london online -force
shell> trepctl -service nyc_from_tokyo online -force
or:
shell> trepctl -service tokyo_from_london online -force
shell> trepctl -service tokyo_from_nyc online -force
```

If the replicator shows an error state similar to the following:

```
pendingErrorSeqno : -1
pendingExceptionMessage: Client handshake failure: Client response validation failed: Master log does not contain requested transaction: master source ID=db1 client source ID=db2 requested seqno=1237 client epoch number=0 master min seqno=5 master max seqno=7
pipelineSource : UNKNOWN
```

The above error is possible if during install the slave replicators came online before the master.

Providing the steps above have been followed, just bringing the replicator online should be enough to get the replicator to retry and carry on successfully:

```
shell> trepctl -service london_from_nyc online
shell> trepctl -service london_from_tokyo online
or:
shell> trepctl -service nyc_from_london online
shell> trepctl -service nyc_from_tokyo online
or:
shell> trepctl -service tokyo_from_london online
shell> trepctl -service tokyo_from_nyc online
```

### Important

**Known Issue [CT-569]**

During an upgrade, the tpm process will incorrectly create additional, empty, tracking schemas based on the service names of the auto-generated cross-site services.

> For example, if your cluster has service names east and west, you should only have tracking schemas for tungsten_east and tungsten_west

In some cases, you will also see tungsten_east_from_west and/or tungsten_west_from_east

These tungsten_x_from_y tracking schemas will be empty and unused. They can be safely removed by issuing DROP DATABASE tungsten_x_from_y on a Master node, or they can be safely ignored.

### 4.4.4. Upgrading from Tungsten Clustering 1.5.3/1.5.4 to Tungsten Clustering 6.1

You can upgrade directly from Tungsten Clustering 1.5.3 or Tungsten Clustering 1.5.4 to Tungsten Clustering 6.1 using the standard update procedures.

When upgrading to Tungsten Clustering 6.1 from Tungsten Clustering 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 B.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 tungsten-clustering-6.1.3-43.tar.gz
   ```
3. Change into the directory:
   ```shell
cd tungsten-clustering-6.1.3-43
   ```
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.

### 4.4.5. Upgrade and Convert: v4 or v5 MultiMaster to v6 Composite Master-Slave

These steps are designed to guide you in the safe conversion of an existing v4 or v5 Multi-Site Multi-Master topology to a v6 Composite Master-Slave Topology, based on an ini installation.

For details of the difference between these two topologies, please review the following pages:

- Section 3.3, “Deploying Multisite/MultiMaster Clustering”
- Section 3.2, “Deploying Composite Master/Slave Clustering”

**Warning**

It is very important to follow all the below steps and ensure full backups are taken when instructed. These steps can be destructive and without proper care and attention, data loss, data corruption or a split-brain scenario can happen.

**Warning**

Parallel apply MUST be disabled before starting your upgrade. You may re-enable it once the upgrade has been fully completed. See Section 4.1.5.3, “How to Disable Parallel Replication Safely” and Section 4.1.2, “Enabling Parallel Apply During Install” for more information.

**Note**

The examples in this section are based on three clusters named ‘nyc’, ‘london’ and ‘tokyo’

Each cluster has two dedicated connectors on separate hosts.

The converted cluster will consist of a Composite Service named ‘global’ and the ‘nyc’ cluster will be the PRIMARY cluster, with ‘london’ and ‘tokyo’ as RELAY clusters.

If you do not have exactly three clusters, please adjust this procedure to match your environment.

Examples of before and after tungsten.ini files can be downloaded here:

- Multi-Site/Multi-Master Config [Before]
4.4.5.1. Conversion Prerequisites

**Warning**

Parallel apply MUST be disabled before starting your upgrade. You may re-enable it once the upgrade has been fully completed. See Section 4.1.5.3, “How to Disable Parallel Replication Safely” and Section 4.1.2, “Enabling Parallel Apply During Install” for more information.

- Obtain the latest v6 Tungsten Clustering software build and place it within `/opt/continuent/software`
- Extract the package
- The examples below refer to the `tungsten_prep_upgrade` script, this can be located in the extracted software package within the `tools` directory.

4.4.5.2. Step 1: Backups

Take a full and complete backup of one node - this can be a slave, and preferably should be either performed by:

- Percona xtrabackup whilst database is open
- Manual backup of all datafiles after stopping the database instance

4.4.5.3. Step 2: Redirect Client Connections

A big difference between Multi-Site/Multi-Master (MSMM) and Composite Master-Slave (CMS) is that with MSMM, clients can write into all clusters. With CMS clients only write into a single cluster.

To be able to complete this conversion process with minimal interruption and risk, it is essential that clients are redirected and only able to write into a single cluster. This cluster will become the PRIMARY cluster after the conversion. For the purpose of this procedure, we will use the ‘nyc’ cluster for this role.

After redirecting your client applications to connect through the connectors associated with the ‘nyc’ cluster, stop the connectors associated with the remaining clusters as an extra safeguard against writes happening.

On every connector node associated with london and tokyo:

```
shell> connector stop
```

4.4.5.4. Step 3: Enter Maintenance Mode

Enable Maintenance mode on all clusters using the `cctrl` command:

```
shell> cctrl
  cctrl> set policy maintenance
```

4.4.5.5. Step 4: Stop the Cross-site Replicators

**Important**

Typically the cross-site replicators will be installed within `/opt/replicator`, if you have installed this in a different location you will need to pass this to the script in the examples using the `--path` option

1. The following commands tell the replicators to go offline at a specific point, in this case when they receive an explicit heartbeat. This is to ensure that all the replicators stop at the same sequence number and binary log position. The replicators will NOT be offline until the explicit heartbeat has been issued a bit later in this step.

- On every nyc node:

  ```
  shell> /tungsten_prep_upgrade -o
  --clear-ranges
  shell> /tungsten_prep_upgrade --service london --offline
  shell> /tungsten_prep_upgrade --service tokyo --offline
  ```
• On every London node:

```bash
shell> ./tungsten_prep_upgrade -o
```

```bash
shell> ./tungsten_prep_upgrade --service nyc --offline
```

```bash
shell> ./tungsten_prep_upgrade --service tokyo --offline
```

• On every Tokyo node:

```bash
shell> ./tungsten_prep_upgrade -o
```

```bash
shell> ./tungsten_prep_upgrade --service london --offline
```

```bash
shell> ./tungsten_prep_upgrade --service tokyo --offline
```

2. Next, on the master hosts within each cluster we issue the heartbeat, execute the following using the cluster-specific `trepctl`, typically in `/opt/continuent`:

```bash
shell> trepctl heartbeat -name offline_for_upg
```

Ensure that every cross-site replicator on every node is now in the `OFFLINE:NORMAL` state:

```bash
shell> mmtrepctl status
```

```bash
shell> mmtrepctl --service {servicename} status
```

3. Capture the position of the cross-site replicators on all nodes in all clusters.

The service name provided should be the name of the remote service(s) for this cluster, so for example in the London cluster you get the positions for NYC and Tokyo, and in NYC you get the position for London and Tokyo, etc.

• On every London node:

```bash
shell> ./tungsten_prep_upgrade -g
```

```bash
shell> ./tungsten_prep_upgrade --service nyc --get
```

(NOTE: saves to ~/position-nyc-YYYYMMDDHHMMSS.txt)

```bash
shell> ./tungsten_prep_upgrade --service tokyo --get
```

(NOTE: saves to ~/position-tokyo-YYYYMMDDHHMMSS.txt)

• On every NYC node:

```bash
shell> ./tungsten_prep_upgrade -g
```

```bash
shell> ./tungsten_prep_upgrade --service london --get
```

(NOTE: saves to ~/position-london-YYYYMMDDHHMMSS.txt)

```bash
shell> ./tungsten_prep_upgrade --service tokyo --get
```

(NOTE: saves to ~/position-tokyo-YYYYMMDDHHMMSS.txt)

• On every Tokyo node:

```bash
shell> ./tungsten_prep_upgrade -g
```

```bash
shell> ./tungsten_prep_upgrade --service london --get
```

(NOTE: saves to ~/position-london-YYYYMMDDHHMMSS.txt)

```bash
shell> ./tungsten_prep_upgrade --service nyc --get
```

(NOTE: saves to ~/position-nyc-YYYYMMDDHHMMSS.txt)

4. Finally, to complete this step, stop the cross-site replicators on all nodes:

```bash
shell> ./tungsten_prep_upgrade --stop
```

### 4.4.5.6. Step 5: Export the tracking schema databases

On every node in each intended RELAY cluster (London and Tokyo), export the tracking schema associated the intended PRIMARY cluster (NYC).

Note the generated dump file is called `tungsten_global.dmp`. Global refers to the name of the intended Composite Cluster service, if you choose a different service name, change this accordingly.

• On every London node:

```bash
shell> mysqldump --opt --single-transaction tungsten_nyc > ~/tungsten_global.dmp
```

• On every Tokyo node:

```bash
shell> mysqldump --opt --single-transaction tungsten_nyc > ~/tungsten_global.dmp
```
4.4.5.7. Step 6: Uninstall the Cross-site Replicators

To uninstall the cross-site replicators, execute the following on every node:

```
shell> cd (replicator software path)
shell> tools/tpm uninstall --i-am-sure
```

4.4.5.8. Step 7: Create Composite Tracking Schema

In this step, we pre-create the database for the composite service tracking schema, we are using global as the service name in this example, if you choose a different Composite service name, adjust this accordingly

On every node in all clusters:

```
shell> mysql -e 'set session sql_log_bin=0; create database tungsten_global'
```

4.4.5.9. Step 8: Reload the tracking schema for RELAY clusters

This step reloads the tracking schema associated with the intended PRIMARY cluster (nyc) into the tracking schema we created in the previous step. This should ONLY be carried out within the intended RELAY clusters at this stage.

We DO NOT want the reloading of this schema to appear in the binary logs on the Master, therefore the reload needs to be performed on each node individually:

- On every london node:
  
  ```
  shell> mysql -e 'set session sql_log_bin=0; use tungsten_global; source ~/tungsten_global.dmp;
  ```

- On every tokyo node:
  
  ```
  shell> mysql -e 'set session sql_log_bin=0; use tungsten_global; source ~/tungsten_global.dmp;
  ```

4.4.5.10. Step 9: Stop local cluster Replicators

On every node in every cluster:

```
shell> replicator stop
```

**Warning**

The effect of this step will now mean that only the MASTER node in the PRIMARY cluster will be up to date with ongoing data changes. You must ensure that your applications handle this accordingly until the replicators are restarted at Step 14

4.4.5.11. Step 10: Remove THL

**Warning**

This step, if not followed correctly, could be destructive to the entire conversion. It is CRITICAL that this step is NOT performed on the intended PRIMARY cluster (nyc)

By default, THL files will be located within `/opt/continuent/thl`, if you have configured this in a different location you will need to adjust the path below accordingly

- On every london node:
  
  ```
  shell> cd /opt/continuent/thl
  shell> rm */thl*
  ```

- On every tokyo node:
  
  ```
  shell> cd /opt/continuent/thl
  shell> rm */thl*
  ```

4.4.5.12. Step 11: Export the tracking schema database on PRIMARY cluster

On every node within the intended PRIMARY cluster (nyc), export the tracking schema associated with the local service

Note the generated dump file is called `tungsten_global.dmp`, global refers to the name of the intended Composite Cluster service, if you choose a different service name, change this accordingly.
• On every nyc node:

```
shell> mysqldump --opt --single-transaction tungsten_nyc > ~/tungsten_global.dmp
```

4.4.5.13. Step 12: Reload the tracking schema for PRIMARY cluster

This step reloads the tracking schema associated with the intended PRIMARY cluster (nyc) into the tracking schema we created in the earlier step.

We DO NOT want the reloading of this schema to appear in the binary logs on the Master, therefore the reload needs to be performed on each node individually:

• On every nyc node:

```
shell> mysql -e 'set session sql_log_bin=0; use tungsten_global; source ~/tungsten_global.dmp;'
```


Update `/etc/tungsten/tungsten.ini` to a valid Composite Master-Slave config. An example of a valid config is as follows, a sample can also be downloaded from Section 4.4.5.1, “Conversion Prerequisites” above:

**Important**

Within a Composite Master-Slave topology, the ini file must be identical on EVERY node, including Connector Nodes

```
[defaults]
user=tungsten
home-directory=/opt/continuent
application-user=app_user
application-password=secret
application-port=3306
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=secret
mysql-allow-intensive-checks=true
skip-validation-check=THLSchemaChangeCheck

[nyc]
topology=clustered
master=db1
slaves=db2,db3
connectors=nyc-conn1,nyc-conn2

[london]
topology=clustered
master=db4
slaves=db5,db6
connectors=ldn-conn1,ldn-conn2b6
relay-source=nyc

[tokyo]
topology=clustered
master=db7
slaves=db8,db9
connectors=tky-conn1,tky-conn2
relay-source=nyc

[global]
composite-datasources=nyc,london,tokyo
```

4.4.5.15. Step 14: Install the Software on PRIMARY Cluster

Validate and install the new release on all nodes in the PRIMARY (nyc) cluster only:

```
shell> cd /opt/continuent/software/tungsten-clustering-6.1.3-43
shell> tools/tpm validate-update
```

If validation shows no errors, run the install:

```
shell> tools/tpm update --replace-release
```

4.4.5.16. Step 15: Start Local Replicators on PRIMARY cluster

After the installation is complete on all nodes in the PRIMARY cluster, restart the replicator services:

```
shell> replicator start
```
After restarting, check the status of the replicator using the `trepctl` and check that all replicators are ONLINE:

```
shell> trepctl status
```

### 4.4.5.17. Step 16: Install the Software on remaining Clusters

Validate and install the new release on all nodes in the remaining RELAY clusters (london and tokyo):

**Important**

The update should be performed on the MASTER nodes within each cluster first, validation will report and error that the roles conflict (MASTER vs RELAY). This is expected and to override this warning the `-f` options should be used on the MASTER nodes only.

```
shell> cd /opt/continuent/software/tungsten-clustering-6.1.3-43
shell> tools/tpm validate-update
```

If validation shows no errors, run the install:

```
On Master Nodes:
shell> tools/tpm update --replace-release -f
On Slave Nodes:
shell> tools/tpm update --replace-release
```

### 4.4.5.18. Step 17: Start Local Replicators on remaining clusters

After the installation is complete on all nodes in the PRIMARY cluster, restart the replicator services:

```
shell> replicator start
```

After restarting, check the status of the replicator using the `trepctl` and check that all replicators are ONLINE:

```
shell> trepctl status
```

### 4.4.5.19. Step 18: Convert Datasource roles for RELAY clusters

Following the upgrades, there are a number of “clean-up” steps that we need to perform within cctrl to ensure the datasource roles have been converted from the previous “master” roles to “relay” roles.

The following steps can be performed in a single cctrl session initiated from any node within any cluster.

```
shell> cctrl
Connect to PRIMARY cluster
cctrl> use nyc
Check Status and verify all nodes online
cctrl> ls
Connect to COMPOSITE service
cctrl> use global
Place PRIMARY service online
cctrl> datasource nyc online
Connect to london RELAY service
cctrl> use london
Convert old master to relay
cctrl> set force true
cctrl> datasource oldmasterhost offline
cctrl> datasource oldmasterhost relay
Repeat on tokyo RELAY service
cctrl> use tokyo
cctrl> set force true
cctrl> datasource oldmasterhost offline
cctrl> datasource oldmasterhost relay
Connect to COMPOSITE service
cctrl> use global
Place RELAY services online
cctrl> datasource london online
cctrl> datasource tokyo online
```

157
Place all clusters into AUTOMATIC

ctrl-k- set policy automatic

4.4.5.20. Step 19: Upgrade the Software on Connectors

Validate and install the new release on all connectors nodes:

```shell
- cd /opt/continuent/software/tungsten-clustering-6.1.3-43
- tools/tpm validate-update
```

If validation shows no errors, run the install:

```shell
- tools/tpm update --replace-release
```

After upgrading previously stopped connectors, you will need to restart the process:

```shell
- connector restart
```

**Warning**

Upgrading a running connector will initiate a restart of the connector services, this will result in any active connections being terminated, therefore care should be taken with this process and client redirection should be handled accordingly prior to any connector upgrade/restart

4.4.6. Installing an Upgraded JAR Patch

**Warning**

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

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

1. Determine your staging directory or untarred installation directory:

   ```shell
   - tpm query staging
   ```

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

   ```shell
   - cd tungsten-clustering-6.1.3-43
   ```

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
   ```

4.4.7. Installing Patches

**Warning**

This procedure should only be followed with the advice and guidance of a Continuent Support Engineer.

There are two ways we can patch the running environment, and the method chosen will depend on the severity of the patch and whether or not your use case would allow for a maintenance window

- Upgrade using a full software update following the standard upgrade procedures
• Use the patch command to patch just the files necessary

From time to time, Continuent may provide you with a patch to apply as a quicker way to fix small issues. Patched software will always be provided in a subsequent release so the manual patch method described here should only be used as a temporary measure to patch a live installation when a full software update may not immediately be possible

You will have been supplied with a file containing the patch, for the purpose of this example we will assume the file you have been given is called `undeployallnostop.patch`

1. Place cluster into `maintenance` mode
2. On each node of your installation:
   a. Copy the supplied patch file to the host
   b. From the installed directory (Typically this would be `/opt/continuent`) issue the following:

```
shell> cd /opt/continuent/tungsten
shell> patch -p1 -t undeployallnostop.patch
```
3. Return cluster to `automatic` mode

**Warning**

If a tpm update `--replace-release` is issued from the original software staging directory, the manual patch applied above will be over-written and removed.

The manual patch method is a temporary approach to patching a running environment, but is not a total replacement for a proper upgrade.

Following a manual patch, you *MUST* plan to upgrade the staged software to avoid reverting to an unpatched system.

If in doubt, always check with a Continuent Support Engineer.

### 4.5. 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 4.5.1, “Removing a Datasource from an Existing Deployment”
- Section 4.5.3, “Removing a Connector from an Existing Deployment”

#### 4.5.1. 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] |
```
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
or make the cluster unavailable.
Do you want to continue? (y/n) > y
```

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 dataset, 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`:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
```
The staging USER is tungsten
The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1
The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.8.5-41

Run the tpm command to update the software with the Staging-based configuration:

```
shell> ./tools/tpm configure alpha \
   --connectors=host1,host2,host3,host4 \
   --members=host1,host2,host3
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

- If you are using the INI file method with tpm:
  - 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 Tungsten Clustering 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 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
     ```

### 4.5.2. Removing a Composite Datasource/Cluster from an Existing Deployment

To remove an entire composite datasource (cluster) 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 cluster west from a composite datasource:

1. Check the current service state:

   ```
   shell> cctrl -multi
   [LOGICAL] / > ls
   # DATA SERVICES:                                             #
   |                                                          | function
   +----------------------------------------------------------------------------+
   | east                                                        |
   |east (composite master:ONLINE)                                 |
   | STATUS [OK] [2017/05/16 01:25:30 PM UTC]                    |
   +----------------------------------------------------------------------------+
   | west                                                        |
   |west                                                        |
   +----------------------------------------------------------------------------+
   | [LOGICAL] / > use global                                      |
   |[LOGICAL] /global > ls                                       |
   |COORDINATOR[db1:AUTOMATIC:ONLINE]                             |
   | DATASOURCES:                                                 |
   |                                                          | function
   +----------------------------------------------------------------------------+
   | east (composite master:ONLINE)                               |
   | [STATUS [OK] [2017/05/16 01:25:31 PM UTC]                  |
   +----------------------------------------------------------------------------+
   ```
2. Switch to **MAINTENANCE** policy mode:

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

3. Remove the composite member cluster from the composite service using the **drop** command.

```
[LOGICAL] /global > drop composite datasource west
COMPOSITE DATA SOURCE 'west@global' WAS DROPPED
```

```
[LOGICAL] /global > ls
COORDINATOR[db1:AUTOMATIC:ONLINE]
DATASOURCES:
+----------------------------------------------------------------------------+
|east(composite master:ONLINE)                                               |
|STATUS [OK] [2017/05/16 01:25:31 PM UTC]                                    |
+----------------------------------------------------------------------------+
```

```
[LOGICAL] /global > cd /
[LOGICAL] / > ls
+----------------------------------------------------------------------------+
|DATA SERVICES:                                                              |
|east                                                                         |
|global                                                                     |
```

4. If the removed composite datasource still appears in the top-level listing, then you will need to clean up by hand. For example:

```
[LOGICAL] /global > cd /
[LOGICAL] / > ls
+----------------------------------------------------------------------------+
|DATA SERVICES:                                                              |
|east                                                                         |
|global                                                                     |
|west                                                                        |
```

Stop all managers on all nodes at the same time

```
[LOGICAL] /global > use west
[LOGICAL] /west > manager * stop
```

```
shell> vtn $CONTINUENT_HOME/cluster-home/conf/dataservices.properties

Before:
east=db1,db2,db3
west=db4,db5,db6

After:
east=db1,db2,db3
```

Start all managers one-by-one, starting with the current master

```
shell> manager start
```

Once all managers are running, check the list again:

```
shell> cctrl -multi
[LOGICAL] / > ls
+----------------------------------------------------------------------------+
|DATA SERVICES:                                                              |
|east                                                                         |
|global                                                                     |
```

5. Switch to **AUTOMATIC** policy mode:

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

Now the cluster has been removed from the composite dataservice, the services on the old nodes must be stopped and then removed from the configuration.
1. Stop the running services on all nodes in the removed cluster:

```
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/tungsten-clustering-6.1.3-43
shell> cd /home/tungsten/tungsten-clustering-6.1.3-43
```

  b. Update the configuration, omitting the cluster datasource name from the list of members of the dataservice:

```
shell> tpm update global --composite-datasources=east
```

- 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 Tungsten Clustering 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
```

### 4.5.3. 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/tungsten-clustering-6.1.3-43
shell> cd /home/tungsten/tungsten-clustering-6.1.3-43
```

  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:
  ```bash
  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 Tungsten Clustering installation entirely:

   • Remove the startup scripts from your server:
     ```bash
     sudo /opt/continuent/tungsten/cluster-home/bin/undeployall
     ```

   • Remove the installation directory:
     ```bash
     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:
     ```bash
     rm /etc/init.d/tconnector
     ```
Chapter 5. Operations Guide

Tungsten Clustering™ 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.

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`. See Section 5.2, “Establishing the Shell Environment” for more information.

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

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
```

5.1. The Tungsten Clustering Home Directory

After installing Tungsten Clustering the home directory will be filled with a set of new directories. The home directory is specified by `--home-directory` or `--install-directory`. If you have multiple installations on a single server; each directory will include the same entries.

- **tungsten** - A symlink to the most recent version of the software. The symlink points into the `releases` directory. You should always use the symlink to ensure the most recent configuration and software is used.
- **releases** - Storage for the current and previous versions of the software. During an upgrade the new software will be copied into this directory and the `tungsten` symlink will be updated. See Section D.1.2, “The releases Directory” for more information.
- **service_logs** - Includes symlinks to the primary log for the replicator, manager and connector. This directory also includes logs for other tools distributed for Tungsten Clustering.
- **backups** - Storage for backup files created through `trepctl` or `cctrl`. See Section D.1.1, “The backups Directory” for more information.
- **thl** - Storage for THL files created by the replicator. Each replication service gets a dedicated sub-directory for storing THL files. See Section D.1.5, “The thl Directory” for more information.
- **relay** - Temporary storage for downloaded MySQL binary logs before they are converted into THL files.
- **share** - Storage for files that must persist between different software versions. The `env.sh` script will setup your shell environment to allow easy access to Tungsten Clustering tools.

5.2. Establishing the Shell Environment

The tools required to operate Tungsten Clustering are located in many directories around the home directory. The best way to access them is by setting up your shell environment.

The `env.sh` file will automatically be included if you specify the `--profile-script` during installation. This option may be included during a configuration change with `tpm update`.

If the `env.sh` file hasn't been included you may do so by hand with source.

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

**Important**

Special consideration must be taken if you have multiple installations on a single server. That applies for clustering and replication or multiple replicators.

Include the `--executable-prefix` and `--profile-script` options in your configuration. Instead of extending the PATH variable; the `env.sh` script will define aliases for each command. If you specified `--executable-prefix=mm` the `trepctl` command would be accessed as `mm_trepctl`.
5.3. 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
Tungsten Clustering (for MySQL) 6.1.3 build 43
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[10393](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)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|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 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 5.4, “Policy Modes”.
  
  - **STATUS**
    
    The current status of the coordinator host.
Operations Guide

- **RULES**

  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:

  ```
  +----------------------------------------------------------------------------+
  | [host1(master:ONLINE, progress=3, THL latency=0.561)]                     |
  | STATUS [OK] [2013/05/03 09:11:10 PM BST]                                  |
  +----------------------------------------------------------------------------+
  ```

  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 5.3.3, “Understanding Datasource Roles”. For information on datasource states, see Section 5.3.4, “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.

  - **MANAGER(state=ONLINE)**
  - 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.

    ```
    [host1(master:ONLINE, progress=3, THL latency=0.561)]
    ```

  - **REPLICATOR(role=slave, master=host1, state=ONLINE)**
  - 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.

    ```
    [host1(master:ONLINE, progress=3, THL latency=0.561)]
    ```

  - **DATASERVER(state=ONLINE)**
  - The status of the dataserver service, which indicates the status of the underlying database service.

    ```
    [host1(master:ONLINE, progress=3, THL latency=0.561)]
    ```

  - **CONNECTIONS(created=0, active=0)**
  - The Tungsten Connector service, showing the number of connections have been created on this service, and the number that are currently active.

  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**.

### 5.3.1. Latency or Relative Latency Display

Tungsten Clustering 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:

  ```
  mysql> show slave status
  *************************** 1. row ***************************
  | Seconds_Behind_Master | 0 |
  1 row in set (0.01 sec)

  mysql> show slave status
  *************************** 1. row ***************************
  | Seconds_Behind_Master | 7 |
  1 row in set (0.01 sec)

  mysql> show slave status
  *************************** 1. row ***************************
  | 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:

  ```
  [LOGICAL] /alpha > ls
  
  COORDINATOR[host1:AUTOMATIC:ONLINE]
  
  ROUTERS:
  
  | connector@host1[6189](ONLINE, created=1, active=0) |
  | connector@host2[1425](ONLINE, created=3, active=2) |
  | connector@host3[2419](ONLINE, created=1, active=0) |
  
  DATASOURCES:
  
  | host1(master:ONLINE, progress=5, 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:

```
[LOGICAL] /alpha > cluster heartbeat
HEARTBEAT 'DEFAULT' INSERTED
[LOGICAL] /alpha > Exiting...
```

### 5.3.2. 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:

```
[LOGICAL:EXPERT] /alpha > ls -l host1
COORDINATOR[host1:AUTOMATIC:ONLINE]

RUTERS:
+----------------------------------------------------------------------------+
|connector@host1[18348](ONLINE, created=403, active=8)                      |
|  host1(master:ONLINE, created=195, active=0)                               |
|  host2(slave:ONLINE, created=6, active=0, latency=146.998)                |
|  host3(slave:ONLINE, created=208, active=0, latency=31.000)               |
|  gateway:host2                                                            |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|host1(master:ONLINE, progress=154146, THL latency=0.390)                    |
|                  activeConnectionsCount:    0                                 |
|                  alertMessage:                                             |
|                  alertStatus:    OK                                       |
|                  alertTime:    1368209428766                                |
|                  appliedLatency:    0.0                                    |
|                  callableStatementsCreatedCount:    0                        |
|                  connectionsCreatedCount:    195                           |

```

5.3.2. 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:

```
[LOGICAL:EXPERT] /alpha > ls -l host1
COORDINATOR[host1:AUTOMATIC:ONLINE]

RUTERS:
+----------------------------------------------------------------------------+
|connector@host1[18348](ONLINE, created=403, active=8)                      |
|  host1(master:ONLINE, created=195, active=0)                               |
|  host2(slave:ONLINE, created=6, active=0, latency=146.998)                |
|  host3(slave:ONLINE, created=208, active=0, latency=31.000)               |
|  gateway:host2                                                            |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|host1(master:ONLINE, progress=154146, THL latency=0.390)                    |
|                  activeConnectionsCount:    0                                 |
|                  alertMessage:                                             |
|                  alertStatus:    OK                                       |
|                  alertTime:    1368209428766                                |
|                  appliedLatency:    0.0                                    |
|                  callableStatementsCreatedCount:    0                        |
|                  connectionsCreatedCount:    195                           |
```
5.3.3. Understanding Datasource Roles

All datasources within a datasource have a specific role within the datasource. 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</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Slave</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
More detailed information for each role:

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

- **standby**
  
  A standby 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 datasource can be enabled as a standard slave and included in load-balanced operations.

- **archive**
  
  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.

5.3.4. Understanding Datasource States

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

### 5.3.4.1. ONLINE State

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

### 5.3.4.2. 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] |
```

### 5.3.4.3. 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 the maintenance of manual policy mode, the datasource will remain in the OFFLINE state until the datasource is explicitly switched to the ONLINE state.

### 5.3.4.4. 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:

```
```

---

171
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.

### 5.3.5. 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.

#### 5.3.5.1. 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
Data Source 'host3' set to SHUNNED
```

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)
```

**Important**

Shunning a datasource does not stop the replicator; replication will continue on a shunned datasource until the replication 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 5.3.5.2, “Recover a Datasource”.

#### 5.3.5.2. 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 into 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.

<table>
<thead>
<tr>
<th>LOGICAL:EXPERT</th>
<th>/alpha &gt; datasource host3 recover</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'host3'</td>
<td></td>
</tr>
<tr>
<td>DATA SERVER 'host3' IS NOW AVAILABLE FOR CONNECTIONS</td>
<td></td>
</tr>
<tr>
<td>RECOVERING 'host3@alpha' TO A SLAVE USING 'host1@alpha' AS THE MASTER</td>
<td></td>
</tr>
<tr>
<td>DataSource 'host3' is now OFFLINE</td>
<td></td>
</tr>
<tr>
<td>RECOVERY OF 'host3@alpha' WAS SUCCESSFUL</td>
<td></td>
</tr>
</tbody>
</table>

During the recovery process, the node will be checked, replication reconfigured, and the node brought back into active service. If this process fails because the databases and replication states are out of sync and cannot be recovered, Tungsten Clustering may advise that a backup of another datasource and recovery to this datasource is performed. For more information on restoring from backups, see Section 5.11, "Restoring a Backup".

### 5.3.5.3. 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:

<table>
<thead>
<tr>
<th>LOGICAL:EXPERT</th>
<th>/alpha &gt; datasource host3 offline</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSource 'host3@alpha' is now OFFLINE</td>
<td></td>
</tr>
</tbody>
</table>

If the datasource 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 datasource must be in **MAINTENANCE** or **MANUAL** policy modes.

### 5.3.5.4. Mark a Datasource as Standby

**standby** datasources receive 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**:

<table>
<thead>
<tr>
<th>LOGICAL:EXPERT</th>
<th>/alpha &gt; datasource host3 standby</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource 'host3' now has role 'standby'</td>
<td></td>
</tr>
</tbody>
</table>

To clear the **standby** state:

<table>
<thead>
<tr>
<th>LOGICAL:EXPERT</th>
<th>/alpha &gt; datasource host3 clear standby</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource 'host3' now has role 'slave'</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

When a slave goes into standby mode, it will finish running any SQL queries that were started before it went into standby mode. New queries, even on the same connection, will not be directed to a slave that has just gone into standby.

### 5.3.5.5. Mark a Datasource as Archive

An **archive** datasource receives replication data and is included as part of the load-balancing provided by Tungsten Connector. 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.

### 5.3.6. 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** [173]

  The **ok** [173] status indicates that the datasource is currently operating correctly.
• **DIMINISHED** [174]

A **DIMINISHED** [174] 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=105, active=0)                                        |
+----------------------------------------------------------------------------+
```

The underlying service fault should be fixed and the status rechecked.

If all the services are **ONLINE**, but one node is in the **DIMINISHED** [174] state, you should let the auto-recovery process complete. To do this:

1. Place the cluster into automatic mode:

   `[LOGICAL] /alpha > set policy automatic`

2. Set the status of the node in the **DIMINISHED** [174] state to **OFFLINE**:

   `[LOGICAL:EXPERT] /alpha > replicator host1 offline`

Automatic recovery will then recover the node for you.

### 5.3.7. 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 5.1. Sequence: Shunning a Datasource

```
cctrl> datasource host1 shun
```

```
Online

Shunned
```
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`.

### 5.4. 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>Automatic</th>
<th>Manual</th>
<th>Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitoring</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Fault Detection</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Failure Fencing</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Failure Recovery</td>
<td>Yes</td>
<td>No</td>
<td>No</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.

### 5.4.1. 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:

[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 5.3, “Checking Dataservice Status”.

175
5.5. 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 monitored by Tungsten Clustering, 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.

5.5.1. Automatic Master Failover

When the dataservice policy mode is AUTOMATIC, the dataservice will automatically failover 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]]                           |
+----------------------------------------------------------------------------+
```
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
```

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]

RUTERS:
|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 12:38:21 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 5.6.1.3, “Slave Datasource Extended Recovery”.

5.5.2. 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 'host2@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
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.

### 5.6. 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 dataservice 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 DATA SERVICE '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 DATA SERVICE 'alpha' SUCCEEDED
  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 5.6.1, “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 5.6.2, “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>Automatic</td>
</tr>
<tr>
<td>AUTOMATIC</td>
<td>Master Recovery</td>
<td></td>
<td>Section 5.6.2, “Recover a failed master”</td>
</tr>
<tr>
<td>AUTOMATIC</td>
<td>Slave Failure</td>
<td></td>
<td>Automatic</td>
</tr>
</tbody>
</table>

178
<table>
<thead>
<tr>
<th>Policy Mode</th>
<th>Scenario</th>
<th>Datasource State</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slave Recovery</td>
<td>Master Failure</td>
<td>master:FAILED(NODE 'host1' IS UNREACHABLE)]</td>
<td>Section 5.6.2.4, “Failing over a master”</td>
</tr>
<tr>
<td></td>
<td>Master Recovery</td>
<td>master:SHUNNED(FAILED-OVER-TO-host2)</td>
<td>Section 5.6.2.2, “Recover a shunned master”</td>
</tr>
<tr>
<td></td>
<td>Slave Failure</td>
<td>slave:FAILED(NODE 'host1' IS UNREACHABLE)</td>
<td>Automatically removed from service</td>
</tr>
<tr>
<td></td>
<td>Slave Recovery</td>
<td>slave:FAILED(NODE 'host1' IS UNREACHABLE)</td>
<td>Section 5.6.1, “Recover a failed slave”</td>
</tr>
<tr>
<td>MAINTENANCE</td>
<td>Master Failure</td>
<td>Use Section 5.6.2.4, “Failing over a master” to promote a different slave</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Master Recovery</td>
<td>Section 5.6.2.3, “Manually Failing over a Master in MAINTENANCE policy mode”</td>
<td></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 5.6.1, “Recover a failed slave”</td>
</tr>
<tr>
<td></td>
<td>No Master</td>
<td>slave:SHUNNED(SHUNNED)</td>
<td>Section 5.6.2.1, “Recover when there are no masters”</td>
</tr>
</tbody>
</table>

### 5.6.1. 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:

```
[LOGICAL:EXPERT] /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 'host1@alpha' AS THE MASTER
DataSource 'host2' is now OFFLINE
RECOVERY OF DATA SERVICE 'alpha' SUCCEEDED
RECOVERED 1 DATA SOURCES IN SERVICE 'alpha'
```

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
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
```

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)                                                   |
+----------------------------------------------------------------------------+
```

179
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.

### 5.6.1.1. 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:

   ```
   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_2013-11-21_09-31_52
   NOTE >> host2 >> Create mysqldump in
   /opt/continuent/backups/provision_mysqldump_2013-11-21_09-31_52/provision.sql.gz
   NOTE >> Load the mysqldump file
   NOTE >> Put the alpha replication service online
   NOTE >> Clear THL and relay logs for the alpha replication service
   ```

   The default backup service for the host will be used; `mysqldump` can be used by specifying the `--mysqldump` option.

   `tungsten_provision_slave` handles the cluster status, backup, restore, and repositioning of the replication stream so that restored slave is ready to start operating again.

   **Important**

   When using a Multisite/Multimaster topology the additional replicator must be put offline before restoring data and put online after completion.

   ```
   shell> mm_trepctl offline
   shell> tungsten_provision_slave --source=host2
   shell> mm_trepctl online
   shell> mm_trepctl status
   ```

   For more information on using `tungsten_provision_slave` see Section 8.34, "The `tungsten_provision_slave` Script".

### 5.6.1.2. 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
Data Source '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 dataservice must be manually switched to the online state:

```
[LOGICAL:EXPERT] /alpha > datasource host3 online
Setting server for data source 'host3' to READ-ONLY
```

```
<table>
<thead>
<tr>
<th>Variable_name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>read-only</td>
<td>ON</td>
</tr>
</tbody>
</table>
```

```
Data Source 'host3@alpha' is now ONLINE
```

180
5.6.1.3. 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[4099](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 `set 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:

```plaintext
[LOGICAL:EXPERT] /alpha > datasource host2 slave
datasource 'host2' now has role 'slave'
```

7. Recover the slave back to the dataservice:

```plaintext
[LOGICAL:EXPERT] /alpha > datasource host2 recover
data datasource 'host2' is now online
```

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 5.11, "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 5.11.3, "Restoring from Another Slave".

### 5.6.2. 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:

```plaintext
[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.

#### 5.6.2.1. 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` command, 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 datasources 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:

```plaintext
[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=8.565) |
| status [OK] [2013/11/04 04:39:28 GMT] |
```

182
Once a host has been chosen, call the recover master using command specifying the full servicename and hostname of the chosen data-source:

```
[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 due 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@alpha' IS NOW OFFLINE
DATA SOURCE 'host1@alpha' WAS FOUND PHYSICAL DATA SOURCE 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
FIND PHYSICAL DATA SOURCE 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, the 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:

```
[LOGICAL] /alpha > set master host3
```
5.6.2.2. 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:

```
[LOGICAL:EXPERT] /alpha > replicator host1 slave host2
Replicator 'host1' is now a slave of replicator 'host2'
```

The recovered master will added back to the dataservice as a slave.

5.6.2.3. 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 datasource 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:

```
[LOGICAL:EXPERT] /alpha > dataservice host1 shun
DataSource 'host1' set to SHUNNED
```

2. Shun the slave **host2** and set the replicator to the offline state:

```
[LOGICAL:EXPERT] /alpha > replicator host2 offline
Replicator 'host2' is now OFFLINE
```

3. Configure **host2** as the master within the replicator service:

```
[LOGICAL:EXPERT] /alpha > replicator host2 master
```

4. Set the replicator on **host2** online:

```
[LOGICAL:EXPERT] /alpha > replicator host2 online
```

5. Recover **host2** online and then set it online:

```
[LOGICAL:EXPERT] /alpha > dataservice host2 welcome
[LOGICAL:EXPERT] /alpha > dataservice host2 online
```

6. Switch the replicator to be in **slave** mode:

```
[LOGICAL:EXPERT] /alpha > replicator host1 slave host2
Replicator 'host1' is now a slave of replicator 'host2'
```

7. Switch the replicator online:

```
[LOGICAL:EXPERT] /alpha > replicator host1 online
Replicator 'host1' is now ONLINE
```

8. Switch the datasource role for **host1** to be in slave mode:

```
[LOGICAL:EXPERT] /alpha > dataservice 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:

```
[LOGICAL:EXPERT] /alpha > dataservice host1 recover
```
5.6.2.4. 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.
5.6.2.5. Split-Brain Discussion

A split-brain occurs when a cluster which normally has a single write master, has two write-able masters.

This means that some writes which should go to the “real” master are sent to a different node which was promoted to write master by mistake.

Once that happens, some writes exist on one master and not the other, creating two broken masters. Merging the two data sets is impossible, leading to a full restore, which is clearly NOT desirable.

We can say that a split-brain scenario is to be strongly avoided.

A situation like this is most often encountered when there is a network partition of some sort, especially with the nodes spread over multiple availability zones in a single region of a cloud deployment.

This would potentially result in all nodes being isolated, without a clear majority within the voting quorum.

A poorly-designed cluster could elect more than one master under these conditions, leading to the split-brain scenario.

Since a network partition would potentially result in all nodes being isolated without a clear majority within the voting quorum, the default action of a Tungsten Cluster is to SHUN all of the nodes.

Shunning ALL of the nodes means that no client traffic is being processed by any node, both reads and writes are blocked.

When this happens, it is up to a human administrator to select the proper master and recover the cluster.

For more information, please see Section 5.6.2, “Recover a failed master”.

5.7. Composite Cluster Switching, Failover and Recovery

Switching of a dataservice 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 a Relay. The master dataservice within a composite cluster can be forced to failover to the slave dataservice in the event the master dataservice is offline.

Switching the master dataservice 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 Tungsten Clustering, 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.
5.7.1. Composite Cluster Site Switch

Our example cluster has two sites, east and west. They are both members of composite cluster global. Site east has hosts db1, db2 and db3. Site west has hosts db4, db5 and db6.

Important

When working with composite clusters, you should use the -multi [284] option to cctrl. With this option enabled the prompt and information provided will be different. You can perform operations both on individual parts of the cluster, and on the entire composite cluster. This can be achieved by using the use COMPOSITESERVICE command.

Tab completion is also available within cctrl when using this mode.
Composite Slave Dataservice (DR) - west

[LOGICAL] /west > use west
[LOGICAL] /west > ls

COORDINATOR[db1: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(relay:ONLINE, progress=6, latency=5.050)                                |
|STATUS [OK] [2015/04/14 01:46:59 AM UTC]                                    |
+----------------------------------------------------------------------------+

Manually switch the composite master role to the other site:

[LOGICAL] / > use global
[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]                                    |
+----------------------------------------------------------------------------+

+----------------------------------------------------------------------------+
|west(composite master:ONLINE)                                              |
|STATUS [OK] [2015/04/14 01:45:47 AM UTC]                                    |
+----------------------------------------------------------------------------+

Composite Slave Dataservice (DR) - east
use east

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)  
  REPLICATOR(role=relay, master=db4, state=ONLINE)  
  DATASERVER(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)  
  REPLICATOR(role=slave, master=db1, state=ONLINE)  
  DATASERVER(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)  
  REPLICATOR(role=slave, master=db1, state=ONLINE)  
  DATASERVER(state=ONLINE)  
  CONNECTIONS(created=0, active=0)

db4(master:ONLINE, progress=3, THL latency=0.671)  
STATUS [OK] [2015/04/14 01:45:42 AM UTC]  
  MANAGER(state=ONLINE)  
  REPLICATOR(role=master, state=ONLINE)  
  DATASERVER(state=ONLINE)  
  CONNECTIONS(created=0, active=0)

db5(slave:ONLINE, progress=3, latency=1.581)  
STATUS [OK] [2015/04/14 01:45:48 AM UTC]  
  MANAGER(state=ONLINE)  
  REPLICATOR(role=slave, master=db4, state=ONLINE)  
  DATASERVER(state=ONLINE)  
  CONNECTIONS(created=0, active=0)

Composite Master Dataservice (Primary) - west

use west

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=8.671)  
STATUS [OK] [2015/04/14 01:45:42 AM UTC]  
  MANAGER(state=ONLINE)  
  REPLICATOR(role=master, state=ONLINE)  
  DATASERVER(state=ONLINE)  
  CONNECTIONS(created=0, active=0)

db5(slave:ONLINE, progress=3, latency=1.581)  
STATUS [OK] [2015/04/14 01:45:48 AM UTC]  
  MANAGER(state=ONLINE)  
  REPLICATOR(role=slave, master=db4, state=ONLINE)  
  DATASERVER(state=ONLINE)  
  CONNECTIONS(created=0, active=0)
5.7.2. 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. It will try to update the configuration of the east data service but will not fail if not successful.

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
Tungsten Clustering 6.1.3 build 43
west: session established
[LOGICAL] / > use global
[LOGICAL] /global > ls
COORDINATOR[db4:AUTOMATIC:ONLINE]
DATASOURCES:
```

Mark the `east` data service as failed to prevent further actions:

```
[LOGICAL] /global > datasource east fail
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
WARNING: UNABLE TO REACH PHYSICAL DATA SERVICE 'east' AT THIS TIME.
EXCEPTION: Unable to unable to continue with command because no manager is available in service 'east'.
CONTINUING WITH COMMAND
COMPOSITE DATA SOURCE 'east' IS NOW IN THE FAILED STATE
```

Issue the `failover` command to force the `west` dataservice to become the composite master:

```
[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 to unable to continue with command because no manager is available in service 'east'.
CONTINUING WITH COMMAND
```

|db6(slave:ONLINE, progress=3, latency=1.559) |
| STATUS [OK] [2015/04/14 01:45:47 AM UTC] |

| MANAGER(state=ONLINE) |
| REPLICATOR(role=slave, master=db4, state=ONLINE) |
| DATASERVER(state=ONLINE) |
| CONNECTIONS(created=0, active=0) |
5.7.3. 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 5.6.1.1, “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
Tungsten Clustering 6.1.3 build 43
west: session established
```
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 'db1@east' 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]
```
**5.7.4. Composite Cluster Relay Recovery**

If the Relay node in a Composite cluster should ever point to the incorrect Master node, you can perform the following procedure to re-point the replicator to the desired Master node.

For example, say we have a composite cluster `global`, with nodes `db1`, `db2` and `db3` in `east` and `db4`, `db5` and `db6` in `west`. `db1` is the Master and `db4` is the Relay.

In the output below, the Relay node `db4` shows that its replicator is using `db2` as the Master instead of `db1`:

```
Recovered Composite Slave Dataservice (DR) - east

[LOGICAL] /global > use east
[LOGICAL] /east > ls
COORDINATOR[db2:AUTOMATIC:ONLINE]
RUTERS:
+----------------------------------------------------------------------------+
| connector@db1.vagrant-erics[10051](ONLINE, created=0, active=0) |
| connector@db2.vagrant-erics[16111](ONLINE, created=0, active=0) |
| connector@db3.vagrant-erics[16636](ONLINE, created=0, active=0) |
| connector@db4.vagrant-erics[9757](ONLINE, created=1, active=0) |
| connector@db5.vagrant-erics[9781](ONLINE, created=1, active=0) |
| connector@db6.vagrant-erics[9944](ONLINE, created=1, active=0) |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
| db1(relay:ONLINE, progress=7, latency=0.000) |
| STATUS [OK] [2015/04/14 04:11:52 AM UTC] |
| MANAGER(state=ONLINE) |
| REPLICATOR(role=relay, master=db4, state=ONLINE) |
| DATASERVER(state=ONLINE) |
| CONNECTIONS(created=3, active=0) |
+----------------------------------------------------------------------------+
| db2(slave:ONLINE, progress=7, latency=6.000) |
| STATUS [OK] [2015/04/14 04:11:56 AM UTC] |
| MANAGER(state=ONLINE) |
| REPLICATOR(role=slave, master=db1, state=ONLINE) |
| DATASERVER(state=ONLINE) |
| CONNECTIONS(created=0, active=0) |
+----------------------------------------------------------------------------+
| db3(slave:ONLINE, progress=7, latency=11.000) |
| STATUS [OK] [2015/04/14 04:12:01 AM UTC] |
| MANAGER(state=ONLINE) |
| REPLICATOR(role=slave, master=db1, state=ONLINE) |
| DATASERVER(state=ONLINE) |
| CONNECTIONS(created=0, active=0) |
+----------------------------------------------------------------------------+

5.7.4. Composite Cluster Relay Recovery

Use the `cctrl replicator` command to adjust the relay source:
5.8. Multimaster Recovery

In a Multi-Master topology, a switch or failover not only promotes a slave to be a new master, but also will require the ability to reconfigure cross-site communications. This process therefore assumes that cross-site communication is online and working. In some situations, it may be possible that cross-site communication is down, or for some reason cross-site replication is in an OFFLINE:ERROR state - for example a DDL or DML statement that worked in the local cluster may have failed to apply in the remote.

If a switch or failover occurs and the process is unable to reconfigure the cross-site replicators, the local switch will still succeed, however the associated cross-site services will be placed into a "SHUNNED(SUBSERVICE-SWITCH-FAILED)" state.

The guide explains how to recover from this situation.

The examples are based on a 2-cluster topology, named NYC and LONDON and the composite dataservice named GLOBAL. The cluster is configured with the following dataservers:

- **NYC**: db1 [master], db2 [slave], db3 [slave]
- **LONDON**: db4 [master], db5 [slave], db6 [slave]

The cross site replicators in both clusters are in an OFFLINE:ERROR state due to failing DDL.

A switch was then issued, promoting db3 as the new master in NYC and db5 as the new master in LONDON.
In the above example, you can see that all services are in the SHUNNED(SUBSERVICE-SWITCH-FAILED) state, and partial reconfiguration has happened.

The Replicators for db4 and db6 should be slaves of db5, db5 has correctly configured to the new master in nyc, db3. The actual state of the cluster in each scenario may be different depending upon the cause of the loss of cross-site communication. Using the steps below, apply the necessary actions that relate to your own cluster state, if in any doubt always contact Continuent Support for assistance.

1. The first step is to ensure the initial replication errors have been resolved and that the replicators are in an online state, the steps to resolve the replicators will depend on the reason for the error.

2. From one node, connect into cctrl at the expert level
   ```
   shell> cctrl -multi -expert
   ```

3. Next, connect to the cross-site subservice, in this example, london_from_nyc
   ```
   / > use london_from_nyc
   london_from_nyc: session established, encryption=false, authentication=false
   ```

4. Next, place the service into Maintenance Mode
   ```
   /london_from_nyc > set policy maintenance
   ```

5. Enable override of commands issued
   ```
   /london_from_nyc > set force true
   ```

6. Bring the relay datasource online
   ```
   /london_from_nyc > datasource db5 online
   ```

7. If you need to change the source for the relay replicator to the correct, new, master in the remote cluster, take the replicator offline. If the relay source is correct, then move on to step 10
   ```
   /london_from_nyc > replicator db5 offline
   ```

8. Change the source of the relay replicator
   ```
   /london_from_nyc > replicator db5 relay nyc/db3
   ```

9. Bring the relay replicator online
   ```
   /london_from_nyc > replicator db5 online
   ```

10. For each datasource that requires the replicator altering, issue the following commands
    ```
    /london_from_nyc > replicator {datasource} offline
    /london_from_nyc > replicator {datasource} slave {relay}
    /london_from_nyc > replicator {datasource} online
    ```
    For example:
    ```
    /london_from_nyc > replicator db4 offline
    /london_from_nyc > replicator db4 slave db5
    /london_from_nyc > replicator db4 online
    ```

11. Once all replicators are using the correct source, we can then bring the cluster back
    ```
    /london_from_nyc > cluster welcome
    ```

12. Some of the datasources may still be in the SHUNNED state, so for each of those, you can then issue the following
    ```
    /london_from_nyc > datasource {datasource} online
    ```
    For example:
    ```
    /london_from_nyc > datasource db4 online
    ```
13. Once all nodes are online, we can then return the cluster to automatic

```
[LOGICAL:EXPERT] /london_from_nyc > set policy automatic
```

14. Repeat this process for the other cross-site subservice if required

## 5.9. Managing Transaction Failures

Inconsistencies between a master and slave dataserver can occur for a number of reasons, including:

- An update or insertion has occurred on the slave independently of the master. This situation can occur if updates are allowed on a slave that is acting as a read-only slave for scale out, or in the event of running management or administration scripts on the slave.
- A switch or failover operation has lead to inconsistencies. This can happen if client applications are still writing to the slave or master at the point of the switch.
- A database failure causes a database or table to become corrupted.

When a failure to apply transactions occurs, the problem must be resolved, either by skipping or ignoring the transaction, or fixing and updating the underlying database so that the transaction can be applied.

When a failure occurs, replication is stopped immediately at the first transaction that caused the problem, but it may not be the only transaction and this may require extensive examination of the pending transactions to determine what caused the original database failure and then to fix and address the error and restart replication.

### 5.9.1. 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** [174], and the replicator state as **SUSPECT**:

```
[LOGICAL] /alpha > ls
COORDINATOR[host3:AUTOMATIC:ONLINE]
...
+----------------------------------------------------------------------------+
|host2(slave:ONLINE, progress=-1, latency=-1.000)                        |
|STATUS [DIMINISHED] [2013/06/26 10:14:12 AM BST]                     |
|REASON [FAILED TO RECOVER REPLICATOR 'host2']                         |
+----------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                 |
| REPLICATOR(role=slave, master=host1, state=SUSPECT)                  |
| DATASERVER(state=ONLINE)                                              |
| CONNECTIONS(created=0, active=0)                                      |
+----------------------------------------------------------------------------+
...
```

More detailed information about the status and the statement that failed can be obtained within `cctrl` using the `replicator` command:

```
[LOGICAL] /alpha > repl replicator host2 status
replicator host2 status
 appliedLastEventId: NONE
 appliedLastSeqno: -1
 appliedLatency: -1.0
 channels: -1
 clansterName: firstcluster
 currentEventId: NONE
 currentTimeMillis: 1372238640236
 dataServerHost: host2
 extensions: latestEpochNumber: -1
 masterConnectorUri: thl://host1/
masterListenUri: thl://host2:2112/
 maxinumStoredSeqNo: -1
 mininumStoredSeqNo: -1
 offlineRequests: NONE
 pendingError: Event application failed: seqno=120 fragno=0
 message=java.sql.SQLException: Statement failed on slave but succeeded on master
 pendingErrorCode: NONE
 pendingErrorEventId: mysql-bin.000912:00000000000912967:0
```
The `trepsvc.log` log file will also contain the error information about the failed statement. For example:

```plaintext
... 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 - pool-2-thread-1] ERROR management.OpenReplicatorManager Received error notification, shutting down services :
INFO   | jvm 1    | 2013/06/26 10:14:12 | Event application failed: seqno=120 fragno=0 message=java.sql.SQLException: Statement failed on slave but succeeded on master
INFO   | jvm 1    | 2013/06/26 10:14:12 | insert into messages values (0,'Trial message','Jack','Jill',now())
INFO   | jvm 1    | 2013/06/26 10:14:12 | com.continuent.tungsten.replicator.applier.ApplierException: java.sql.SQLException: Statement failed on slave but succeeded on master
...```

Once the error or problem has been found, the exact nature of the error should be determined so that a resolution can be identified:

1. Identify the reason for the failure by examining the full error message. Common causes are:
   - Duplicate primary key

   A row or statement is being inserted or updated that already has the same insert ID or would generate the same insert ID for tables that have auto increment enabled. The insert ID can be identified from the output of the transaction using `thl`. Check the slave to identify the faulty row. To correct this problem you will either need to skip the transaction or delete the offending row from the slave dataserver.

   The error will normally be identified due to the following error message when viewing the current replicator status, for example:

   ```plaintext
   [LOGICAL] /alpha > replicator host3 status ...
   pendingErrorSeqno: 120
   pendingExceptionMessage: java.sql.SQLException: Statement failed on slave but succeeded on master
   insert into messages values (0,'Trial message','Jack','Jill',now())
   ...```

   The 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:

   ```plaintext
   shell> thl list -seqno 29
   SEQ# = 29 / FRAG# = 0 (last frag)
   - TIME = 2014-01-09 16:47:40.0
   - EVENTID = mysql-bin.080032:0800000000001872;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
   ...```
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 what the current value of the corresponding row:

```
mysql> select * from myent where id = 2;
+----+---------------+
<table>
<thead>
<tr>
<th>id</th>
<th>msg</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Other Message</td>
</tr>
</tbody>
</table>
+----+---------------+
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 5.9.2, “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 5.10, “Creating a Backup”, or Section 5.11, “Restoring a Backup”. To reprovision a slave from the master or another slave, see `tungsten_provision_slave`.

5.9.2. 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 5.9.1, “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`:

  ```bash
  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:

  ```bash
  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:

  ```bash
  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.

5.10. Creating a Backup

The `datasource 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.

Backing 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 71 Apr  4 16:09 storage.index
-rw-r--r-- 1 tungsten tungsten 133907646 Apr  4 16:09 store-0000000001-mysqldump_2013-04-04_16-08_42.sql.gz
-rw-r--r-- 1 tungsten tungsten 317 Apr  4 16:09 store-0000000001.properties
```
For information on managing backup files within your environment, see Section D.1.1, "The backups Directory".

The `storage.index` contains the backup file index information. The actual backup data is stored in the GZipped file. The properties of the backup file, including the tool used to create the backup, and the checksum information, are located in the corresponding `.properties` file. Note that each backup and property file is uniquely numbered so that it can be identified when restoring a specific backup.

A backup can also be initiated and run in the background by adding the & (ampersand) to the command:

```
[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
```

---

### 5.10.1. Using a Different Backup Tool

If `xtrabackup` is installed when the dataservice is first created, `xtrabackup` will be used as the default backup method. Four built-in backup methods are provided:

- **mysqldump** — SQL dump to a single file. This is the easiest backup method but it is not appropriate for large data sets.
- **xtrabackup** — Full backup. This will take longer to take the backup and to restore.
- **xtrabackup-full** — Full backup to a directory (this is the default if `xtrabackup` is available and the backup method is not explicitly stated).
- **xtrabackup-incremental** — Incremental backup from the last `xtrabackup-full` or `xtrabackup-incremental` backup.

The default backup tool can be changed, and different tools can be used explicitly when the backup command is executed. The Percona `xtrabackup` tool can be used to perform both full and incremental backups. Use of the this tool is optional and can be 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 `datasource backup` command within `cctrl`:

```
[LOGICAL:EXPERT] /alpha > datasource host2 backup xtrabackup-full
Backup of dataSource 'host2' succeeded; url=storage://file-system/store-0000000006.properties
```

---

### 5.10.2. Automating Backups

Backups cannot be automated within Tungsten Clustering, instead a cron job should be used to automate the backup process. `cluster_backup` is packaged with Tungsten Clustering 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 8.8, "The cluster_backup Command" for more information.

```
[shell] /opt/continuent/tungsten/cluster-home/bin/cluster_backup -v >> /opt/continuent/service_logs/cluster_backup.log
```

The command output will be appended to `/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.

An example cron entry:

```
[shell] crontab -l
00 00 * * * /opt/continuent/tungsten/cluster-home/bin/cluster_backup >> /opt/continuent/service_logs/cluster_backup.log 2>&1
```

All output will be appended to `/opt/continuent/service_logs/cluster_backup.log`.

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] echo "datasource host2 backup" | /opt/continuent/tungsten/tungsten-manager/bin/cctrl -expert
```

---

### 5.10.3. Using a Different Directory Location

The default backup location the `backups` directory of the Tungsten Clustering installation directory. For example, using the recommended installation location, backups are stored in `/opt/continuent/backups`.

See Section D.1.1.4, "Relocating Backup Storage" for details on changing the location where backups are stored.
5.10.4. Creating an External Backup

There are several considerations to take into account when you are using a tool other than Tungsten Clustering to take a backup. We have taken great care to build all of these into our tools. If the options provided do not meet your needs, take these factors into account when taking your own backup.

- How big is your data set?

  The mysqldump tool is easy to use but will be very slow once your data gets too large. We find this happens around 1GB. The xtrabackup tool works on large data sets but requires more expertise. Choose a backup mechanism that is right for your data set.

- Is all of your data in transaction-safe tables?

  If all of your data is transaction-safe then you will not need to do anything special. If not then you need to take care to lock tables as part of the backup. Both mysqldump and xtrabackup take care of this. If you are using other mechanisms you will need to look at stopping the replicator, stopping the database. If you are taking a backup of the master then you may need to stop all access to the database.

- Are you taking a backup of the master?

  The Tungsten Replicator stores information in a schema to indicate the restart position for replication. On the master there can be a slight lag between this position and the actual position of the master. This is because the database must write the logs to disk before Tungsten Replicator can read them and update the current position in the schema.

  When taking a backup from the master, you must track the actual binary log position of the master and start replication from that point after restoring it. See Section 5.11.2, “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
cmysqldump --opt --single-transaction --all-databases --add-drop-database --master-data=2
```

5.11. 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 5.6.1.1, “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:

```
LOGICAL: /alpha > datasource host2 shun
LOGICAL: /alpha > replicator host2 offline
```

2. Restore the backup using cctrl:

```
LOGICAL: /alpha > datasource host2 restore
```

By default, the restore process takes the latest backup available for the host being restored. Tungsten Clustering does not automatically locate the latest backup within the dataservice across all datasources.

5.11.1. Restoring a Specific Backup

To restore a specific backup, specify the location of the corresponding properties file using the format:

```
storage://storage-type/location
```

For example, to restore the backup from the filesystem using the information in the properties file store-0000000004.properties, login to the failed host:

1. 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.
5.11.2. Restoring an External Backup

If a backup has been performed outside of Tungsten Clustering, for example from filesystem snapshot or a backup performed outside of the dataservice, follow these steps:

1. 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 5.10.4, “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.0000000473863524;1 db1-east.continuent.com
     shell> exit
     ```

   - Return to the slave node and run `tungsten_set_position` to update the `trep_commit_seqno` table:

     ```
     shell> tungsten_set_position --service=alpha --source=host1 --seqno=32033678
     ```

5. 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.

5.11.3. 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 5.6.1.1, “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; url=storage://file-system/store-0008800009.properties
   ```

2. Copy the backup information from `host2` to `host3`. See Section D.1.1.3, “Copying Backup Files” for more information on copying backup information between hosts. If you are using `xtrabackup` there will be additional files needed before the next step. The example below uses `scp` to copy a `mysqldump` backup:
If you are using xtrabackup:

```bash
shell> cd /opt/continuent/backups/xtrabackup
shell> rsync -aze ssh full_xtrabackup_2014-08-16_15-44_86 host3:$PWD/
```

3. Shun the datasource to be restored, and put the replicator service offline using `cctrl`:

```bash
[LOGICAL] /alpha > datasource host2 shun
[LOGICAL] /alpha > replicator host2 offline
```

4. Restore the backup using `cctrl`:

```bash
[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.

### 5.11.4. Manually Recovering from Another Slave

In the event that a restore operation fails, or due to a significant failure in the dataserver, an alternative option is to seed the failed dataserver directly from an existing running slave.

For example, on the host `host2`, the data directory for MySQL has been corrupted, and `mysqld` will no longer start. This status can be seen from examining the MySQL error log in `/var/log/mysql/error.log`:

```
130520 14:37:08 [Note] Recovering after a crash using /var/log/mysql/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: 5506
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`:

```bash
[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`:

```bash
[LOGICAL] /alpha > datasource host3 shun
[LOGICAL] /alpha > replicator host3 offline
```

3. Stop the `mysqld` service on `host2`:

```bash
shell> sudo /etc/init.d/mysql stop
```

4. Stop the `mysqld` service on `host3`:

```bash
shell> sudo /etc/init.d/mysql stop
```

5. Delete the `mysqld` data directory on `host2`:

```bash
shell> sudo rm -rf /var/lib/mysql/*
```

6. If necessary, ensure the `tungsten` user can write to the MySQL directory:

```bash
shell> sudo chmod 777 /var/lib/mysql
```

7. Use `rsync` on `host3` to send the data files for MySQL to `host2`:
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:

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

9. Update the ownership and permissions on the data files on host2:

   ```
   host2 shell> sudo chown -R mysql:mysql /var/lib/mysql
   host2 shell> sudo chmod 770 /var/lib/mysql
   ```

10. Clear out the THL files on the target node host2 so the slave replicator service may start cleanly:

    ```
    host2 shell> thl purge
    ```

11. Recover host2 back to the dataservice:

    ```
    [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.

5.11.5. 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 pre-requisites, as described in Appendix B, 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 Tungsten Clustering 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 Tungsten Clustering installation, and change to the staging directory. To determine the staging directory, use:

      ```
      shell> tpm query staging
      ```

   b. Repeat the installation of the service on the host being brought back:

      ```
      shell> ./tools/tpm update svc_name --hosts=host3
      ```

      The update process will re-install Tungsten Clustering 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
      ```
b. Login in to the server used for staging your Tungsten Clustering 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 [459]` 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.

5.11.6. 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 Tungsten Clustering 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/alpha/*
   ```

   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 Tungsten Clustering services:

```
shell> startall
```

Now start the services using the same command on each of the remaining servers.

5.12. Migrating and Seeding Data

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

If you are migrating an existing MySQL native replication deployment to use Tungsten Clustering the configuration of the Tungsten Clustering replication must be updated to match the status of the slave.
1. Deploy Tungsten Clustering using the model or system appropriate according to Chapter 2, Deployment. Ensure that the Tungsten Clustering is not started automatically by excluding the --start or --start-and-report 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 | \n   grep ' 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 Tungsten Clustering 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 Tungsten Clustering automatically reconfiguring services:

   ```shell
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 | \n   grep ' Master_Host| Last_Error| Master_Log_File| Exec_Master_Log_Pos'
   Master_Host: tr-ssl1
   Master_Log_File: mysql-bin.000025
   Last_Error: Error executing row event: 'Table 'tungsten_alpha.heartbeat' doesn't exist'
   Exec_Master_Log_Pos: 181268
   ```

   If you have multiple 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 Tungsten Clustering 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.

   ```shell
trepctl online -from-event 000025:181268
   ```

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

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 Tungsten Clustering services:

```
shell> trepctl 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> trepctl online -from-event 000025:188249
shell> startall
```

10. Check that replication is operating correctly by using `trepctl status` on the master and each slave to confirm the correct position.

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

12. Put the cluster back into automatic mode:

```
cctrl> set policy automatic
```

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

14. 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, Tungsten Clustering should be operating as the replication service for your MySQL servers. Use the information in Chapter 5, Operations Guide to monitor and administer the service.

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

When running an existing MySQL native replication service that needs to be migrated to a Tungsten Clustering service, one solution is to create the new Tungsten Clustering 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 5.2, “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`.

**Figure 5.2. Migration: Migrating Native Replication using a New Service**

<table>
<thead>
<tr>
<th>Old host1</th>
<th>Service Beta</th>
<th>Old host2</th>
<th>host3</th>
<th>host4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataserver</td>
<td></td>
<td>Dataserver</td>
<td>Tungsten Replicator</td>
<td>Tungsten Replicator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Old host2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To configure the service:

1. Stop replication on a slave for the existing native replication installation:
Operations Guide

1. STOP SLAVE;

2. SHOW SLAVE STATUS;

   Master_Host: host3
   Master_Log_File: mysql-bin.000002
   Exec_Master_Log_Pos: 559

3. START SLAVE;

4. 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 option with mysqldump.

5. Restart the slave using native replication:

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

7. Configure a second replication service, beta to apply data using the existing MySQL native replication server as the master, and the master of newalpha.

   For more information, see Section 3.7, “Replicating Data Into an Existing Dataservice”. Do not start the new service.

8. 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/dsctl set 
   -seqno=0 -epoch=0 -service=beta 
   -source-id=host3 -event-id=mysql-bin.000002:559

9. 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.

5.13. Resetting a Tungsten Clustering Dataservice

Follow these steps to reset replication for an entire dataservice. The current master will remain the master. Use the switch after completion to change the master.

Warning

The procedures in this section are designed for the pre-v6.x Multisite/Multimaster topology ONLY. Do NOT use these procedures with version 6.x Multisite Clusters.

For version 6.x Multisite Clustering, please refer to Deploying Composite Multimaster Clustering.

See Section 5.11.6, “Resetting an Entire Dataservice from Filesystem Snapshots” if you would like to restore a file system snapshot to every server as part of this process.

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

   cctrl> set policy maintenance

2. Enable force mode:

   cctrl> set force true

3. Shun each datasource:
4. Put each Tungsten Connector offline:
```
cctrl> router * offline
```

5. On each datasource, reset the service:
```
shell> trepctl -service east offline
shell> trepctl -service east reset -all -y
```

6. Reconfigure the replicator and datasource configuration on each host, starting with the master:
```
cctrl> set force true
```
```
cctrl> replicator new-master master
```
```
cctrl> replicator new-master online
```
```
cctrl> datasource new-master master
```
```
cctrl> datasource new-master online
```
```
cctrl> replicator slave1 slave new-master
```
```
cctrl> replicator slave1 online
```
```
cctrl> datasource slave1 slave
```
```
cctrl> datasource slave1 online
```
```
cctrl> replicator slave2 slave new-master
```
```
cctrl> replicator slave2 online
```
```
cctrl> datasource slave2 slave
```
```
cctrl> datasource slave2 online
```

7. The connector can now be re-enabled and the cluster returned to operational state:
```
cctrl> router * online
```
```
cctrl> set policy automatic
```
```
cctrl> cluster heartbeat
```

Any servers not matching the master must be reprovisioned. Use the `tungsten_provision_slave` tool to reprovision from the master or valid slave server.

### 5.13.1. Reset a Single Site in a Multisite/Multimaster Topology

**Warning**
The procedures in this section are designed for the pre-v6.x Multisite/Multimaster topology ONLY. Do NOT use these procedures with version 6.x Multisite Clusters.

For version 6.x Multisite Clustering, please refer to Deploying Composite Multimaster Clustering.

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 dataservices in the `east` service will be reprovisioned from one of the dataservices 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 datasource into **MAINTENANCE** mode. This ensures that Tungsten Clustering will not attempt to automatically recover the service.
```
cctrl [east]> set policy maintenance
```

2. Put the Tungsten Connector for the datasource offline:
```
cctrl [east]> router * offline
```

3. Stop all services running in `east`:
```
shell east> /opt/continuent/tungsten/cluster-home/bin/stopall
shell east> /opt/replicator/tungsten/cluster-home/bin/stopall
```

4. Disable cross-site replication and reset the replication position:
```
shell west> /opt/replicator/tungsten/tungsten-replicator/bin/trepctl -service east offline
```
5. Reprovision the master node in `east`:

```bash
shell east(master)> /opt/continuent/tungsten/tungsten-replicator/scripts/tungsten_provision_slave --source west(slave)
```

6. Restart the services in `east`:

```bash
shell east> /opt/continuent/tungsten/cluster-home/bin/startall
shell east> /opt/replicator/tungsten/cluster-home/bin/startall
```

7. Ensure that the new service is functioning normally:

```bash
shell east> echo ls | /opt/continuent/tungsten/tungsten-manager/bin/cctrl
shell east> /opt/replicator/tungsten/tungsten-replicator/bin/trepctl status
```

8. Bring the remote replicators back ONLINE

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

Set the cluster to normal operational state:

```bash
cctrl> router * online
cctrl> set policy automatic
```

9. Reprovision the remaining nodes in the `east` cluster

```bash
shell east(slave1)> /opt/continuent/tungsten/tungsten-replicator/scripts/tungsten_provision_slave \
--source east(master)
shell east(slave1)> /opt/continuent/tungsten/cluster-home/bin/startall
shell east(slave1)> /opt/replicator/tungsten/cluster-home/bin/startall
shell east(slave2)> /opt/continuent/tungsten/tungsten-replicator/scripts/tungsten_provision_slave \
--source east(slave)
shell east(slave2)> /opt/continuent/tungsten/cluster-home/bin/startall
shell east(slave2)> /opt/replicator/tungsten/cluster-home/bin/startall
```

### 5.13.2. Reset All Sites in a multisite/multimaster topology

**Warning**

The procedures in this section are designed for the pre-v6.x Multisite/Multimaster topology ONLY. Do NOT use these procedures with version 6.x Multisite Clusters.

For version 6.x Multisite Clustering, please refer to Deploying Composite Multimaster Clustering.

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

1. Put the both dataservices into **MAINTENANCE** mode. This ensures that Tungsten Clustering will not attempt to automatically recover the service.

   ```bash
cctrl [east]> set policy maintenance
cctrl [west]> set policy maintenance
```

2. Put the Tungsten Connector for both dataservices offline:

   ```bash
cctrl [east]> router * offline
cctrl [west]> router * offline
```

3. Stop the services on each server in the east region `[east(1,2,3)]`

   ```bash
   shell east> /opt/continuent/tungsten/cluster-home/bin/stopall
   shell east> /opt/replicator/tungsten/cluster-home/bin/stopall
   ```

4. Stop the services on each server in the west region `[west(1,2,3)]`

   ```bash
   shell west> /opt/continuent/tungsten/cluster-home/bin/stopall
   shell west> /opt/replicator/tungsten/cluster-home/bin/stopall
   ```

5. Reset the cluster on `east(1,2,3)`:

   ```bash
   shell east> /opt/continuent/tungsten/tools/tpm reset
   ```
6. Reset the cluster on west{1,2,3}:

   shell west > /opt/continuent/tungsten/tools/tpm reset
   shell west > /opt/replicator/tungsten/tools/tpm reset

7. Reset the replication services on east{1,2,3}:

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

8. Reset the replication services on west{1,2,3}:

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

9. Restart the services on each server in the east region east{1,2,3}:

   shell east > /opt/continuent/tungsten/cluster-home/bin/startall

10. Restart the services on each server in the west region west{1,2,3}:

    shell west > /opt/continuent/tungsten/cluster-home/bin/startall

11. Place all the Tungsten Replicator services on east{1,2,3} back online:

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

12. Place all the Tungsten Replicator services on west{1,2,3} back online:

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

5.14. Replicator Fencing

Tungsten Clustering 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 repli-
cators. 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.

5.14.1. 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 config-
ures 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 iden-
tified, the node will be placed in the online state.

5.14.2. 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 5.5.1, “Automatic Master Failover”). 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/stops 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
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
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 must be explicitly recovered using the `recover` or `datasource host recover` commands.

### 5.15. Performing Database or OS Maintenance

When performing database or operating system maintenance, datasources should be temporarily removed from the dataservice and the replicator should be disabled. Follow these rules for the best results. Detailed steps are provided below for different scenarios.

- For maintenance operations on a master, the current master should be switched, the required maintenance steps performed, and then the master switched back.
- Disable a datasource using the `datasource shun` command.
- Put the replicator offline using `trepctl offline`.
- If you are using the Multisite/Multimaster topology, put the extra replicator offline using `mm_trepctl offline`. The `mm_trepctl` alias will only work if you configured Tungsten Replicator with the `--executable-prefix=mm` option.
- When making changes to a MySQL system the binary log should be disabled for your session. This will prevent corrective actions from replicating to other servers. Ignore this suggestion if you are making changes to a master that should be replicated.
  ```sql
  mysql> SET SESSION SQL_LOG_BIN=0;
  ```
- Restart replication and recover the datasource after maintenance is complete using `datasource recover`, `trepctl online` and optionally `mm_trepctl online`.

#### 5.15.1. 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
   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.
5.15.2. 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>Shunned</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>6</td>
<td>Validate the host1 server configuration</td>
<td>tpm validate</td>
<td>Shunned</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>7</td>
<td>Recover the slave (host1) back</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>
<tr>
<td>9</td>
<td>Switch master back to host1</td>
<td>switch to host1</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>10</td>
<td>Set automatic policy</td>
<td>set policy automatic</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
</tbody>
</table>

5.15.3. 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 host2</td>
<td>datasource host2 shun</td>
<td>Master</td>
<td>Shunned</td>
<td>Slave</td>
</tr>
<tr>
<td>4</td>
<td>Perform maintenance</td>
<td></td>
<td>Master</td>
<td>Shunned</td>
<td>Slave</td>
</tr>
<tr>
<td>5</td>
<td>Validate the host2 server configuration</td>
<td>tpm validate</td>
<td>Master</td>
<td>Shunned</td>
<td>Slave</td>
</tr>
<tr>
<td>6</td>
<td>Recover the slave (host2) 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 (host2) has caught up</td>
<td></td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>8</td>
<td>Shun slave host3</td>
<td>datasource host3 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>
</tbody>
</table>
## 5.15.4. 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, adding/removing columns or migrating table data between table definitions, is best performed individually on each dataserver while it has been temporarily taken out of the dataservice.

As with all maintenance operations it is advisable to have fully tested your DDL in a staging environment. In some cases, the impact of DDL change is minimal and therefore can safely applied to the Master node and allowing the change to be replicated down to the slaves.

In situations where the overhead of the DDL change would cause an outage to your application through table locking, use the rolling maintenance procedure below which is specific for DDL changes.

The basic process comprises of a number of steps, these are as follows:

1. If the DDL adds or removes columns, then enable the `colnames` and `dropcolumn` filters
2. If the DDL adds or removes tables, and you do not want to simply apply to the master and allow replication to handle it, then enable the `replicate` filters
3. Perform schema changes following the process summarised in the table below
4. Optionally, remove the filters enabled in the first step

Enable filters for column changes

The use of the `colnames` and `dropcolumn` filters allow you to make changes to the structure of tables, without impacting the flow of replication.

### Important

During these schema changes, and whilst the filters are in place, applications **MUST** be forwards and backwards compatible, but **MUST NOT** alter data within any columns that are filtered out from replication until after the process has been completed on all hosts, and the filters disabled. Data changes to filtered columns will cause data drift and inconsistencies, resulting in potentially unexpected behaviour.

- To enable the filters, first create a file called `schemachange.json` in a directory accessible by the OS user that the software is running as.

Typically, this will the tungsten user and the default location for this file will be `/opt/continuent/share`

The file needs to contain a JSON block outlining ALL the columns being added and removed from all tables affected by the changes.

In the example below, we are removing the column, `operator_code` and adding `operator_desc` to the `system_operators` table and adding the column `action_date` to the `system_actions` table:

```json
{
    "system_operators": {
        "dropcolumn": ["operator_code"]
    },
    "system_actions": {
        "addColumn": ["action_date"]
    }
}
```
• Place your cluster into **maintenance** mode

```
shell> cctrl
shell> cctrl > set policy maintenance
```

**Note**

- If running a Composite Multimaster or Composite HA/DR topology, issue the command at the top global level to place all clusters in maintenance, or execute individually within each cluster.

• Next, enable the filters within your configuration by adding the following two parameters to the `tungsten.ini` (if running in INI method) to the `[defaults]` section on EVERY cluster node:

  ```
  svc-extractor-filters=colnames,dropcolumn
  property=replicator.filter.dropcolumn.definitionsFile=/opt/continuent/share/schemachange.json
  ```

  Followed by `tpm update` to apply the changes:

• Or, if running as a staging install:

```
shell> cd staging_dir
shell> tools/tpm update alpha \
  --svc-extractor-filters=colnames,dropcolumn \ 
  --property=replicator.filter.dropcolumn.definitionsFile=/opt/continuent/share/schemachange.json
```

• Monitor replication to ensure there are no errors before continuing.

Enable filters for adding/removing tables

The use of the `replicate` filter allows you to add and remove tables without impacting the flow of replication.

**Important**

During these schema changes, and whilst the filter is in place, applications **MUST** be forwards and backwards compatible, but **MUST NOT** modify data in any new tables until after the process has been completed on all hosts, and the filters disabled. Data changes to filtered tables will cause data drift and inconsistencies, resulting in potentially unexpected behaviour.

• Place your cluster into **maintenance** mode.

```
shell> cctrl
shell> cctrl > set policy maintenance
```

**Note**

- If running a Composite Multimaster or Composite HA/DR topology, issue the command at the top global level to place all clusters in maintenance, or individually within each cluster.

• Next, enable the filters within your configuration by adding the following two parameters to the `tungsten.ini` (if running in INI method) to the `[defaults]` section on EVERY cluster node.

- In this example we plan to **ADD** the table `system_actions` and **REMOVE** the table `system_operations`, both within the `ops` Schema:

  ```
  svc-extractor-filters=replicate
  property=replicator.filter.replicate.ignore=ops.system_actions,ops.system_operations
  ```

  Followed by `tpm update` to apply the changes.

• Or, if running as a staging install:

```
shell> cd staging_dir
shell> tools/tpm update alpha \
  --svc-extractor-filters=replicate \ 
  --property=replicator.filter.replicate.ignore=ops.system_actions,ops.system_operations
```

• Monitor replication to ensure there are no errors before continuing.
Apply DDL Changes

- Follow the steps outlined in the table below to make the DDL changes to all nodes, in all clusters.

- If filtering columns, once all the changes have been complete, edit the `schemachange.json` to contain an empty document:

  ```shell
echo "[]" > /opt/continuent/share/schemachange.json
  ``

  Then, restart the replicators:

  ```shell
tcpctl replicator restart
  ``

- If filtering tables, repeat the process of adding the replicate filter removing any tables from the ignore parameter that you have ADDED to your database.

- You can optionally fully remove the filters if you wish by removing the entries from the configuration and re-running `tpm update` however it is also perfectly fine to leave them in place. There is a potentially small CPU overhead in very busy clusters by having the filters in place, but otherwise should not have any impact.

  It is advisable to monitor the system usage and make the decision based on your own business needs.

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>trepctl -host host2 offline</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>2</td>
<td>Set the slave <code>host2</code> offline</td>
<td><code>trepctl -host host2 offline</code></td>
<td>Master</td>
<td>Offline</td>
<td>Slave</td>
</tr>
<tr>
<td>3</td>
<td>Connect to dataserver for <code>host2</code> and update schema</td>
<td><code>set sql_log_bin=0; run ddl statements</code></td>
<td>Master</td>
<td>Offline</td>
<td>Slave</td>
</tr>
<tr>
<td>4</td>
<td>Set the slave online</td>
<td><code>trepctl -host host2 online</code></td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>5</td>
<td>Ensure the slave <code>host2</code> has caught up</td>
<td><code>trepctl -host host2 status</code></td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>6</td>
<td>Set the slave <code>host3</code> offline</td>
<td><code>trepctl -host host3 offline</code></td>
<td>Master</td>
<td>Slave</td>
<td>Offline</td>
</tr>
<tr>
<td>7</td>
<td>Connect to dataserver for <code>host3</code> and update schema</td>
<td><code>set sql_log_bin=0; run ddl statements</code></td>
<td>Master</td>
<td>Slave</td>
<td>Offline</td>
</tr>
<tr>
<td>8</td>
<td>Set the slave <code>host3</code> online</td>
<td><code>trepctl -host host3 online</code></td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>9</td>
<td>Ensure the slave <code>host3</code> has caught up</td>
<td><code>trepctl -host host3 status</code></td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>10</td>
<td>Switch master to <code>host2</code></td>
<td>See Section 5.5, “Switching Master Hosts”</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>11</td>
<td>Set the slave <code>host1</code> offline</td>
<td><code>trepctl -host host1 offline</code></td>
<td>Offline</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>12</td>
<td>Connect to dataserver for <code>host1</code> and update schema</td>
<td><code>set sql_log_bin=0; run ddl statements</code></td>
<td>Offline</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>13</td>
<td>Set the slave <code>host1</code> online</td>
<td><code>trepctl -host host1 online</code></td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>14</td>
<td>Ensure the slave <code>host1</code> has caught up</td>
<td><code>trepctl -host host1 status</code></td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>15</td>
<td>Switch master back to <code>host1</code></td>
<td>See Section 5.5, “Switching Master Hosts”</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.

5.15.5. Upgrading or Updating your JVM

When upgrading your JVM version or installation, care should be taken as changing the JVM will momentarily remove and replace required libraries and components which may upset the operation of Tungsten Clustering while the upgrade or update takes place.
For this reason, JVM updates or changes must be treated as an OS upgrade or event, requiring a master switch and controlled stopping/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>3</td>
<td>Stop all services on host2</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 host2 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 (host2) has caught up</td>
<td>ls</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>8</td>
<td>Shun slave host3</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 host3</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 host3 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 (host3) 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 host2</td>
<td>switch to host2</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>15</td>
<td>Shun host1</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 host1</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 host1 slave</td>
<td>startall</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>19</td>
<td>Recover host1 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 (host1) 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 host1</td>
<td>switch to host1</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
</tbody>
</table>

5.16. Performing Tungsten Clustering Maintenance

Changes to the configuration should be made with tpm update. This continues the procedure of using tpm install during installation. See Section 9.5.17, "tpm update Command" for more information on using tpm update.

For information on performing upgrades to new versions of the product with the existing configuration, see Section 4.4, "Upgrading Tungsten Clustering".

5.17. Monitoring Tungsten Clustering

It is your responsibility to properly monitor your deployments of Tungsten Clustering and Tungsten Replicator. The minimum level of monitoring must be done at three levels. Additional monitors may be run depending on your environment but these three are required in order to ensure availability and uptime.

1. Make sure the appropriate Tungsten Clustering and Tungsten Replicator services are running.
2. Make sure all datasources and replication services are ONLINE.
3. Make sure replication latency is within an acceptable range.

**Important**

Special consideration must be taken if you have multiple installations on a single server. That applies for clustering and replication or multiple replicators.
These three points must be checked for all directories where Tungsten Clustering or Tungsten Replicator are installed. In addition, all servers should be monitored for basic health of the processors, disk and network. Proper alerting and graphing will prevent many issues that will cause system failures.

5.17.1. Managing Log Files with logrotate

You can manage the logs generated by Tungsten Clustering using logrotate.

- **connector.log**
  ```sh
  /opt/continuent/tungsten-connector/log/connector.log {
    notempty
    daily
    rotate 3
    missingok
    compress
    copytruncate
  }
  ```

- **tmsvc.log**
  ```sh
  /opt/continuent/tungsten-manager/log/tmsvc.log {
    notempty
    daily
    rotate 3
    missingok
    compress
    copytruncate
  }
  ```

- **trepsvc.log**
  ```sh
  /opt/continuent/tungsten-replicator/log/trepsvc.log {
    notempty
    daily
    rotate 3
    missingok
    compress
    copytruncate
  }
  ```

5.17.2. Monitoring Status Using cacti

Graphing Tungsten Replicator data is supported through Cacti extensions. These provide information gathering for the following data points:

- Applied Latency
- Sequence Number [Events applied]
- Status [Online, Offline, Error, or Other]

To configure the Cacti services:

1. Download both files from https://github.com/continuent/monitoring/tree/master/cacti
2. Place the PHP script into /usr/share/cacti/scripts.
3. Modify the installed PHP file with the appropriate $ssh_user and $tungsten_home location from your installation:
   - $ssh_user should match the user used during installation.
   - $tungsten_home is the installation directory and the tungsten subdirectory. For example, if you have installed into /opt/continuent, use /opt/continuent/tungsten.

   Add SSH arguments to specify the correct id_rsa file if needed.
4. Ensure that the configured $ssh_user has the correct SSH authorized keys to login to the server or servers being monitored. The user must also have the correct permissions and rights to write to the cache directory.
5. Test the script by running it by hand:

   ```sh
   shell> php -q /usr/share/cacti/scripts/get_replicator_stats.php --hostname replserver
   ```

   If you are using multiple replication services, add --service servicename to the command.
6. Import the XML file as a Cacti template.
7. Add the desired graphs to your servers running Tungsten Clustering. If you are using multiple replications services, you'll need to specify the desired service to graph. A graph must be added for each individual replication service.

Once configured, graphs can be used to display the activity and availability.

Figure 5.3. Cacti Monitoring: Example Graphs

- **Graph Template: Tungsten Replicator Events Applied**
  - west1 - Tungsten Replicator Events Applied
  - Events Applied: Cur: 52.22, Avg: 16.80, Max: 57.41

- **Graph Template: Tungsten Replicator Latency**
  - west1 - Tungsten Replicator Latency
  - Latency: Cur: 971.61 ms, Avg: 1.03, Max: 1.79

- **Graph Template: Tungsten Replicator State**
  - west1 - Tungsten Replicator State
  - State: Online, Offline, Error, Other
5.17.3. Monitoring Status Using nagios

In addition to the scripts bundled with the software, there is a Ruby gem available with expanded checks and a mechanism to add custom checks. See https://github.com/continuent/continuent-monitors-nagios for more details.

Integration with Nagios is supported through a number of scripts that output information in a format compatible with the Nagios NRPE plugin. Using the plugin the check commands, such as `check_tungsten_latency` can be executed and the output parsed for status information.

The available commands are:

- `check_tungsten_latency`
- `check_tungsten_online`
- `check_tungsten_policy`
- `check_tungsten_progress`
- `check_tungsten_services`

To configure the scripts to be executed through NRPE:

1. Install the Nagios NRPE server.
2. Start the NRPE daemon:
   ```shell
   sudo /etc/init.d/nagios-nrpe-server start
   ```
3. Add the IP of your Nagios server to the `/etc/nagios/nrpe.cfg` configuration file. For example:
   ```
   allowed_hosts=127.0.0.1,192.168.2.20
   ```
4. Add the Tungsten check commands that you want to execute to the `/etc/nagios/nrpe.cfg` configuration file. For example:
   ```
   command[check_tungsten_online]=/opt/continuent/tungsten/cluster-home/bin/check_tungsten_online
   ```
5. Restart the NRPE service:
   ```shell
   sudo /etc/init.d/nagios-nrpe-server start
   ```
6. If the commands need to be executed with superuser privileges, the `/etc/sudo` or `/etc/sudoers` file must be updated to enable the commands to be executed as root through `sudo` as the nagios user. This can be achieved by updating the configuration file, usually performed by using the `visudo` command:
   ```
   nagios ALL=(tungsten) NOPASSWD: /opt/continuent/tungsten/cluster-home/bin/check*
   ```
   In addition, the `sudo` command should be added to the Tungsten check commands within the Nagios `nrpe.cfg`, for example:
   ```
   command[check_tungsten_online]=/usr/bin/sudo -u tungsten /opt/continuent/tungsten/cluster-home/bin/check_tungsten_online
   ```
   Restart the NRPE service for these changes to take effect.
7. Add an entry to your Nagios `services.cfg` file for each service you want to monitor:
   ```
   define service {
     host_name database
     service_description     check_tungsten_online
     check_command           check_nrpe! -H $HOSTADDRESS$ -t 30 -c check_tungsten_online
     retry_check_interval    1
     check_period            24x7
     max_check_attempts      3
     max_check_attempts      3
     flap_detection_enabled  1
     notifications_enabled   1
     notification_period     24x7
     notification_interval   60
     notification_options    c,f,r,u,w
     normal_check_interval   5
   }
   ```

The same process can be repeated for all the hosts within your environment where there is a Tungsten service installed.

5.18. Rebuilding THL on the Master

If THL is lost on a master before the events contained within it have been applied to the slave[s], the THL will need to be rebuilt from the existing MySQL binary logs.
Operations Guide

**Important**

If the MySQL binary logs no longer exist, then recovery of the lost transactions in THL will NOT be possible.

The basic sequence of operation for recovering the THL on both master and slaves is:

1. Gather the failing requested sequence numbers from all slaves:
   
   ```shell>
trepctl status
```
   
   ```pendingError          : Event extraction failed
   pendingErrorCode       : NONE
   pendingErrorEventId    : NONE
   pendingErrorSeqno      : -1
   pendingExceptionMessage: Client handshake failure: client response validation failed:
   Master log does not contain requested transaction:
   master source ID=db1 client source ID=db2 requested seqno=4 client epoch number=0 master min seqno=8 master max seqno=8```

   In the above example, when slave db2 comes back online, it requests a copy of the last seqno in local thl (4) from the master db1 to compare for data integrity purposes, which the master no longer has.

   Keep a note of the lowest sequence number and the host that it is on across all slaves for use in the next step.

2. On the slave with the lowest failing requested seqno, get the epoch, source-id and event-id [binlog position] from the THL using the command `thl list -seqno [330]` specifying the sequence number above. This information will be needed on the extractor (master) in a later step. For example:

   ```tungsten@db2:/opt/replicator> thl list -seqno 4
   SEQ# = 4 / FRAG# = 0 (last frag)
   - TIME = 2017-07-14 14:49:00.0
   - EPOCH# = 0
   - EVENTID = mysql-bin.000009:0000000000001844;56
   - SOURCEID = db1
   - METADATA = mysql_server_id=33155307;dbms_type=mysql;is_metadata=true;
   service=east;shard=#UNKNOWN;heartbeat=None]
   - TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
   - OPTIONS = 
   - SCHEMA = tungsten_east
   - SQL(0) = UPDATE tungsten_east.heartbeat SET source_tstamp= '2017-07-14 14:49:00', salt= 5, name= 'NONE' WHERE id= 1```

   There are two more ways of getting the same information, use the one you are most comfortable with:

   ```tungsten@db2:/opt/replicator> dsctl get
   ["extract_timestamp":"2017-07-14 14:49:00.0","eventid":"mysql-bin.080009:0800000000001844;56","fragno":0,"last_frag":true,"seqno":4,"update_timestamp":"2017-07-14 14:49:00.0","
   "shard_id":"#UNKNOWN","applied_latency":0,"epoch_number":0,"task_id":0,"source_id":"db1"]
   
   tungsten@db2:/opt/replicator> tungsten_get_position
   {
   "applied_latency": 0,
   "epoch_number": 0,
   "eventid": "mysql-bin.080009:0800000000001844;56",
   "extract_timestamp": "2017-07-14 14:49:00.0",
   "fragno": 0,
   "lastfrag": 1,
   "seqno": 4,
   "shard_id": "#UNKNOWN",
   "source_id": "db1",
   "task_id": 0,
   "update_timestamp": "2017-07-14 14:49:00.0"
   }
   
   3. Clear all THL on the master since it is no longer needed by any slaves:

   ```shell> thl purge```

4. Use the `tungsten_set_position` command on the master with the values we got from the slave with the lowest seqno to tell the master replicator to begin generating THL starting from that event in the MySQL binary logs:

   ```shell> tungsten_set_position --seqno=4 --epoch=0 --source-id=db1 --event-id=mysql-bin.000009:0000000000001844```

   You may also use `dsctl`, but that requires executing the `dsctl reset` command first.

5. Switch the master to online state:

   ```221```
6. Switch the slaves to online state once the master is fully online:

```
shell> trepcctl online
```
Chapter 6. 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 6.1. Tungsten Connector Basic Architecture

In addition to this basic structure, Tungsten Connector also works with the other components of Tungsten Clustering to handle some specific scenarios and situations:

- The connector works in harmony with the Tungsten Manager as part of Tungsten Clustering 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 Tungsten Clustering 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.

To start using the connector run the `tpm connector` command. This will open a connection with the MySQL CLI. See Section 6.3, “Clients and Deployment” for more detail on configuring your application to use the connector.

Important

After installation the connector will only work with the `--application-user` and `--application-password` that were provided during installation. See Section 6.6, “User Authentication” if you need more information on adding users to `user.map`.

6.1. 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 6.2, “Basic MySQL/Application Connectivity”.

Shell > tpm connector
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 93422
Server version: 5.5.34-log-tungsten MySQL Community Server (GPL)
Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>
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 6.3, “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 Tungsten Clustering 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 Tungsten Clustering 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.
For example, in Figure 6.5, “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.

### 6.1.1. Connector Control Flow

The Connector makes a TCP connection to any available Manager, then all command-and-control traffic uses that channel. The Manager never initiates a connection to the Connector.

When there is a state change (i.e. shun, welcome, failover, etc.), the Manager will communicate to the Connector over the existing channel. The Connector will re-establish a channel to an available Manager if the Manager it is connected to is stopped or lost.

### 6.2. Basic Connector Configuration

By default, the Tungsten Connector listens on port 9999 so as to avoid conflicting with the default MySQL listener port of 3306.

The best practice is to change the MySQL port to 13306 and the Tungsten Connector port to 3306.

For example, modify `my.cnf` to define the port, then restart MySQL:

```
port = 13306
```

For more information, see Section B.3.2, “MySQL Configuration”.

Configure the the Tungsten Connector to listen on port 3306:

```
shell > tpm update alpha --application-port=3306
```

**Important**

This change is especially important if the Tungsten Connector will be installed directly on the database nodes. ALL traffic needs to flow through the Tungsten Connector, so hiding the actual MySQL port prevents mistaken connections directly to the database.

### 6.3. Clients and Deployment

In order to get the benefits of Tungsten Clustering your application must use the Tungsten Connector. The connector is compatible with MySQL drivers and applications. Use the `tpm connector --samples` command to see examples of how you can invoke a connection on your own. You may need to adapt these examples to your application and configuration method but the connection details should be the same.

```
shell > tpm connector --samples
```
Tungsten Connector

After installation the connector will only work with the \(--application-user\) and \(--application-password\) options that were provided during installation. See Section 6.6, "User Authentication" if you need more information on adding users to user.map.

By default the connection will always be sent to the current master. This behavior can be modified by implementing one of the routing methods to send some traffic to slave datasources.

6.3.1. Connection Pools

Tungsten Connector can work behind a connection pool without any issue.

Upon switch or failover, all connections in the pool will be broken but not closed. The very next time one of these pooled connections is used, it will be transparently reconnected unless autoReconnect has been disabled by the tpm installation flag:

```shell
shell > tpm update alpha --connector-autoreconnect=false
```

Important

When using a connection pool mechanism in applications, the dynamic connection setting maxAppliedLatency should not be used. This setting is only meaningful at connection time; since connection pools open a set of connections, the applied latency will most probably be outdated when the application will actually use the pooled connection.

6.4. Routing Methods

Bridge Mode is Default in Tungsten Clustering 5.0.0. Bridge mode is now the default connector routing method if no other routing method is explicitly stated. The bridge-mode does not use the user.map file which reflects other changes to take a more secure default deployment. A warning will be displayed during the validation process to tell you if bridge-mode is being enabled. It will not be enabled in the following cases:

- The \(--connector-smartscale\) option is set to true.
- The user.map file contains @direct entries.
- The user.map file contains @hostoption entries.
- The \(--property=selective.rwsplitting\) connector option is set to true.

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.
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 6.4.1, “Connector Routing Methods”
- Quality of Service (QoS) specification, see Section 6.4.2, “Connector Quality of Service (QoS) Selection”
- Load balancer selection (implied by QoS), see Section 6.4.3, “Connector Load Balancers”
- Slave latency, including optional maximum latency setting, see Section 6.4.4, “Specifying Required Latency”
- Affinity specification, see Section 6.4.5, “Setting Read 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</td>
<td>Yes</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 6.7, “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.

### 6.4.1. Connector Routing Methods

The supported routing methods, typical uses and use cases are listed in Table 6.1, “Routing Method Selection”.

---

**Figure 6.6. Tungsten Connector routing architecture**

![Connector Routing Architecture Diagram](image-url)
Table 6.1. 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>Maximum Applied Latency</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 6.4.10, “Host-based Routing” or Section 6.4.11, “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 6.4.9, “SQL Routing”.
- If the application uses non-auto-commit statements (for example, Hibernate), Section 6.4.10, “Host-based Routing”, or Section 6.4.9, “SQL Routing”.
- If the application does not fit any of these categories, or is not replication aware use either either Section 6.4.8, “Direct Routing” or Section 6.4.7, “SmartScale Routing”.

6.4.2. 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 [229]**
  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 [229]**
  This setting indicates that the query is a write and should be directed to a master.
- **RW_SESSION [229]**
  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 selected routing mechanism together.

<table>
<thead>
<tr>
<th>SmartScale</th>
<th>QoS</th>
<th>Maximum Applied Latency</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>RO_RELAXED</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>RO_RELAXED</td>
<td>Specified</td>
<td>Not Specified</td>
<td>Slave:Master</td>
<td>Only if slave latency &lt; -connec-tor-max-applied-latency [443]</td>
<td>Yes</td>
</tr>
</tbody>
</table>
6.4.3. 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>RW_STRICT</td>
<td>Always selects the master data source</td>
</tr>
<tr>
<td>MostAdvancedSlaveLoadBalancer</td>
<td>RO_RELAXED</td>
<td>Selects the slave that has replicated the most events, by comparing data sources “high water” 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 cctrl 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>RW_SESSION</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>

The default setting is MostAdvancedSlaveLoadBalancer.

To change the Connector load balancer, specify the property in the configuration, i.e to use Round Robin:

```shell
dirtools/tpm update alpha \
  --property=dataSourceLoadBalancer_RO_RELAXED=com.continuent.tungsten.router.resource.loadbalancer.RoundRobinSlaveLoadBalancer
```

6.4.4. Specifying Required Latency

Depending on the selected routing method, load balancer and QoS setting, a slave will automatically be chosen when the host connects. The maximum allowed latency 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:

```sql
jdbc://connector1:3306/database?maxAppliedLatency=5
```

Will specify that a host with a replication latency of less than 5 seconds should be selected.

The option can be set globally by configuring the JDBC options used by the connector via the --connector-max-slave-latency [443] option to tpm (in seconds):

```shell
dirtools/tpm update alpha --connector-max-slave-latency=10
```

**Warning**

The Connector computes latency by polling the Replicator every 3 seconds for the current replication-view latency. This gives the Connector an accuracy of +/- 3 seconds, which means that values of 3 or less will not function as expected.

For any queries that have a very low tolerance for replication latency, we strongly suggest you read directly from the master database server only. This ensures the latest data is being read.

**Important**

The --connector-max-slave-latency [443] flag does not ensure the slave has applied the latest sequence number, just that its latency at the last commit was under the specified number. This behavior can be adjusted by specifying --use-relative-latency=true [445] in the configuration.

**NOTE:** --use-relative-latency=true [445] is a cluster-wide setting, cctrl and trepctl will also report relative latency based on this setting.

6.4.4.1. Applied and Relative Latency Comparison

- Applied Latency
The appliedLatency is the latency between the commit time of the source event and the time the last committed transaction reached the end of the corresponding pipeline within the replicator. Within a master, this indicates the latency between the transaction commit time and when it was written to the THL. In a slave, it indicates the latency between the commit time on the master database and when the transaction has been committed to the destination database. Clocks must be synchronized across hosts for this information to be accurate.

appliedLatency : 0.828 The latency is measured in seconds. Increasing latency may indicate that the destination database is unable to keep up with the transactions from the master. In replicators that are operating with parallel apply, appliedLatency indicates the latency of the trailing channel. Because the parallel apply mechanism does not update all channels simultaneously, the figure shown may trail significantly from the actual latency.

See Section E.2.6, “Terminology: Fields appliedLatency” for more information.

• Relative Latency

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.

See Section E.2.70, “Terminology: Fields relativeLatency” for more information.

• Comparing Relative and Applied Latencies

Both relative and applied latency are visible via the trepctl status. Relative indicates the latency since the last time the appliedLastSeqno advanced; for example:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
...                     ...
appliedLastEventId     : mysql-bin.000211:0000000020094766;0
appliedLastSeqno       : 79822
appliedLatency         : 0.571
...                     ...
relativeLatency        : 8.944
Finished status command...
```

In this example the last transaction had an applied latency (time to write to the slave DB) of 0.571 seconds from the time it committed on the master, and last committed something to the slave DB 8.944 seconds ago.

If relative latency increases significantly in a busy system, it may be a sign that replication is stalled. This is a good parameter to check in monitoring scripts.

• For more information, see:
  --connector-max-slave-latency
  --use-relative-latency
  Section 4.1.6.3, “Relative Latency”

### 6.4.4.2. Advanced Troubleshooting for Latency-based Routing

To troubleshoot the latency-based routing decisions the connector makes, and uncomment the below lines in /opt/continuent/tungsten/tungsten-connector/conf/log4j.properties

```
#log4j.logger.com.continuent.tungsten.router.resource.loadbalancer=debug, stdout
#log4j.additivity.com.continuent.tungsten.router.resource.loadbalancer=false
```

The log will then show and explain what node the connector choose and why.

---

**Important**

No connector restart is required to enable logging. If you re-comment the lines, a restart will be required. To disable without a connector restart, replace the word `debug` with the word `info`.

### 6.4.5. Setting Read 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 be specified within the connection string:

jdbc://connector1:3306/database?affinity=Host1&qos=RO_RELAXED

Additionally, affinity JDBC options can be set globally via the tpm command option connector-affinity [441].

6.0.3. The read affinity setting now supports multiple dataservices with ordering and exclusion (only one was previously allowed).

You may now fine-tune the affinity by specifying an ordered list of dataservice names.

• The *affinity* string is an ordered list of dataservice names, separated by commas, where the first dataservice entry will be the one used by default. If the first dataservice in the list is not available, the connector will use the next one listed, and so forth.

• Dataservices not specified in the list, if any, will be used last and randomly.

• It is also possible to exclude one or more dataservices by adding a hyphen ("-") in front of the dataservice name.

Affinity can also be combined with other node selection, such as QoS. For example, by combining the affinity and RO_RELAXED [229], 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 datasource with three nodes, where node1 is the master:

```
shell> mysql -h127.0.0.1 -P3306 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 datasource within the composite datasource. For example in a composite service with east and west physical dataservices:

```
shell> mysql -h127.0.0.1 -P3306 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
```

6.4.6. Connector Datasource Selection in Composite Clusters

Composite Datasource Selection in Tungsten Clustering 6.0.3. Deterministic datasource selection was enabled in Tungsten Clustering 6.0.3.

Within composite clusters, both composite and composite multimaster, the selection of datasources can be made to be deterministic by specifying the list of datasources within the *user.map* configuration. The configuration is based on an ordered, comma-separated list of services to use which are then selected in order.

The format of the specification is as follows:

```
user pass compositeDS datasource_by_order_of_priority"[[dataservice_to_exclude]"
```

The specification operates on the following rules:

• List of service names in order

• If the service name has a dash prefix it is always explicitly excluded from the list of available datasources

• If a datasource is not specified, it will always be picked last

For example, in a setup made of three data service, usa, asla and europe using the following configuration:

```
app_user secret usa,asla,-europe
```

Will select data sources in data service usa. If usa is not available, in asla. If asla is not available, then connection will not succeed since europe has been negated.

6.4.7. Smartscale Routing

SmartScale allows you to read your data, as much as possible, from a slave data source.
In this read-write splitting mode, the connector intelligently determines if slaves are up-to-date with respect to the master, and selects them in such a way that read operations are always strictly consistent with the last write of their current session. Sessions are per-connector-instance, in-memory objects that allow different connections to share SmartScale benefits: by providing the same session id, two connections will be able to see each other's write operations consistently.

Possible session ids are:

- **DATABASE**: applications will see write operations made to the same database as it is connected to. Reads from other databases might be outdated depending on the slave latency
- **USER**: all connections that use the same user will read data consistent with the writes made by the current user. Other users data might be outdated.
- **CONNECTION**: only writes made by the current connection are guaranteed to be read consistently. Writes from other connections might be outdated
- **PROVIDED_IN_DBNAME**: Allows you to specify a variable session id in the database connection string. An application, typically PHP, can pass its own session id to make Smart Scale even more efficient.

Typical use cases

- Applications which can use this level of consistency typically do relatively few writes and many reads. The writes that are performed can be considered to be in a 'silos' of their own, that is, a given application 'session' only writes and reads its own data and is not concerned with the data read/written by other application 'sessions'.
- PHP applications are good candidates for Smart Scale since PHP has embedded session IDs that can be passed at connection time.
- Web based applications with user profiles match the scenario where users will update their own profile and want to see their modifications right away, but can accept latency on other users profiles.

Comparison with Direct Reads

- Smart Scale allow session consistency, while Direct Reads always read from slave, no matter whether data is up-to-date.
- However, the cost for consistency appears at the performance level, since the Connector constantly needs to check slave progress.
- If your application needs to see the data it just wrote, use Smart Scale
- If your application does a lot of small reads that do not need to be up-to-date, use Direct Reads

Limitations

- Prepared Statements - Prepared statements will need to be enclosed between transaction boundaries in order to work correctly with read/write splitting. This way, they will always execute on the master. Note that all prepared statements will become invalid upon switches or failover
- Ephemeral objects - Temporary tables, session variables and other objects that are connection specific will not be accessible when reading data using Smart Scale. If you need to use these ephemeral object, you should either add a "for update" statement in you selects or avoid using SmartScale
- Read/write functions - Functions that create or modify data in the database should never be called with a simple SELECT statement. Always add "for update" a the end of the select call. Example:

  ```
  SELECT my_function_that_writes('param') FOR UPDATE
  ```

6.4.7.1. Specifying the Session ID

While the three first keywords [DATABASE, USER and CONNECTION] are connector-wide (a single connector instance will use these session ids for all connections), it is possible to configure the connector to allow a free string session ID. This string will have to be passed through the database name as `{db_name}?sessionId={sessionID}`. For example, when using a database named "test" and a session ID number 1234, the database name passed to the connector will be:

```
test?sessionId=1234
```

With `mysql` command line utility, the connection command will look like:

```
mysql -h connectorHost -u user -ppass -P 3306 test?sessionId=1234
```

JDBC clients will have to pass this session ID with a special tag, as follows:

```
jdbc:mysql://connector_host:3306/dbname?sessionId=1234#otherJdbcDriverOption=value
```
In order to use this feature, the special session id \textit{PROVIDED\_IN\_DBNAME} needs to be specified at installation time.

Also note that a session ID specified in the database name string will override the default provided in the connector configuration file. You can thus have a default session ID set for applications that can’t specify it dynamically, and still allow other applications to connect with their own session ID variable.

### 6.4.7.2. Enabling SmartScale Routing

To enable SmartScale routing, configure the dataservice using the \texttt{--connector-smartscale} option. The session ID identification should also be specified by using the \texttt{--connector-smartscale-sessionid} option with one of the following values \texttt{DATABASE}, \texttt{USER}, \texttt{CONNECTION} or \texttt{PROVIDED\_IN\_DBNAME}:

```shell
tpm configure alpha \
  ... \
  --connector-smartscale=true \
  --connector-smartscale-sessionid=\{DATABASE\|USER\|CONNECTION\|PROVIDED\_IN\_DBNAME\}
```

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 \texttt{REPLICATION\_CLIENT} privilege so that the user can compare the current replicator progress for session information. This can be granted using:

```shell
GRANT REPLICATION CLIENT ON \*.* to app_user@\%'
```

### 6.4.7.3. Disabling SmartScale Routing

To disable SmartScale routing if it has been previously configured:

```shell
tpm configure alpha --connector-smartscale=false
```

### 6.4.7.4. SmartScale Exploit

When using DATABASE session ID, you can bypass session consistency to read from available slaves by simply connecting to one database and reading from another. For example:

```shell
mysql -u... -p... -P3306 db1
mysql> select * from db2.user
```

As long as no update is made on db1 in the meantime, the select will be executed on a slave (if available).

### 6.4.8. 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 of 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 should use \texttt{SmartScale}.

Read/Write splitting is supported through examination of the submitted SQL statement:

- If the statement contains \texttt{SELECT} and does not contain \texttt{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.

6.4.8.1. Enabling Direct Routing

To enable direct routing for a specific user, the `user.map` must be modified. Update the file with the `@direct` directive on every host running a connector. The connector will automatically read the changes after the file is saved. For example:

```plaintext
@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.

6.4.8.2. 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:

```plaintext
BEGIN
PREPARE ...
EXECUTE ...
COMMIT
```

• Ephemeral objects (i.e. anything that is not replicated), will not be available on the slave. Session variables are an excellent example of this.

• Stored procedures that update data in the database should never be called using a basic `SELECT` statement:

```plaintext
mysql> SELECT update_function('data');
```

Instead, add the `FOR UPDATE` keywords to ensure it is routed to the master:

```plaintext
mysql> SELECT update_function('data') FOR UPDATE;
```

6.4.9. SQL Routing

<table>
<thead>
<tr>
<th>Auto Read/Write Splitting</th>
<th>No</th>
</tr>
</thead>
<tbody>
<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, or whatever slave is selected by the read-write splitting configuration, if enabled. To specify that a statement can be executed on the slave, place a comment before the statement:

```plaintext
/* TUNGSTEN USE qos=RO_RELAXED */ SELECT * FROM TABLENAME
```

This style of comment indicates to the connector that the specific query that follows should go to a slave. If unavailable, the query may still be executed on the master.

```plaintext
-- TUNGSTEN USE qos=RO_RELAXED
```

This style of comment indicates to the connector that all queries that follow should go to a slave. If unavailable, any query may still be executed on the master.

**Warning**

If you force the Connector to send traffic to a slave using `qos=RO_RELAXED`, then any write operations that follow will also go to the slave until you tell the Connector to go back to the master by indicating `qos=RW_STRICT`. The application is fully responsible for where the traffic is routed to. If care is not taken, the application could send writes to a slave this way which is unacceptable from a clustering perspective. All writes must go to the master or they will be lost to a non-authoritative node, and may corrupt the data badly.

The below forces all following queries to go to the master directly, effectively "turning off" reads from the slave.
Important

Please note that employing the -- style will override any /* */ comments.

6.4.9.1. Enabling SQL Routing

To enable SQL routing, use the following operations with tpm:

```shell
shell> tpm configure alpha
    --property=selective.rwsplitting=true
```

6.4.9.2. 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
shell> mysql -c -h host -u tungsten -ppassword -P3306 test
```

6.4.10. Host-based Routing

<table>
<thead>
<tr>
<th>Auto Read/Write Splitting</th>
<th>No</th>
</tr>
</thead>
<tbody>
<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
shell> mysql -h master.localhost
```

Will connect to the currently active master, while:

```shell
shell> mysql -h slave.localhost
```

Would connect to any currently available slave.

6.4.10.1. 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
   ```
When using DNS to resolve addresses, the DNS should also be updated with hostnames to match those configured for each IP interface.

Update the `user.map` file on every host running a connector to reflect the desired QoS for each hostname. The connector will automatically read the changes after the file is saved.

```plaintext
@hostoption master.host1 qos=RW_STRICT
@hostoption master.host2 qos=RW_STRICT
@hostoption master.host3 qos=RW_STRICT
@hostoption slave.host1 qos=RO_RELAXED
@hostoption slave.host2 qos=RO_RELAXED
@hostoption slave.host3 qos=RO_RELAXED
```

Once configured, client applications must be configured to select the appropriate host based on the operation they are performing.

### 6.4.10.2. 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.

### 6.4.11. Port-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 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 -P3306
```

Will connect to the currently active master, while:

```shell
shell> mysql -P3307
```

will connect to a read-resource, ideally a slave, but will revert to the master if no appropriate slave is available.

**Note**

The ports to be used for each connection type are configurable during installation.

### 6.4.11.1. 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:

```bash
shell> tpm configure alpha \
    --connector-readonly-listen-port=3307 \
    --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` [229].

### 6.4.12. 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 for a standalone connector, or a single connector within a dataservice.
Tungsten Connector

This setting places the connector into RO_RELAXED mode. The connector will choose the master if there is no available slave. See Section 6.4.2, "Connector Quality of Service (QoS) Selection" for more detail on Quality of Service modes.

To enable this functionality, configure the connector using the --connector-readonly option:

```shell
tpm configure alpha --connector-readonly=true
```

To enable this functionality on specific hosts only, add the --hosts option:

```shell
tpm configure alpha --connector-readonly=true --hosts=host1,host3
```

### 6.5. Using Bridge Mode

Bridge Mode is the default in Tungsten Clustering 5.0.0. Bridge mode is now the default connector routing method if no other routing method is explicitly stated. The bridge-mode does not use the `user.map` file which reflects other changes to take a more secure default deployment. A warning will be displayed during the validation process to tell you if bridge-mode is being enabled. It will not be enabled in the following cases:

- The --connector-smartscale option is set to true.
- The `user.map` file contains `@direct` entries.
- The `user.map` file contains `@hostoption` entries.
- The --property=selective.rwsplitting connector option is set to true.

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 6.7. Tungsten Connector Bridge Mode Architecture**

Bridge mode provides a simpler method for connecting clients to MySQL, but with reduced facilities, as outlined in the table below:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Proxy 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>Automatic Read/Write Splitting</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Application-based Read/Write Splitting</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Seamless Reconnects</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Data Source Selection</td>
<td>Current data source is checked to confirm latency and affinity</td>
<td>Pass-through</td>
</tr>
<tr>
<td>Session KeepAlive</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.

One key difference is in how slave latency checking is handled:

- In Bridge mode, the latency is checked at connection time, then you will stick to the slave for the connection lifetime (which can be shortened if the slave goes offline).
- In Proxy mode, the latency is re-evaluated before each transaction, which can bring the connection to another slave if the latency becomes too high during the life of the connection.

If you have long-lasting read-only connections that should not read from stale slaves, then use Proxy mode.

If your connection lifetime is short (i.e make/break - one transaction then disconnect), or your application is not sensitive to reasonably outdated data for reads, then use Bridge mode and its optional read-only port.

### 6.5.1. Enabling Bridge Mode

To enable Bridge Mode, the `--connector-bridge-mode` option to `tpm` must be set to `true`:

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

**Show Staging**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tpm query staging</code></td>
<td>Tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41</td>
</tr>
<tr>
<td>`echo The staging USER is $(tpm query staging</td>
<td>cut -d: -f1</td>
</tr>
<tr>
<td>`echo The staging HOST is $(tpm query staging</td>
<td>cut -d: -f1</td>
</tr>
<tr>
<td>`echo The staging DIRECTORY is $(tpm query staging</td>
<td>cut -d: -f2`</td>
</tr>
<tr>
<td><code>ssh $(STAGING_USER)@$(STAGING_HOST)</code></td>
<td><code>cd $(STAGING_DIRECTORY)</code></td>
</tr>
</tbody>
</table>

Run the `tpm` command to update the software with the Staging-based configuration:

Run the `tpm` command to update the software with the Staging-based configuration:

**Show INI**

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

[alpha]
  ...
  Connector-bridge-node=true
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

---

239
Run the `tpm` command to update the software with the INI-based configuration:

```
shell> tpm query staging
```

```
Tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
shell> cd {STAGING_DIRECTORY}
```

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

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

The default value is `true`, i.e. use the connector in bridge mode.

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 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 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.

### 6.6. 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 6.8. 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.

### 6.6.1. `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 6.6.7, "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
  \t # 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.

5.3.0. The rules for passwords were updated to support quoting.

- Quotes `’` and double quotes `"` are now supported in the user.map password.

- If there's a space in the password, the password needs to be surrounded with `’` or `"`

  ```
  "password with space"
  ```

- If there's one or several `’` or `"` in the password without space, the password doesn't need to be surrounded

  ```
  my"pas’w’or"d"
  ```

- If the password itself starts and ends with the same quote (`’` or `"`), it needs to be surrounded by quotes.

  ```
  "mypassword" so that the actual password is 'mypassword'.
  ```
As a general rule, if the password is enclosed in either single or double quotes, these are not included as part of the password during 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.

The affinity feature routes both reads and writes when using a Multimaster topology.

**6.0.3.** The read affinity setting now supports multiple dataservices with ordering and exclusion (only one was previously allowed).

You may now fine-tune the affinity by specifying an ordered list of dataservice names.

- The **affinity** string is an ordered list of dataservice names, separated by commas, where the first dataservice entry will be the one used by default. If the first dataservice in the list is not available, the connector will use the next one listed, and so forth.
- Dataservices not specified in the list, if any, will be used last and randomly.
- It is also possible to exclude one or more dataservices by adding a hyphen ("-" ) in front of the dataservice name.

For example, to configure the user **sales** with the password **secret** to use MySQL servers within the **alpha** dataservice:

```
sales secret alpha
```

To configure a user that has no password:

```
sales - alpha
```

To configure a user within a composite service:

```
sales secret global
```

To configure a user within a composite service, preferring the **east** service for read-only connections:

```
sales secret global east
```

Composite Specification in 6.0.3

The deployment used in the examples below consists of a composite dataservice **global**, with four member dataservices (i.e. one cluster per site): **east**, **west**, **north** and **south**.

For example, to exclude site **south** from servicing read requests:

```
sales global east,west,north,-south
```

The above affinity string "**east**,west,north,-south" will try **east**, then **west**, then **north**. If none of the first three are available, a connection request would not succeed since **south** has been excluded by the negation ("- ").

In the following example, dataservices **north** and **south** would be available as random candidates if the first two (**east** and **west**) were unavailable:

```
sales secret global east,west
```

**6.6.2. user.map Direct Routing**

To enable direct reads, as defined within Section 6.4.8, "Direct Routing", the entries for the user within **user.map** must be prefixed using the @direct. For example:

```
@direct sales
```

Note that the standard user, password and service must be defined:

```
sales secret alpha
@direct sales
```

For limitations of direct routing, see Section 6.4.8.2, "Limitations of Direct Routing".

**6.6.3. 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 6.4.10, "Host-based Routing".
When configuring the host options, the hostnames must have previously been defined and be resolvable.

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
```

### 6.6.4. `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` directive, use the `tungsten flush privileges` command:

  ```
  mysql> tungsten flush privileges;
  +--------------------------------+
  | Message                        |
  +--------------------------------+
  | user.map reloaded successfully |
  +--------------------------------+
  1 row in set (0.00 sec)
  ```

  If you are using the `--application-readonly-port` option, this command must be run through both ports. Alternatively, you can trigger a simultaneous flush by running:

  ```
  shell> touch /opt/continuent/tungsten/continuent-connector/conf/user.map
  ```

- When using `@direct` entries in `user.map`, the connector may need to be restarted using `connector restart`:

  ```
  shell> 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`.

### 6.6.5. 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`:

```plaintext
@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.:

  ```plaintext
  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 you are using the `--application-readonly-port` option, this command must be run through both ports. Alternatively, you can trigger a simultaneous flush by running:

  ```
  shell> touch /opt/continuent/tungsten/continuent-connector/conf/user.map
  ```

- 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:
Tungsten Connector

```
#!/bin/bash

echo 'app_user password dsone'
```

This generates a simple user entry.

### 6.6.6. Encrypting `user.map` Data

The `user.map` file allows you to use an encrypted version of the file by using the `@script` directive. Here is an example of how you can decrypt a file and return the results to `user.map`.

1. Change to a directory outside of the currently installed Tungsten

   Do this to ensure that the OpenSSL key and encrypted file are available after upgrades and other operations.

   ```shell
cd /opt/continuent/share
```

2. Create an OpenSSL key

   In this example we will use a 1024-bit RSA private key to do the encryption and decryption. There are many options for encrypting and decrypting files but this documentation will not explore those. The same process will work with other encryption techniques. You must ensure that the decryption command runs without user input.

   ```shell
   openssl genrsa -out usermap.pem 1024
   ```

3. Create the encrypted file of `user.map` entries:

   ```
   tungsten secret nyc_sjc sjc
   tungsten_sjc secret sjc
   tungsten_nyc secret nyc
   ```

   Create an encrypted version of the file:

   ```shell
   openssl rsautl -encrypt -inkey usermap.pem -in user.map.entries -out user.map.entries.ssl
   ```

4. Test decryption of the encrypted file:

   ```shell
   openssl rsautl -decrypt -inkey usermap.pem -in user.map.entries.ssl
   ```

   This should return the unencrypted `user.map`:

   ```
   tungsten secret nyc_sjc
   tungsten_sjc secret sjc
   tungsten_nyc secret nyc
   ```

5. Update the installed and configured `tungsten-connector/conf/user.map` file:

   ```
   ...
   # Examples:
   # user tungstenuser has password secret and uses 'sjc_nyc' composite
   # data service, but prefers nyc site for reading:
   #   tungstenuser secret sjc_nyc nyc
   ...
   ```

   Now add a `@script` directive to point to the encrypted file and certificate:

   ```
   @script openssl rsautl -decrypt -inkey /opt/continuent/share/usermap.pem -in /opt/continuent/share/user.map.entries.ssl
   ```

6. Repeat the process on each host. The `user.map` file will be copied to the new version when you upgrade Tungsten so this process must only be completed once per host.

### 6.6.7. Synchronizing `user.map` Data

Tungsten Clustering 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:

```
#!/bin/bash

for HOST in host1 host2 host3 host4
done
```

rsync /opt/continuent/tungsten/tungsten-connector/conf/user.map

```shell
SHELL=/opt/continuent/tungsten/tungsten-connector/conf/user.map
```
If @scripts 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.

6.6.8. 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.

6.6.9. 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.

Host-based authentication is not enabled in the default installation. To enable it, create a file authorized_hosts within the tungsten/tungsten-connector/conf/ directory of the active installation. The connector will then need to be restarted before host-based authentication is enabled.

Important

The authorized_hosts file is not automatically distributed during deployment and updates. The file must be manually copied to other hosts.

If the content of the authorized_hosts file is changed, the connector must be restarted before the changes take effect.

If the file exists, host-based authentication is enabled. If it is 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.

6.7. 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 6.8, "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:

```
cctrl> router * online
```

While in `AUTOMATIC` policy mode, connectors will automatically be placed online if they have entered the `OFFLINE` state automatically 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:

### 6.7.1. 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 6.7.3, "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.

### 6.7.2. Connections During Manual Switch

When a manual `switch` operation has been initiated, the Connector follows this sequence:

1. New connection attempts to the old 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 datasource.

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.

### 6.7.3. 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.
6.7.4. 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.

6.7.5. 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 autodetect.

When set to autodetect default, the values are automatically calculated by the connector computing suitable values based on the wait_timeout value configured in the MySQL server.

```
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> tpm update alpha --property=connection.keepAlive.interval=30
```

**Warning**

Please note that Connector Keepalive is not compatible with Bridge mode.

In Bridge mode, the client session is directly connected to the MySQL server at the TCP level, literally forwarding the client's packet to the server. This means that closing connections is the responsibility of the MySQL server based on the configured wait_timeout value, not the Connector.

6.7.6. Connector Change User as Ping

When using PHP with connection pooling enabled, the change user command is used to ping a connection within a pool to ensure that the connection is open and active before using it. The Tungsten Connector uses JDBC to connector to MySQL, which does not support the change user protocol option.

To provide an alternative to this for PHP applications communication through the connector, the connector can be configured to respond to the COM_CHANGE_USER command from a client application. Rather than performing the change user operation, instead the connector will respond to the client with an acknowledgement, emulating the ping operation.

This operation is disabled by default, and must be explicitly enabled. This can be achieved by setting the correct property value, treat.com.change.user.as.ping, to true during configuration with tpm:

```
shell> tpm configure alpha --property=treat.com.change.user.as.ping=true
```
6.8. 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 outline in Section 6.7, "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 [471]`) and 12000 (managed by `-mgr-rmi-remote-port [461]`). 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 failsafe mode called `On Hold [245]`. 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 [245]` 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 [245]` state until a manager can be reached.

6.9. Connector Command-Line Interface

The `connector` command is used for various operations that affect the Tungsten Connector, for example, starting and stopping the Tungsten Connector, getting status, updating and debugging.

When using the `connector` command-line tool, the following sub-commands are available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>client-list [303]</td>
<td>Return a list of the current client connections through this connector.</td>
</tr>
<tr>
<td>cluster-status [303]</td>
<td>Return the cluster status, as the connector currently understands it. This is the command-line alternative to the inline cluster status command.</td>
</tr>
<tr>
<td>condrestart [303]</td>
<td>Restart only if already running</td>
</tr>
<tr>
<td>console [303]</td>
<td>Launch in the current console (instead of a daemon)</td>
</tr>
<tr>
<td>dump [303]</td>
<td>Request a Java thread dump (if connector is running)</td>
</tr>
<tr>
<td>graceful-stop {seconds} [303]</td>
<td>Stops the connector gracefully, allowing outstanding open connections to finish and close before the connector process is stopped. (seconds) is an integer of zero or more, and is required. The connector will shut down immediately if there are no active connections. Specifying zero (0) will cause a shutdown without waiting for connections to terminate.</td>
</tr>
<tr>
<td>install [303]</td>
<td>Install the service to automatically start when the system boots</td>
</tr>
<tr>
<td>reconfigure [303]</td>
<td>Reconfigure the connector by forcing the connector to reread the configuration, including the configuration files and <code>user.map</code>.</td>
</tr>
<tr>
<td>remove [304]</td>
<td>Remove the service from starting during boot</td>
</tr>
<tr>
<td>restart [304]</td>
<td>Stop connector if already running and then start</td>
</tr>
<tr>
<td>start [304]</td>
<td>Start in the background as a daemon process</td>
</tr>
<tr>
<td>status [304]</td>
<td>Query the current status</td>
</tr>
<tr>
<td>stop [304]</td>
<td>Stop if running (whether as a daemon or in another console)</td>
</tr>
</tbody>
</table>

For more information, please see Section 8.9, "The connector Command".

248
6.10. 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.

Important
When using Bridge Mode, these commands will not be available because you are connected directly to the database server.

Table 6.3. Inline Interface Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tungsten cluster status</code></td>
<td>Displays a detailed view of the information the connector has about the cluster</td>
</tr>
<tr>
<td><code>tungsten connection count</code></td>
<td>Display the current number of active connection to each datasource</td>
</tr>
<tr>
<td><code>tungsten connection status</code></td>
<td>Displays information about the connection status for the last statement executed</td>
</tr>
<tr>
<td><code>tungsten flush privileges</code></td>
<td>Reload the user.map file and update the user credentials</td>
</tr>
<tr>
<td><code>tungsten gc</code></td>
<td>Executes the connector garbage collector to free memory</td>
</tr>
<tr>
<td><code>tungsten help</code></td>
<td>Shows help description each statement</td>
</tr>
<tr>
<td><code>tungsten mem info</code></td>
<td>Display the memory usage information for the connector</td>
</tr>
<tr>
<td><code>tungsten show processlist</code></td>
<td>List all active queries on this connector instance</td>
</tr>
<tr>
<td><code>tungsten show variables</code></td>
<td>Display the connector configuration options currently in use</td>
</tr>
</tbody>
</table>

6.10.1. Connector `tungsten cluster status` Command

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:

```
mysql> tungsten cluster status;
+-----------------+-------+-------+--------+---------+----------------+-----------------------+-------------------------+
| dataServiceName | name  | host  | role   | state   | appliedLatency | activeConnectionCount | connectionsCreatedCount |
+-----------------+-------+-------+--------+---------+----------------+-----------------------+-------------------------+
| alpha           | host1 | host1 | slave  | SHUNNED | 60.0           | 0                     | 0                       |
| alpha           | host2 | host2 | master | ONLINE  | 0.0            | 1                     | 2                       |
| alpha           | host3 | host3 | slave  | SHUNNED | 61.0           | 0                     | 0                       |
+-----------------+-------+-------+--------+---------+----------------+-----------------------+-------------------------+
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.

6.10.1.1. Connector connector cluster status on the Command-line

The tungsten cluster status is also available from the command-line through the connector command. To use this command from the command line, place the cluster-status command on the command-line. For example:

```
shell> connector cluster-status
Executing Tungsten Connector Service --cluster-status ...

+--------------+--------------------+-------------+--------------+--------+--------+------------------------------------------+----------------------+-----------------+------------------+--------------------+---------------------+
| Data service | Data service state | Data source | Is composite | Role   | State  | High water                               | Last shun reason     | Applied latency | Relative latency | Active connections | Connections created |
+--------------+--------------------+-------------+--------------+--------+--------+------------------------------------------+----------------------+-----------------+------------------+--------------------+---------------------+
| alpha        | ONLINE             | demo-c11    | false        | master | ONLINE | 11(mysql-bin.000033:0000000000000523;-1) |                      | 7528198.0       | 7528609.0        | 0                  | 0                   |
| alpha        | ONLINE             | demo-c12    | false        | slave  | ONLINE | 11(mysql-bin.000033:0000000000000523;-1) | SHUNNED-FOR-RECOVERY | 7528198.0       | 7528611.0        | 0                  | 0                   |
+--------------+--------------------+-------------+--------------+--------+--------+------------------------------------------+----------------------+-----------------+------------------+--------------------+---------------------+
```

Done Tungsten Connector Service --cluster-status

The information can also be output in JSON format by adding the -json:

```
shell> connector cluster-status -json
Executing Tungsten Connector Service --cluster-status -json...
{
  "alpha" : {
    "demo-c11" : {
      "name" : "demo-c11",
      "dataServiceName" : "alpha",
      "host" : "demo-c11",
      "activeConnectionCount" : 0,
      "alertMessage" : "",
      "alertStatus" : "OK",
      "alertTime" : 1496310617376,
      "appliedLatency" : 7528198.0,
      "available" : true,
      "callableStatementsCreated" : 0,
      "composite" : false,
      "compositeMember" : null,
      "connectionsCreated" : 0,
      "container" : false,
      "dataSourceHost" : "demo-c11",
      "dataSourceRole" : "master",
      "driver" : "org.drizzle.jdbc.DrizzleDriver",
      "enabled" : 0,
      "executable" : false,
      "isAvailable" : true,
      "key" : "demo-c11",
      "lastError" : "",
      "lastShunReason" : "",
      "lastUpdate" : 1511778569541,
      "master" : true,
      "masterConnectUri" : "",
      "precedence" : 99,
      "preparedStatementsCreated" : 0,
      "relativeLatency" : 7528879.0,
      "relay" : false,
      "role" : "master",
      "sequence" : {
        "wrapAround" : 9223372836854775807,
        "generation" : 0,
        "identity" : "b70818d0-6506-4fd1-8e23-c303ccc2bca",
        "currentValue" : 0,
        "lastValue" : 0,
        "wrapped" : false
      },
      "slave" : false,
      "standby" : false,
      "state" : "ONLINE",
      "statementsCreated" : 0,
      "type" : "DATASOURCE",
      "updateTimestamp" : 1511778569541,
      "url" : "jdbc:mysql:thin://demo-c11:3306/${DBNAME}?jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull&tinyInt1isBit=false&allowMultiQueries=true",
      "vendor" : "mysql",
      "vipAddress" : "",
      "vipInterface" : ""
    },
    "demo-c12" : {
      "name" : "demo-c12",
      "dataServiceName" : "alpha",
      "host" : "demo-c12",
      "activeConnectionCount" : 0,
      "alertMessage" : "",
      "alertStatus" : "OK",
      "alertTime" : 1496310617376,
      "appliedLatency" : 7528198.0,
      "available" : true,
      "callableStatementsCreated" : 0,
      "composite" : false,
      "compositeMember" : null,
      "connectionsCreated" : 0,
      "container" : false,
      "dataSourceHost" : "demo-c12",
      "dataSourceRole" : "slave",
      "driver" : "org.drizzle.jdbc.DrizzleDriver",
      "enabled" : 0,
      "executable" : false,
      "isAvailable" : true,
      "key" : "demo-c12",
      "lastError" : "",
      "lastShunReason" : "",
      "lastUpdate" : 1511778569541,
      "master" : false,
      "masterConnectUri" : "",
      "precedence" : 99,
      "preparedStatementsCreated" : 0,
      "relativeLatency" : 7528879.0,
      "relay" : false,
      "role" : "slave",
      "sequence" : {
        "wrapAround" : 9223372836854775807,
        "generation" : 0,
        "identity" : "b70818d0-6506-4fd1-8e23-c303ccc2bca",
        "currentValue" : 0,
        "lastValue" : 0,
        "wrapped" : false
      },
      "slave" : false,
      "standby" : false,
      "state" : "ONLINE",
      "statementsCreated" : 0,
      "type" : "DATASOURCE",
      "updateTimestamp" : 1511778569541,
      "url" : "jdbc:mysql:thin://demo-c12:3306/${DBNAME}?jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull&tinyInt1isBit=false&allowMultiQueries=true",
      "vendor" : "mysql",
      "vipAddress" : "",
      "vipInterface" : ""  
    }
  }
}
```
6.10.2. Connector tungsten connection count Command

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)
```

6.10.3. Connector tungsten connection status Command

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;
```
### 6.10.4. Connector tungsten flush privileges Command

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    |
+----------------------------------+
```

### 6.10.5. Connector tungsten gc Command

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(17025K)       |
|                       | committed = 68438664(66384K) max = 259522560(253440K) |
| Non-heap Memory       | init = 24313856(23744K) used = 13970024(13642K)       |
|                       | committed = 24313856(23744K) max = 224395264(219316K) |
| Thread Count          | 16                                                    |
+-----------------------+-------------------------------------------------------+
```

```
mysql> tungsten gc;
+-------------------------------+
| Message                       |
+-------------------------------+
| Garbage collection successful |
+-------------------------------+
```

```
mysql> tungsten mem info;
+-----------------------+-------------------------------------------------------+
| JVM Memory statistics | Value in bytes                                        |
+-----------------------+-------------------------------------------------------+
| Peak Thread Count     | 18                                                    |
| Heap Memory           | init = 67108864(65536K) used = 4110088(4013K)         |
|                       | committed = 64063676(62424K) max = 259522560(253440K) |
| Non-heap Memory       | init = 24313856(23744K) used = 13970024(13642K)       |
|                       | committed = 24313856(23744K) max = 224395264(219316K) |
| Thread Count          | 16                                                    |
+-----------------------+-------------------------------------------------------+
```

### 6.10.6. Connector tungsten help Command

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 in 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                         |
```
6.10.7. Connector tungsten mem info Command

```sql
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 = 259522560(253440K) |
| Non-heap Memory       | init = 24313856(23744K) used = 14227336(13893K)       |
|                       | committed = 24313856(23744K) max = 224395264(219136K) |
| Thread Count          | 18                                                    |
+-----------------------+-------------------------------------------------------+
```

4 rows in set (0.05 sec)

6.10.8. Connector tungsten show [full] processlist Command

```sql
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 |
+------------+--------+----------+---------------+----------------+---------+------+-------+------+
```

3 rows in set (0.05 sec)

6.10.9. Connector show slave status Command

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.

```sql
mysql> 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_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_Key:
```

3 rows in set (0.05 sec)
6.10.10. Connector tungsten show variables Command

The connector will intercept the command `tungsten show variables;` and display its current (loaded) configuration. The `variable Type` column indicates the file where the configuration item is stored.

**Important**

This command applies only to proxy mode [ie. not bridge mode].

Just as in the MySQL `show variables` command, Tungsten Connector supports filters through the "like string" and its wildcard ('%') sign.

```
mysql> tungsten show variables like '%timeout%';
+----------------------+----------------------------------------+----------+
<table>
<thead>
<tr>
<th>Variable_Type</th>
<th>Variable_name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>connector.properties</td>
<td>bridgeServerToClientForcedCloseTimeout</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>connection.close.idle.timeout</td>
<td>28800000</td>
</tr>
<tr>
<td></td>
<td>connection.keepAlive.timeout</td>
<td>20160</td>
</tr>
<tr>
<td></td>
<td>server.port.binding.timeout</td>
<td>60</td>
</tr>
<tr>
<td>router.properties</td>
<td>gatewayConnectTimeoutMs</td>
<td>5000</td>
</tr>
<tr>
<td></td>
<td>keepAliveTimeout</td>
<td>30000</td>
</tr>
<tr>
<td></td>
<td>readCommandRetryTimeoutMs</td>
<td>10000</td>
</tr>
<tr>
<td></td>
<td>waitForDisconnectTimeout</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>waitIfDisabledTimeout</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>waitIfUnavailableTimeout</td>
<td>0</td>
</tr>
</tbody>
</table>
+-----------------------+----------------------------------------+----------+
18 rows in set (0.01 sec)
```

6.11. Advanced Configuration

6.11.1. Using Multiple Dataservices

The connector is able to work with multiple dataservices. It may be a combination of master/slave or composite dataservices. The connector will communicate with managers in each dataservice and provide connectivity.

1. Configure the host as a connector for one of the dataservices. This will be the default dataservice for the connector. Any version upgrades for this cluster will also upgrade the connector. See Section 3.5, "Deploying Tungsten Connector Only" if you want the host to be fully independent.

2. Update the `dataservices.properties` file in `/opt/continuent/tungsten/cluster-home/conf`. Add a line for each new master/slave cluster the connector will connect to. Keep this file updated as you add and remove servers from each cluster.

   **Important**

   Do not list composite dataservices in this file. The connector will automatically discover those from the managers in each cluster.

3. Restart the connector. Any users connected to this connector will be disconnected at this time.

   `shell > connector restart`

4. Update the `user.map` to list new users for each new dataservice. See Section 6.6, "User Authentication" for more details. Specifically, the `user.map` may not include multiple users with the same name but different dataservices. Create unique users in each dataservice before updating `user.map`.

   6.0.3. The read affinity setting now supports multiple dataservices with ordering and exclusion (only one was previously allowed).

   You may now fine-tune the affinity by specifying an ordered list of dataservice names.

   - `affinity` is defined as the host (single cluster only) or dataservice (composite clusters) to prefer for reads when QOS `RO RELAXED` is in use.
     
     The affinity feature routes both reads and writes when using a Multimaster topology.

   - This can be handled per user in `user.map`, globally via the `tpm` option `connector-affinity [441]`, or via the connection string.
• The **affinity** string is an ordered list of dataservice names, separated by commas, where the first dataservice entry will be the one used by default. If the first dataservice in the list is not available, the connector will use the next one listed, and so forth.

• Dataservices not specified in the list, if any, will be used last and randomly.

• It is also possible to exclude one or more dataservices by adding a hyphen ("-”) in front of the dataservice name.

• The deployment used in the examples below consists of a composite dataservice **global**, with four member dataservices (i.e. one cluster per site): **east**, **west**, **north** and **south**.

Just for reference, this is the user definition syntax in the **user.map** file:

```
{User} {Password} {DataServiceName-or-CompositeDataServiceName} [Affinity]
```

For example, to exclude site **south** from servicing read request in **user.map**:

```
sales secret global east,west,north,-south
```

To do the same globally via the **tpm** command, use the following configuration setting:

```
connector-affinity=east,west,north,-south
```

The above affinity string "**east,west,north,-south**" will try **east**, then **west**, then **north**. If none of the first three are available, a connection request would not succeed since **south** has been excluded by the negation ("-“).

In the following **user.map** example, dataservices **north** and **south** would be available as random candidates if the first two (**east** and **west**) were unavailable:

```
sales secret global east,west
```

To do the same globally via the **tpm** command, use the following configuration setting:

```
connector-affinity=east,west
```

See Section 6.6.1, "**user.map** File Format" for more details.

In 6.0.4 and later, the effect of affinity during cluster changes has been updated to support reconnection when the correct datasource becomes available.

When a site goes offline, connections to this site will be forced closed. Those connections will reconnect, as long as the site stays offline, they will be connected to remote site.

You can now enable an option so that when the site comes back online, the connector will disconnect all these connections that couldn’t get to their preferred site so that they will then reconnect to the expected site with the appropriate affinity.

Note that this only applies to bridge mode. In proxy mode, relevancy of connected data source will be re-evaluated before every transaction.

When not enabled, connections will continue to use the server originally configured until they disconnect through normal attribution. This is the default option.

To enable forced reconnection, use the **--connector-reset-when-affinity-back=true** [435] option to **tpm**.

The connector is now ready to accept users for each of the new dataservices. Keep the **dataservices.properties** and **user.map** files updated to make sure the connector works properly.

### 6.11.2. Connector Automatic Reconnect

Automatic reconnect enables the Connector to re-establish a connection in the event of a transient failure. Under specific circumstances, the Connector will also retry the query.

1. **Connector automatic reconnect is enabled by default in Proxy and Smartscale modes. Use the **tpm** command option **--connector-autoreconnect=false** [442] to disable automatic reconnect.**

2. **This feature is not available while running in Bridge Mode. Use the **tpm** command option **--connector-bridge-mode=false** [442] to disable Bridge mode.**

3. **Automatic reconnect enables retries of statements under the following circumstances:**

   • not in bridge mode
   • not inside a transaction
Tungsten Connector

- no temp table has been created
- no lock acquired and not released
- the request is a read

To disable:

Show Staging

Show INI

```shell
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten
shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
shell> cd (STAGING_DIRECTORY)
```

Run the `tpm` command to update the software with the Staging-based configuration:

```shell
shell> ./tools/tpm configure alpha \       
    --connector-autoreconnect=false
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

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

```
[alpha]
...
Connector-autoreconnect=false
```

Run the `tpm` command to update the software with the INI-based configuration:

```shell
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
shell> cd (STAGING_DIRECTORY)
shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

Configuration group alpha

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

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

- `--connector-autoreconnect=false [442]

  connector-autoreconnect=False [442]

  Enable auto-reconnect in the connector

The autoreconnect status can be monitored within the `autoReconnect` parameter output by the `tungsten show variables` while connected to the Connector. For example:

```shell
shell> tpm connector
mysql> tungsten show variables like "autoReconnect";
+-----------------------+---------------+-------------------------+
| Variable_Type | Variable_name | Value |
The above output indicates that the autoreconnect feature is disabled. The `tungsten show` command is not available in Bridge mode.

### 6.11.3. Using the Connector 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.

There are two primary ways to monitor MySQL health via HAProxy:

- **mysql-check** - an haproxy-native test
  
  The check consists of sending two MySQL packets, one Client Authentication packet, and one QUIT packet, to correctly close the MySQL session. HAProxy then parses the MySQL Handshake Initialisation packet and/or Error packet. It is a basic but useful test which does not produce errors or aborted connects on the server. This solution requires adding a user to MySQL:

  ```
  INSERT INTO mysql.user (Host,User) values ('{ip_of_haproxy}','{username}');
  FLUSH PRIVILEGES;
  ```

  **Warning**
  
  This method does NOT check for database presence nor database consistency. To do this, we must use an external check script (via xinetd) which is explained in the next section.

- **A check script** - normally launched via xinetd, and allows for custom monitoring of the database health. This is the preferred method.

  **Note**
  
  Click here to see the HA Proxy documentation for more information

#### 6.11.3.1. Configuring HA Proxy using the native MySQL Check

A practical example for deploying the HAProxy's native `mysql-check` option:

```
INSERT INTO mysql.user (Host,User) values ('%', 'haproxy');
FLUSH PRIVILEGES;
```

#### 6.11.3.2. Configuring HA Proxy with a Check Script

A suitable MySQL check script configuration can be added to a basic HA Proxy installation using the following settings:

```
# backend
# listen connector
listen connector
bind *:3306
mode tcp
option tcpka  # enables keep-alive both on client and server side
balance roundrobin
option mysql-check user haproxy post-41
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 HAProxy, a check script needs to be installed on all hosts running Tungsten Connector that will respond to A HAProxy connector check script 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 HAProxy and will return the status of the connector to HAProxy 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`:

   ```
   #!/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)
   # "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"
     echo -e "Content-Type: text/plain\r"
     echo -e "MySQL is running.\n"
     exit 0
   }

   return_fail() {
     echo -e "HTTP/1.1 503 Service Unavailable\r"
     echo -e "Content-Type: text/plain\r"
     echo -e "MySQL is *down*.\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:

   ```
   shell> chown tungsten.tungsten /opt/continuent/share/connectorchk.sh
   shell> chmod 700 /opt/continuent/share/connectorchk.sh
   shell> chmod +x /opt/continuent/share/connectorchk.sh
   ```

4. Install `xinetd` and add the xinetd service. On RedHat/CentOS:

   ```
   shell> yum -y install xinetd telnet
   ```
On Debian/Ubuntu:

```
shell> apt-get install xinetd telnet
```

5. Add an entry for the connector check script to `/etc/services`:

```
shell> echo "connectorchk 9200/tcp" >> /etc/services
```

6. Add a configuration to `xinetd` by creating the file `/etc/xinetd.d/connectorchk` with the following content:

```
# default: off
# 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: text/plain
MySQL is running.
```

### 6.11.4. Using Fall-Back Bridge Mode

This feature will allow the Tungsten Connector to fall back to bridge mode if a user cannot be successfully authenticated through `user.map`.

The connector is able to employ a special fall-back bridge mode which allows for a hybrid configuration of both Proxy and Bridge modes. By default, the bridge mode fallback feature is disabled.

When `fallBackBridgeMode` is set to either RW_STRICT or RO_RELAXED, the Connector will first check the user.map file for an entry that matches the user name passed in the connection request. If a match is found in the user.map, the Connector will act in Proxy mode so the conversation with the client will be handled locally, and a new connection will be opened from the connector to the database server based on the normal Proxy mode routing rules. If the user name is not found in user.map, then the connector will act in Bridge mode, and the connection will be forwarded directly to the specified database server, either to the master [RW_STRICT] or to the slave [RO_RELAXED] for handling with no intercept, just a TCP-layer packet routing. There will be no query interpretation or analysis, and no auto-reconnect, just failover handling.

Fall-Back Bridge Mode is NOT compatible with MySQL 8 at this time.

MySQL 8 support was added in versions 5.4.0 and 6.1.0, which broke the Fall-Back Bridge mode functionality. When switching to Bridge mode, the calling client is unable to complete the handshake due to authentication flags, namely `CLIENT_DEPRECATE_EOF` and `CLIENT_CONNECT_ATTRS`.

For more information, see Section 6.5, "Using Bridge Mode", and Section 6.6, "User Authentication".

To enable Fall-Back Bridge Mode using the DB Master:

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

Show Staging

Show INI
Run the `tpm` command to update the software with the Staging-based configuration:

```
sh> /tools/tpm update
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

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

```ini
[alpha]
... property=fallbackBridgeMode=RW_STRICT
connector-bridge-node=false
```

Run the `tpm` command to update the software with the INI-based configuration:

```
sh> /tools/tpm update
```

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

To enable Fall-Back Bridge Mode using a DB Slave (if available):

**Show Staging**

**Show INI**

```
sh> /tools/tpm configure alpha \
   --property=fallbackBridgeMode=RO_RELAXED \
   --connector-bridge-node=false
```

```
[alpha]
... property=fallbackBridgeMode=RO_RELAXED
connector-bridge-node=false
```

**Warning**

Updating these values require a connector restart (via tpm update) for the changes to be recognized.

**Important**

To be consistent, Bridge mode should be disabled when fallbackBridgeMode is enabled. The `--connector-bridge-node` option to `tpm` must be set to `false`. A consistency check is performed when starting the connector.

### 6.11.4.1. Using Fall-Back SSL To Bridge Mode

SSL connections are by design unreadable until the handshake has been exchanged. Because of this, the MySQL user name in the request is not visible to the Connector immediately, and therefore the Connector is unable to check against user.map for `fallbackBridgeMode`.

Due to this situation, another feature was created to address SSL connections while the `fallbackBridgeMode` is enabled called `fallbackSSLToBridge`. When `fallbackSSLToBridge` is set to `true` [default], then all SSL connections will use Bridge mode, while non-SSL connections will use the fallback-
BridgeMode setting [i.e. RW_STRICT which routes traffic to the Master or RO_RELAXED which routes to the slaves]. When FallBackSSLToBridge is set to false, then SSL connections will run in non-Bridge mode - if the specified user doesn’t exist in user.map, an error will be raised.

Important

The FallBackSSLToBridge setting is ONLY available when FallBackBridgeMode is enabled, and is ignored when FallBack-BridgeMode is set to false.

Since FallBackSSLToBridge is enabled by default when FallBackBridgeMode is enabled, you may turn it off as follows:

Show Staging

Show INI

shell> ./tools/tpm configure alpha \
--property=FallBackSSLToBridge=false

交汇点

Warning

Updating these values require a connector restart (via tpm update) for the changes to be recognized.

6.11.5. Using the Max Connections Feature

This feature will allow the connector behavior to be changed based on the connection count. The connector is able to mimic MySQL’s max_connections. Depending on your needs, the connector can be configured to pile up or reject connections above this number. This is served by the following two tpm flags:

- connector-max-connections [443] - defines the maximum number of connections the connector should allow at any time.

When the connector-max-connections [443] is set to a non-zero numeric value, the connector denies access to the client in one of two ways: queue (default) or reject.

- connector-drop-after-max-connections [443] - defines how the connector should handle new connection requests - queue (default) or reject.

Enabling this option causes the connector to drop new connection requests when connector-max-connections [443] is reached by immediately sending a “Too Many Connections” error to the client, just like MySQL would.

Important

When a client connection request arrives at the connector, an object is created to track that client connection which uses a certain amount of memory.

The connector then checks the value of connector-max-connections [443] against the current connection count.

If the connection limit has been reached, the connector decides how to behave by checking the value of connector-drop-after-max-connections [443].

If connector-drop-after-max-connections [443] is false (the default), the connector will queue the connection request, but send nothing back to the client at all. This connection check will repeat after a delay. Once the connection count falls below connector-max-connections [443] an attempt to connect to a server is made. In this mode, connections will continue to pile up in memory as new requests are queued, resulting in an Out of Memory error. For this reason, the connector-drop-after-max-connections [443] is available to prevent connection queueing when the maximum number of connections has been reached.

If connector-drop-after-max-connections [443] is enabled, the connector will return a “Too Many Connections” error to the client and remove the client connection object, freeing memory.

To enable connector-max-connections [443]:

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

Show Staging

Show INI

shell> tpm query staging
Tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
Tungsten Connector

Run the `tpm` command to update the software with the Staging-based configuration:

```
shell> ./tools/tpm configure alpha \
     --connector-max-connections=2500
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

Show Staging

Show INI

To enable `connector-drop-after-max-connections` you must also set a non-zero value for `connector-max-connections`:

```
shell> /etc/tungsten/tungsten.ini
[alpha]
... 
connector-max-connections=2500
```

Configuration group `alpha`

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

- **--connector-drop-after-max-connections**

  `connector-drop-after-max-connections=true`

  Instantly drop connections that arrive after `connector-max-connections` has been reached

- **--connector-max-connections**

  `connector-max-connections=2500`

  The maximum number of connections the connector should allow at any time

**Important**

Updating these values require a connector restart (via `tpm update`) for the changes to be recognized.
To select a real-world value for `connector-max-connections` set the value to a value slightly lower than the MySQL value of `max_connections` to prevent the server from ever hitting maximum. You may use the following formula for a more complex calculation: 

\[
\text{connector-max-connections} = \left( \frac{\text{MySQL Master } \text{max_connections}}{\text{number of connectors}} \right) \times 0.95
\]

**Important**

When `connector-drop-after-max-connections` is enabled, be sure that your load balancers are configured to identify that max connections have been reached and to switch to another connector when that happens.

### 6.11.6. Adjusting the Client Disconnect Delay During Manual Switch

This feature controls how the connector handles existing connections when a manual switch is invoked.

When a graceful switch is invoked via `cctrl`, by default the Connector will wait for five (5) seconds to allow in-flight activities to complete before forcibly disconnecting all active connections from the application side, no matter what type of query was in use.

If connections still exist after the timeout interval, they are forced closed, and the application will get back an error.

**Important**

This setting ONLY applies to a manual switch. During a failover, there is no wait and all connections are force-closed immediately.

This timeout is adjusted via the `tpm` option `--connector-disconnect-timeout`.

For example, to change the delay to 10 seconds:

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

**Show Staging**

**Show INI**

Run the `tpm` command to update the software with the Staging-based configuration:

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

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

Run the `tpm` command to update the software with the INI-based configuration:

```
./tools/tpm update
```
For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

Configuration group alpha

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

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

- **--connector-disconnect-timeout=10**

  Time (in seconds) to wait for active connection to disconnect before forcing them closed [default: 5]

  **Warning**

  If you increase this value, you delay the manual switch! ONLY change this if you accept the fact that the manual switch process will last at least as long as this setting in seconds.

  Do not set this value to zero (0) or there will be no attempt to disconnect at all. If you wish to disable the wait entirely, set **--property=waitForDisconnect=false** in your configuration on the connectors and run `tpm update`.

  **Important**

  Updating these values require a connector restart (via tpm update) for the changes to be recognized.

  This value is reflected in the `waitForDisconnectTimeout` setting located in `cluster-home/conf/router.properties`.

### 6.11.7. Adjusting the Bridge Mode Forced Client Disconnect Timeout

This feature controls how long the Connector in Bridge mode waits before forcibly disconnecting the server side of the session after a client session ends.

Default: 50ms

When a client application opens a socket and connects to the connector, a second socket/connection to the server is created. The Connector in bridge mode then simply transfers data between these two sockets.

When a client application brutally closes a connection without following the proper disconnection protocol, the server will not know about that disconnect until the connector closes the Connector<>server socket. If the connector closes the Connector<>server connection too soon after a client disconnect, there is a chance that the proper disconnection messages will be missed, if sent late. If the connector does not close this connector<>server connection, it would stay open indefinitely, using memory and resources that would otherwise be reclaimed. The default of 500ms is very conservative and will fit most environments where client applications disconnect properly. When the volume of connections opened and never closed exceeds a certain level, the timeout must be tuned (lowered) to close idle connections faster, or the available resources will get used up.

Many times this situation is caused by health checks, especially from monitoring scripts and load balancers checking port liveness. Many of these check do not gracefully close the connection, triggering the need for tuning the Connector.

If connections still exist after the timeout interval, they are forced closed, and a warning will be printed in the connector logs (C>S ended. S>C streaming did not finish within bridgeServerToClientForcedCloseTimeout=500 [ms]. Will be closed anyway!).

**Important**

This setting ONLY applies to Bridge mode.

This timeout is adjusted via the `tpm` property **--property=bridgeServerToClientForcedCloseTimeout** in milliseconds.

For example, to change the delay to 50 milliseconds:

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

Show Staging

Show INI

```bash
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
cut -d: -f1 | cut -d@ -f1

shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten
```
Tungsten Connector

Run the `tpm` command to update the software with the Staging-based configuration:

```
shell> ./tools/tpm configure alpha \
    --property=bridgeServerToClient ForcedCloseTimeout=50
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

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

[alpha]
...  
property=bridgeServerToClient ForcedCloseTimeout=50

Run the `tpm` command to update the software with the INI-based configuration:

```
shell> tpm query staging  
    tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
shell> cd (STAGING_DIRECTORY)
```

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

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

**Warning**

- If you decrease this value, you run the risk of disconnecting valid but slow sessions.

**Important**

- Updating these values require a connector restart (via tpm update) for the changes to be recognized.

### 6.11.8. Adjusting the Connector Response to Resource Losses

This section describes how to control the Connector responses in the event of the loss of a required DataSource or all Managers.

#### 6.11.8.1. Adjusting the Connector Response to DataSource Loss

**Summary:** Whenever no master datasource is found, the Connector will reject connection requests.

This feature controls how long the Connector waits for the given type of DataSource to come ONLINE before forcibly disconnecting the client application.

By default, wait indefinitely for a resource to become available.

**Warning**

- Prior to software versions 5.3.2/6.0.1, the ONHOLD state would reject new connection attempts instead of pausing them. Also, `waitIfUnavailableTimeout` was ignored, and connection attempts were never severed after timeout.

There are two (2) parameters involved in this decision-making. They are:

- `waitIfUnavailable` [default: true]

  If `waitIfUnavailable` is true, then the Connector will wait for up to the time period specified by `waitIfUnavailableTimeout` to make a connection for a given QOS. If the timeout expires, the Connector will disconnect the client application [reject connection attempts and close ongoing connections].

  If `waitIfUnavailable` is false, the Connector will immediately disconnect the client with an error if a connection for a given QOS cannot be made immediately.
• \texttt{waitIfUnavailableTimeout} [default: 0, wait indefinitely]

If \texttt{waitIfUnavailable} is true, the Connector will wait for up to \texttt{waitIfUnavailableTimeout} number of seconds before disconnecting the client. If \texttt{waitIfUnavailable} is false, this parameter is ignored. If this parameter is set to zero (0) seconds, the Connector will wait indefinitely [client connection requests will hang forever].

For example, to immediately reject connections uponDatasource loss:

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

Show Staging

Show INI

\begin{verbatim}
shell> tpm query staging
stonwen@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten
shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
shell> ssh {STAGING_USER}@{STAGING_HOST}
shell> cd {STAGING_DIRECTORY}
shell> ./tools/tpm configure alpha \
        --property=waitIfUnavailable=false
\end{verbatim}

Run the \texttt{tpm} command to update the software with the Staging-based configuration:

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

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”:

\begin{verbatim}
shell> vi /etc/tungsten/tungsten.ini

[alpha]
...
    property=waitIfUnavailable=false
\end{verbatim}

Run the \texttt{tpm} command to update the software with the INI-based configuration:

\begin{verbatim}
shell> tpm query staging
stonwen@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
shell> cd (STAGING_DIRECTORY)
shell> ./tools/tpm update
\end{verbatim}

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

Warning

PLEASE NOTE: this will make switch and failover much less transparent to the application since the connections will error until the new master is elected and back online.

Important

Updating these values require a connector restart (via tpm update) for the changes to be recognized.

These entries will NOT work if placed into \texttt{[defaults]}, each service must be handled individually.

6.11.8.2. Adjusting the Connector Response to Manager Loss

Summary: Whenever the Connector loses sight of the managers for a given data service, it will either suspend or reject new connection requests.

By default, suspend requests indefinitely until Manager communications are re-established.
This feature controls how long the Connector waits during a manager loss event to either suspend or reject the client connection.

Here is the decision chain and associated settings for what happens when the connector loses sight of the managers:

1. Delay for the value of `delayBeforeOnHoldIfNoManager` seconds which is 0/no delay by default.

2. Change state to **ON-HOLD** and begin the countdown timer starting from the `delayBeforeOfflineIfNoManager` value.

   In the **ON-HOLD** state, the connector will hang all new connections and allow existing connections to continue.

3. When the `delayBeforeOfflineIfNoManager` timer expires (30 seconds by default), change state to **OFFLINE**.

   Once **OFFLINE**, the Connector will break existing connections because there is no authoritative Manager node from the Connector's perspective. Without a Manager link, any change to the cluster configuration will remain invisible to the Connector, potentially leading to writes on a slave node.

   By default, all new connection requests will hang in the **OFFLINE** state. If `waitIfDisabled` is set to `false`, then the Connector will instead reject all new connections.

There are multiple parameters involved in this decision-making. They are:

- `delayBeforeOnHoldIfNoManager` [in seconds, default: 0, i.e. no delay]

  When the connector loses sight of the managers, delay before going **ON-HOLD** for the value of `delayBeforeOnHoldIfNoManager` seconds, which is 0/no delay by default.

- `delayBeforeOfflineIfNoManager` [in seconds, default: 30]

  Once **ON-HOLD**, delay before going **OFFLINE** for the value of `delayBeforeOfflineIfNoManager` seconds, 30 by default.

- `waitIfDisabled` [default: true]

  If the Dataservice is **OFFLINE** because it is unable to communicate with any Manager, the `waitIfDisabled` parameter determines whether to suspend connection requests or to reject them. If `waitIfDisabled` is set to `true` [the default], then the Connector will wait indefinitely for manager communications to be re-established. If `waitIfDisabled` is set to `false`, the Connector will return an error immediately.

To check for data service state, use the tungsten-connector/bin/connector cluster-status command. For example:

```
shell> connector cluster-status
Executing Tungsten Connector Service --cluster-status ... 
+--------------+--------------------+-------------+--------------+--------+--------+--------------------------------------+------------------+-----------------+------------------+--------------------+---------------------+
| Data service | Data service state | Data source | Is composite | Role   | State  | High water                           | Last shun reason | Applied latency | Relative latency | Active connections | Connections created |
+--------------+--------------------+-------------+--------------+--------+--------+--------------------------------------+------------------+-----------------+------------------+--------------------+---------------------+
| europe       | OFFLINE            | c1          | false        | master | ONLINE | 0(c1-bin.000002:0000000000000510;-1) | MANUALLY-SHUNNED | 0.0             | 5193.0           | 0                  | 3                   |
| europe       | OFFLINE            | c2          | false        | slave  | ONLINE | 0(c1-bin.000002:0000000000000510;-1) |                  | 1.0             | 5190.0           | 0                  | 0                   |
| europe       | OFFLINE            | c3          | false        | slave  | ONLINE | 0(c1-bin.000002:0000000000000510;-1) |                  | 2.0             | 5190.0           | 0                  | 1                   |
+--------------+--------------------+-------------+--------------+--------+--------+--------------------------------------+------------------+-----------------+------------------+--------------------+---------------------+
```

For more information, see Connector On-Hold State [245].

For example, to decrease the **ON-HOLD** time to 15 seconds:

```
shell> tpm query staging
```

```
shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten
```

```
shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1
```

```
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
shell> ssh {STAGING_USER}@{STAGING_HOST}
```

```
shell> cd {STAGING_DIRECTORY}
```

```
shell> ./tools/tpm configure alpha 
    --property=delayBeforeOfflineIfNoManager=15
```

Run the `tpm` command to update the software with the Staging-based configuration:
For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

Run the tpm command to update the software with the INI-based configuration:

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

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

For example, to immediately reject connections upon Manager loss:

Show Staging

Show INI

```
shell> .tools/tpm configure alpha \
  --property=waitIfDisabled=false
```

Warning

PLEASE NOTE: this will make switch and failover much less transparent to the application since the connections will error until communications with at least one manager has been established and the Connector is back online.

Important

Updating these values require a connector restart (via tpm update) for the changes to be recognized.

These entries will NOT work if placed into [defaults], each service must be updated individually.

### 6.11.9. Connector Logging Configuration

The connector is able to modify what is logged using different configuration options.

Informational messages like those below indicate that a client-side session did not properly close connection to the connector using a regular mysql close() call.

```
INFO [MySQLBridge] - [10.3.1.240:35686] Server>Client thread didn't finish cleanly - return code was: 1
```

Load balancers (especially physical ones) tend to do this when testing the connector, so these messages are expected when behind a load-balancer. Since the same error could come from buggy application code, the default is to log these circumstances.

If you know where the disconnects are coming from, you can safely disable this feature by setting the tpm option --connector-disable-connection-warnings [442] to true.

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

Show Staging

Show INI

```
shell> tpm query staging
```
Run the `tpm` command to update the software with the Staging-based configuration:

```
shell> ./tools/tpm configure alpha \
    --connector-disable-connection-warnings=true
```

Run the `tpm` command to update the software with the INI-based configuration:

```
shell> tpm query staging
```

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

Configuration group `alpha`

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

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

- `--connector-disable-connection-warnings=true [442]`

  `connector-disable-connection-warnings=true [442]`

  Hide Connector warnings in log files

  **Warning**

  Updating these values require a connector restart (via `tpm update`) for the changes to be recognized.

### 6.11.9.1. Connector Logging to Syslog

The Connector is able to send log information to Syslog.

You can control the full connector log by editing the `tungsten-connector/conf/wrapper.conf` file.

Depending on what you want to do, you can:

- Send full connector log to the syslog:
  
  ```
  wrapper.syslog.loglevel=NONE =>
  wrapper.syslog.loglevel=INFO
  ```

- Stop sending log to `connector.log`:
  
  ```
  wrapper.logfile.loglevel=INFO =>
  wrapper.logfile.loglevel=NONE
  ```

- Change the log file path:
wrapper.logfile=../../tungsten-connector/log/connector.log

Note

For logging purposes, our software uses both the Tanuki wrapper and Apache’s Log4J tool.

Valid log levels for the Tanuki wrapper include:

- NONE for no output
- FATAL to only show fatal error messages
- ERROR to show all error messages
- WARN to show all warning messages
- STATUS to show all state changes
- INFO shows all JVM output and informative messages
- DEBUG shows detailed debug information

Valid log levels for Apache's log4j include: TRACE, DEBUG, INFO, WARN, ERROR and FATAL

Links to the full documentation for each are below:


6.11.10. Connector SSL Advertisement Configuration

When SSL is enabled, the Connector automatically advertises the ports and itself as SSL capable. With some clients, this triggers them to use SSL even if SSL has not been configured. This causes the connections to fail and not operate correctly.

You can safely disable SSL advertisement in the Connector by setting the `tpm` option `connector-ssl-capable` to `false`:

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

Show Staging

Show INI

Run the `tpm` command to update the software with the Staging-based configuration:

Run the `tpm` command to update the software with the INI-based configuration:
For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

Configuration group alpha

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

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

- --connector-ssl-capable=false [444]

When SSL is enabled, the Connector automatically advertises the ports and itself as SSL capable. With some clients, this triggers them to use SSL even if SSL has not been configured. This causes the connections to fail and not operate correctly.

Warning

Updating these values require a connector restart (via tpm update) for the changes to be recognized.

### 6.11.11. Connector IP Address Configuration

When the Connector starts up, it binds to all local IP addresses on the selected port using the wildcard syntax, for example: 0.0.0.0:3306.

You can force the Connector to bind to a specific IP address by setting the tpm option property=server.listen.address={IP_ADDRESS} [407]

If you wished to bind to the localhost IP address only, perhaps for security reasons, here is an example:

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

Show Staging

Show INI

Run the tpm command to update the software with the Staging-based configuration:

Run the tpm command to update the software with the INI-based configuration:
The staging DIRECTORY is `/opt/continuent/software/tungsten-clustering-6.0.5-41`.

```
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
```

```
shell> cd {STAGING_DIRECTORY}
```

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

For information about making updates when using an INI file, please see Section 9.4.4, "Configuration Changes with an INI file".

---

**Warning**

Updating these values require a connector restart (via `tpm update`) for the changes to be recognized.

---

### 6.12. 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 7. Tungsten Manager

The Tungsten Manager provides the management and monitoring of the Tungsten Clustering services to ensure that datasources, connectors and other components are running, datasources are replicating to each other, and handles failover and maintenance schedules.

7.1. Tungsten Manager Introduction

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
- Checks to determine the availability of datasources by using either the system ping protocol (default), or using the Echo TCP/IP protocol on port 7 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 B.2.2.3, “Host Availability Checks”.
- 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.

In order to be able to avoid split brain, a cluster needs an odd number of members such that if there is a network partition, there’s always a chance that a majority of the members are in one of the network partitions. If there is not a majority, it’s not possible to establish a quorum and the partition with the master, and no majority, will end up with a shunned master until such time a quorum is established.

To operate with an even number of database nodes, a witness node is required, preferably an active witness, since the dynamics of establishing a quorum are more likely to succeed with an active witness than with a passive witness.

7.2. Tungsten Manager Failover Behavior

The Tungsten Clustering default position is to protect the integrity of the data. Sometimes this means that a failover may be delayed. It is more important to have all the data than for the write master to be available. This behavior is configurable.

When the Manager quorum invokes a failover automatically, there are various states that each replication stream may be in that must be taken into account by the Manager layer as it coordinates the failover.

This is due to the loosely-coupled, asynchronous nature of Tungsten Replicator.

7.2.1. Failover Replication State Scenarios

The multiple scenarios regarding the state of the data are described below, along with how to locate any associated events:

- First, the normal situation - all events in the binary logs have been extracted into the master THL, all master THL has been transferred to all slave nodes, and all THL on the slave nodes has been fully applied to all slave databases.
- Scenario 1 - There are orphaned MySQL binary log events on the old master that were not extracted into THL before a failover occurred
The `tungsten_find_orphaned` script can help locate orphaned binary log events that did not make it to the THL on the old master before a failover. For more information, please see Scenario 1 in Section 8.24, "The `tungsten_find_orphaned` Command".

- Scenario 2 - There is orphaned THL left on the old master that did not make it to the new master

The `tungsten_find_orphaned` script can help locate orphaned master THL left on the old master that did not make it to the new master before a failover. For more information, please see Scenario 2 in Section 8.24, "The `tungsten_find_orphaned` Command".

- Scenario 3 - There is orphaned slave THL on the new master node that did not get applied into the database before getting promoted to master role
7.2.2. Recovery Behavior After Failover

The Tungsten Manager is able to detect un-extracted, desirable binary log events when recovering the old master via cctrl after a failover. The cctrl recover command will fail if:

- any unextracted binlog events exist on the old master that we are trying to recover (Scenario 1)
- the old master THL contains more events than the slave that will be promoted to a new master (Scenario 2)

In this case, the cctrl recover command will display text similar to the following:

Recovery failed because the failed master has unextracted events in the binlog. Please run the tungsten_find_orphaned script to inspect this events. Provided you have a recent backup available, you can try to restore the data source by issuing the following command:

datasource {hostname} restore

Please consult the user manual at:
https://docs.continuent.com/tungsten-clustering-6.1/operations-restore.html

The tungsten_find_orphaned script is designed to locate orphaned events. For more information, please see Section 8.24, "The tungsten_find_orphaned Command".

7.2.3. Failover Response when MySQL Server Fails

The Manager and Replicator behave in concert when MySQL dies on the master node. When this happens, the replicator is unable to update the trep_commit_seqno table any longer, and therefore must either abort extraction or continue extracting without recording the extracted position into the database.

By default, the Manager will delay failover until all remaining events have been extracted from the binary logs on the failing master node as a way to protect data integrity.

This behavior is configurable via the following setting, shown with the default of false:

```
--property=replicator.store.thl.stopOnDBError=false
```

Failover will only continue once:

- all available events are completely read from the binary logs on the master node
- all events have reached the slaves

When `--property=replicator.store.thl.stopOnDBError=true [407]`, then the Replicator will stop extracting once it is unable to update the trep_commit_seqno table in MySQL, and the Manager will perform the failover without waiting, at the risk of possible data loss due to leaving binlog events behind. All such situations are logged.
For use cases where failover speed is more important than data accuracy, those NOT willing to wait for long failover can set `replicator.store.thl.stopOnDBError=true` and still use `tungsten_find_orphaned` to manually analyze and perform the data recovery. For more information, please see Section 8.24, “The `tungsten_find_orphaned` Command”.

7.2.4. Failover Response when Slave Applier is Latent

During a failover, the manager will wait until the slave that is the candidate for promotion to master has applied all stored THL events before promoting that node to master.

This wait time can be configured via the `manager.failover.thl.apply.wait.timeout=0` property.

The default value is 0, which means "wait indefinitely until all stored THL events are applied".

Any value other than zero invites data loss due to the fact that once the slave is promoted to master, any unapplied stored events in the THL will be ignored, and therefore lost.

Whenever a failover occurs, the slave with most events stored in the local THL is selected so that when the events are eventually applied, the data is as close to the original master as possible with the least number of events missed.

That is usually, but not always, the most up-to-date slave, which is the one with the most events applied.

There should be a good balance between the value for `manager.failover.thl.apply.wait.timeout` and the value for `policy.slave.promotion.latency.threshold=900`, which is the "maximum slave latency" - this means the number of seconds to which a slave must be current with the master in order to qualify as a candidate for failover. The default is 15 minutes (900 seconds).

7.3. Tungsten Manager Failover Tuning

There are currently three discrete faults that can cause a failover of a master:

- **Database server failure** - failover will occur 20 seconds after the initial detection.
  - `--property=policy.liveness.dbping.fail.threshold=1` [407]

  The Tungsten Manager is unable to connect to the database server and gets an i/o error. If the database cannot respond to a tcp connect request after the configured number of attempts, the database server is flagged as STOPPED which initiates the failover.

  This would mean, literally, that the process for the database server is gone and cannot respond to a tcp connect request. In this case, by default, the manager will try two more times, once every 10 seconds, after the initial i/o error is detected and after the then 30 second interval has elapsed, will flag the database server as being in the STOPPED state and this, in turn, initiates the failover.

- **Host failure** - failover will occur 30 seconds after the initial detection
  - `--property=policy.liveness.hostPing.fail.threshold=2` [407]

  The host on which the master database server is running is 'gone'. The first indication that the master host is gone could be because the manager on that host no longer appears in the group of managers, one of which runs on each database server host. It could also be that the managers on the hosts besides the master do not see a 'heartbeat' message from the master manager. In a variety of circumstances like this, both of the managers will, over a 60 second interval of time, once every 10 seconds, attempt to establish, definitively, that the master host is indeed either gone or completely unreachable via the network. If this is established, the remaining managers in the group will establish a quorum and the coordinator of that group will initiate failover.

- **A replicator failure**, if `--property=policy.fence.masterReplicator=true` [407] is set to `true`, will cause a failover 70 seconds after initial detection
  - `--property=policy.fence.masterReplicator.threshold=6` [407]

  Depending on how you have the manager configured, a master replicator failure can also start a process of initiating a failover. There's a specific manager property `--property=policy.fence.masterReplicator` [407] that tells a manager to 'fence' a master replicator that goes into either a failed or stopped state. The manager will then try to recover the master replicator to an online state and, again, after an interval of 60 seconds, if the master replicator does not recover, a failover will be initiated. BY DEFAULT, THIS BEHAVIOR IS TURNED OFF. Most customers prefer to keep a fully functional master running, even if replication fails, rather than have a failover occur.

Important

The interval of time from the first detection of a fault until a failover occurs is configurable over 10 second intervals. The formula for determining the listed default failover intervals is based on the value of 'threshold' in the properties file `tungsten-manager/conf/manager.properties`: Interval = [threshold + 1] * 10 seconds

Additionally, there are multiple ways to influence the behavior of the cluster AFTER a failover has been invoked. Below are some of the key variables:

- **Behavior when MySQL is not available but the binary logs are** - wait for the Replicator to finish extracting the binary logs or not?
The Manager and Replicator behave in concert when MySQL dies on the master node. When this happens, the replicator is unable to update the `trep_commit_seqno` table any longer, and therefore must either abort extraction or continue extracting without recording the extracted position into the database.

The default of `false` means that the Manager will delay failover until all remaining events have been extracted from the binary logs on the failing master node as a way to protect data integrity.

Failover will only continue once:

- all available events are completely read from the binary logs on the master node
- all events have reached the slaves

When `--property=replicator.store.thl.stopOnDBError=true`, then the Replicator will stop extracting once it is unable to update the `trep_commit_seqno` table in MySQL, and the Manager will perform the failover without waiting, at the risk of possible data loss due to leaving binlog events behind. All such situations are logged.

For use cases where failover speed is more important than data accuracy, those NOT willing to wait for long failover can set `replicator.store.thl.stopOnDBError=true` and still use `tungsten_find_orphaned` to manually analyze and perform the data recovery. For more information, please see Section 8.24, “The `tungsten_find_orphaned` Command”.

- Slave THL apply wait time before failover - how long to wait term of seconds for a slave to finish applying all stored THL to the database before failing over to it.

  `--property=manager.failover.thl.apply.wait.timeout=0`

During a failover, the manager will wait until the slave that is the candidate for promotion to master has applied all stored THL events before promoting that node to master.

The default value is 0, which means "wait indefinitely until all stored THL events are applied".

**Warning**

Any value other than zero [0] invites data loss due to the fact that once the slave is promoted to master, any unapplied stored events in the THL will be ignored, and therefore lost.

Whenever a failover occurs, the slave with most events stored in the local THL is selected so that when the events are eventually applied, the data is as close to the original master as possible with the least number of events missed.

That is usually, but not always, the most up-to-date slave, which is the one with the most events applied.

- Slave latency check - how far behind in term of seconds is each slave? If too far behind, do not use for failover.

  `--property=policy.slave.promotion.latency.threshold=900`

The `policy.slave.promotion.latency.threshold` option is the "maximum slave latency" - this means the number of seconds to which a slave must be current with the master in order to qualify as a candidate for failover. The default is 15 minutes [900 seconds].

7.4. Tungsten Manager Fault Detection, Fencing and Recovery

The information contained in this topic is meant to give a relatively comprehensive understanding of how the Tungsten Manager detects faults and the subsequent processing that leads to fencing the fail and possible recovery from faults. The main focus of this topic is the set of business rules, implemented in the Tungsten Manager, which, collectively, perform fault detection, fault fencing, and fault recovery.

7.4.1. Tungsten Manager Definitions

These definitions assume a familiarity with concepts like failover, switch, master and slave datasource etc.

- **coordinator** — every cluster designates one of the Tungsten managers in the cluster as the coordinator and it is this manager that will be responsible for taking action, if action is required, to recover the cluster's database resources to the most highly available state possible.

- **rules** — this term specifically refers to a set of 'business rules', implemented in a format required by the 'JBoss Drools' rules engine, and which are used to perform fault detection, fencing and recovery for Tungsten Clustering.

- **rule firing/triggering** — refers to the action of a rule becoming active due to the coincidence of one or more conditions specified in the rule itself. For example, a rule that detects a potential missing cluster member, called a 'heartbeat gap detection' rule, can fire if there are
no other active alarms and if the rule has not ‘seen’ a cluster member heartbeat within the last 30-45 seconds. If a rule ‘fires’, further pro-
cessing, specified in Java, will take effect.

• fault/fault detection — any condition which, if left unresolved, could lead to a lack of availability of database resources or to data inconsis-
tency etc. Faults are detected. An example of a fault detection is to detect that a specific database server has stopped.

• (fault) alarm — the Tungsten Manager uses raises and processes entities that we will refer to as ‘alarms’ as an initial part of fault detection.
A set of manager rules raise alarms under specific circumstances and that alarm stays active or triggers further processing depending on
other rules. An alarm, depending on its type, may not necessarily mean that a fault has been definitively detected but that something has
been detected that may or may not lead to an actual fault condition. An example of this, as you will see later, is a HeartbeatGapAlarm which
occurs when the rules on a specific manager detect that a heartbeat event has not been received from one or more of the other man-
agers in the group.

• (fault) alarm retraction — the action taken, by the rules, to remove an alarm from from consideration for further action. For example, if the
manager raises a ‘Heartbeat Gap Alarm’ and then, subsequently, detects that the heartbeat from the errant member has resumed, a rule
will retract that alarm.

• (fault) fence/fencing — the action which leads to a first-level amelioration of a fault i.e. an action that keeps a fault from causing further
harm - applications are isolated from the fault through the action of fencing. Fencing a fault results in the removal of the original fault con-
dition but may also result in further recovery actions. An example of fencing the fault that is detected when a database server has stopped
is to set the associated datasource to the FAILED state. This effectively makes the datasource unavailable to applications, immediately, thus
isolating the application from the fault state.

• (fault) recover/recovery — the action which may occur after a fault is initially fenced and which leads to a condition of continued availabil-
ity, data consistency etc. An example of a recovery operation is, for example, the failover that occurs after a stopped master database fault
has been fenced.

• split-brain — a condition of a cluster such that members of the cluster group have different views of the same set of database resources
and, in the most damaging incidence of split-brain, more than different cluster members designate different database resources as the mas-
ter resource, resulting in applications being able, for example, to perform database updates on a database resource that should be a slave.
This condition is to be avoided at all costs since the result is data loss or data corruption.

7.4.2. Cluster Monitoring and Notification Events

The key fact that must be understood in order that the operation of the rules can be grasped is that the rules are primarily driven by notification
events which are, in turn, generated by the cluster monitoring subsystem or, in some cases, by threads internal to the manager that act
as monitors for specific conditions.

As a general rule, notifications are generated in the monitoring subsystem, on each host, independently of the other hosts, and sent, by each
manager, to all of the managers in the group, including itself, via group communications and the notifications are then routed to the rules en-
gine in each manager.

These notifications are sent via a messaging protocol that guarantees that the monitoring events are received by all managers in the group in
exactly the same order. This ordering is a critical part of correct rules operation for if notifications are received in a different order and used
to drive the rules evaluation, the rules on each manager could arrive at a different state/conclusion based on the ordering.

The cluster monitoring subsystem driven by a set of checker threads that are configured from files found in tungsten-manager/conf and are
named checker*.properties. There are currently four of these checkers, with their corresponding checker configurations stored in the follow-
ing tungsten-manager/conf/checker* properties files:

• checker.heartbeat.properties - runs a thread the generates a manager heartbeat notification for the manager. If a manager goes away
from the group, either by crashing, being stopped, or having a network outage, other managers won’t ‘see’ that manager’s heartbeat and
will use that as a clue that there may be something amiss on that manager’s host.

• checker.instrumentation.properties - TBD

• checker.mysqlserver.properties - polls the local mysql server for liveness/state. This checker attempts to establish a connection with the lo-
cal MySQL server and to execute the query that is found in tungsten-manager/conf/mysql_checker_query.sql. The checker then evaluates the suc-
cess or failure of the connect attempt and subsequent query execution and establishes the state of the MySQL server based on that eval-
uation. Because database server state is so critical to the operation of the cluster, particularly when it comes to availability of database re-
sources, this particular checker uses a very fine-grained and configurable process for evaluating the state.

• checker.tungstenreplicator.properties - polls the local Tungsten replicator for liveness/state. This checker connects to the local replicator
via the JMX interfaces and queries the replicator for its current state. If, in the process of connecting to the replicator, a connection cannot
be established because the replicator is not running, the checker simply returns the STOPPED state.

Let’s look at the contents of one of these files - checker.mysqlserver.properties:
Tungsten Manager

```java
requiresProxy=true
name=mysql_response
class=com.continuent.tungsten.monitor.checkers.JDBCMySQLDatabaseServerChecker
# delay between each monitoring run - default 3000ms
frequency=3000
# connection will be renewed after this period
reconnectAfter=30000
serverName=viveka
host=viveka
vendor=mysql
port=3306
driver=org.drizzle.jdbc.DrizzleDriver
url=jdbc:mysql:thin://viveka:3306/tungsten
username=tungsten
password=passw0rd
query=select 1
queryTimeout=5
queryFileName=mysql_checker_query.sql
```

The key thing to understand about this configuration file is that the value for the class property indicates which Java class the manager will load and that class, when alive, will then be configured with the additional properties that can be seen in this file.

Every checker will have a property frequency which indicates how often the checker thread will become active.

Then, since this checker is for MySQL server, you can see that the configuration file indicates which jdbc driver to use, which port to use to connect directly to MySQL, username, password etc. for the connection as well as the name of a file that determines what SQL the checker will run to check for the MySQL server liveness.

7.4.3. Rule Organization - Detection, Investigation, Fencing, Recovery

The main focus of manager operations are fault processing business rules, and since these rules are organized into categories based on their function. As you can infer from some of the previous definitions, there are four major categories of rules, with some ancillary 'housekeeping' categories. The four major categories are:

1. Fault Detection — raise alarms for specific or nascent faults.
   
   The text below comes directly from the source code for the manager rules and comprise the major faults detected by the manager:
   
   • **0600: DETECT MEMBER HEARTBEAT GAP**
     
     This rule fires if there are not already other alarm types pending, for a specific member, and if at least 30-45 seconds has elapsed since the last time a given manager send a ClusterMemberHeartbeat event to the group of managers. The result of this rule firing will be that a MemberHeartbeatGapAlarm is raised as well as a MembershipInvalidAlarm. Both of these alarms trigger other rules, explained later, which do further investigations of the current cluster connectivity and membership.
   
   • **0601: DETECT STOPPED CLUSTER MANAGER**
     
     This rule fires after a MemberHeartbeatGapAlarm has been raised since the reason for a heartbeat gap can be that a manager has stopped. Further processing determines whether or not the manager is, indeed, stopped. This rule generates a ManagerStoppedAlarm if it determines that the manager in question is, in fact, stopped.
   
   • **0602: DETECT DATASERVER FAULT**
     
     This rule fires whenever monitoring detects that a given database server is not in the ONLINE state. The result of this rule firing is a DataServerFaultAlarm which, in turn, results in further investigation via one of the investigation rules described later.
   
   • **0604: DETECT UNREACHABLE REMOTE SERVICE**
     
     In the case of a composite cluster, the current coordinator on each site will connect to one manager on the remote site. After establishing this connection, the local manager will poll the remote manager for liveness and generate a RemoteServiceHeartbeatNotification that has a status of REACHABLE. If the remote manager is not reachable, the manager that is polling the remote service will generate a RemoteServiceHeartbeatNotification that has a resource state of UNREACHABLE in which case this rule triggers and a RemoteDataServiceUnreachableAlarm is raised by this rule.
   
   • **0606: DETECT REPLICATOR FAULT**
     
     As the name of this rule indicates, it will detect a replicator fault. This rule triggers if it sees a replicator notification that indicates that any of the replicators in the cluster are in a STOPPED, SUSPECT or OFFLINE state. The result of triggering of this rule is at a ReplicatorFaultAlarm raised.

2. Fault Investigation - process alarms by iteratively investigating whether or not the alarm represents a true fault that requires further action.
• **0525: INVESTIGATE MY LIVENESS**

In this rule title the word MY refers to the manager that is currently evaluating the rule. This rule triggers when it sees a MemberHeartbeatGapAlarm and, as a result of triggering, the manager checks to see if it has both network connectivity as well as visibility of the other cluster members including, if necessary, visibility of a passive witness host. If, during this connectivity check, a manager determines that it is isolated from the rest of the cluster, it will restart itself in a failsafe mode meaning that it will shun all of its database resources and then attempt to join an existing cluster group as a part of a quorum. This rule is particularly important in cases where there are transient or even protracted network outages since it forms a part of the strategy used to avoid split-brain operations. If the manager, after restarting, is able to become part of a cluster quorum group, the process of joining that group will result in shunned resources becoming available again if appropriate.

• **0530: INVESTIGATE MEMBERSHIP VALIDITY**

This rule is triggered when it sees a MembershipInvalidAlarm which was previously generated as the result of a member heartbeat gap. The previous rule, INVESTIGATE MY LIVENESS checks for network connectivity for the current manager. This rule checks to see if the current manager is a part of a cluster quorum group. This type of check implies a connectivity check as well but goes further to see if other managers in the group are alive and operational. This rule is another critical part of split brain avoidance since, depending on what it determines, it will take one of the following actions:

• If the manager does not have network connectivity after checking, every 10 seconds, for a period of 60 seconds, it will restart itself in two different modes:
  a. If the manager detects that it is the last man standing, meaning that it is currently responsible for the master datasource and all of the other cluster members had previously stopped, it will restart normally, leaving the master datasource available, and will be prepared to be the leader of any new group of managers.
  b. If the manager is not the last man standing, it will restart in failsafe mode i.e. will restart with all of its resources shunned and will attempt to join an existing group.

• If the manager has network connectivity i.e. can see all of the other hosts in the cluster, it then checks to see if it is a part of a primary partition i.e. a cluster quorum group. If, the first time it checks, it determines that it is not a part of a primary partition, it immediately suspends all new database connection requests until such time that the manager determines that it is in a primary partition. This is, again, a critical part of avoiding split brain operation since it makes it impossible for connectors to satisfy new connection requests until a valid cluster quorum can be definitively validated.

The manager will then keep doing this check for quorum group membership, every 10 seconds, for a period of 60 seconds. If it determines that it is not a member of a quorum group, it will use the same criteria, as mentioned previously in the network connectivity case, to determine how it shall restart i.e.:

  a. If it is the last man standing, it will, as in the above case, restart normally, leaving the master datasource available.
  b. If the manager is not the last man standing, it will restart in failsafe mode i.e. will restart with all of its resources shunned and will attempt to join an existing group.

• If, after all of the previous checks, the manager establishes that it is a part of a quorum group, it will, if necessary because it disconnected Tungsten connectors during quorum validation, it will become available for Tungsten connectors to connect to it again after synchronizing its view of the cluster with the current cluster coordinator, and will then continue normal operations.

• **0550: INVESTIGATE: TIME KEEPER FOR HEARBEAT GAP ALARM**

• **0550: INVESTIGATE: TIME KEEPER FOR INVALID MEMBERSHIP ALARM**

• **0550: INVESTIGATE: TIME KEEPER FOR MANAGER STOPPED ALARM**

• **0550: INVESTIGATE: TIME KEEPER FOR DATASERVER STOPPED ALARM**

• **0551: INVESTIGATE: TIME KEEPER FOR REMOTE SERVICE STOPPED ALARM**
3. Fault Fencing - fences validated faults, rendering them less disruptive/harmful from the standpoint of the application.
   • 0303: FENCE FAILED NODE
   • 0304: FENCE FAULTED DATASERVER
   • 0305: FENCE UNREACHABLE REMOTE SERVICE
   • 0306: FENCE REPLICATOR FAULT - DIMINISHED DATASOURCE
   • 0306: FENCE REPLICATOR FAULT - EXPIRED ALARM

4. Fault Recovery - attempts to render the fault completely harmless by taking some action that either corrects the fault or by providing alternative resources to manage the fault.
   • 0200a: RECOVER MASTER DATASOURCE BY FAILING OVER - NON-REPLICATOR FAULT
   • 0200b: RECOVER MASTER DATASOURCE BY FAILING OVER - REPLICATOR FAULT
   • 0201: RECOVER COMPOSITE DATASOURCES TO ONLINE
   • 0201a: RECOVER FAILSAFE PHYSICAL SLAVE WITH ONLINE PRIMARY
   • 0201: RECOVER FAILSAFE SHUNNED COMPOSITE SLAVE TO ONLINE
   • 0202: RECOVER OFFLINE PHYSICAL DATASOURCES TO ONLINE
   • 0203: RECOVER FAILED PHYSICAL DATASOURCES TO ONLINE
   • 0204: RECOVER MASTER REPLICATORS TO ONLINE
   • 0205: RECOVER SLAVE REPLICATORS TO ONLINE
   • 0206: RECOVER FROM DIMINISHED STATE WHEN STOPPED REPLICATOR RESTARTS
   • 0207: RECOVER MERGED MEMBERS
   • 0208: RECOVER AND RECONCILE REMOTE DATA SERVICE STATE
   • 0209: PREVENT MULTIPLE ONLINE MASTERS
   • 0210: RECOVER WITNESSES TO ONLINE
   • 0211: RECOVER REMOTE FAILSAFE SHUNNED COMPOSITE MASTER TO ONLINE
   • 0212: RECOVER NON-READ-ONLY SLAVES TO READ-ONLY
7.5. Cluster State Savepoints

The Savepoint functionality was created to support clean, consistent rollbacks during aborted switch and failover operations. This functionality works for both physical clusters as well as for composite clusters.

Savepoints are created automatically with every switch and failover command. The savepoint is only used if there is an exception during switch or failover that is actually able to be rolled-back.

Below is an example of the output:

```
[LOGICAL] /alpha > switch
SELECTED SLAVE: db3@alpha
SET POLICY: AUTOMATIC => MAINTENANCE
Savepoint switch_2(cluster=alpha, source=db1, created=2019/06/29 13:51:56 BST) created
...
```

Not all exceptions during switch and failover will cause a rollback.

In particular, if an exception happens during switch or failover AFTER a new primary datasource has been put online [relay or master] then the switch or failover operation cannot be rolled back.

The Manager is configured, by default, to hold a maximum of 50 savepoints. When that limit is hit, the Manager resets the current-savepoint-id to 0 and starts to overwrite existing savepoints, starting at 0.
Chapter 8. Command-line Tools

Tungsten Clustering is supplied with a number of different command-line tools and utilities that help to install manage, control and provide additional functionality on top of the core Tungsten Clustering product.

The content in this chapter provides reference information for using and working with all of these tools. Usage and operation with these tools in particular circumstances and scenarios are provided in other chapters. For example, deployments are handled in Chapter 2, Deployment, although all deployments rely on the tpm command.

Commands related to the deployment
- tpm — Tungsten package manager

Commands related to managing Tungsten Clustering
- 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
- tungsten_send_diag — assists with diag and file uploads to Continuent support
- tungsten_prep_upgrade — assists with upgrades from one topology to another

8.1. 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.

8.1.1. cctrl Command-line Options

```

Where:
- `-admin` [283]

Table 8.1. cctrl Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-admin</code> [283]</td>
<td>Enter admin mode when connecting</td>
<td>string</td>
</tr>
</tbody>
</table>

Automatically enters `admin` mode when `cctrl` connects to the cluster:

```
shell> cctrl -admin
Tungsten Clustering (for MySQL) 6.1
```
### Command-line Tools

**Table 8.2. cctrl Command-line Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>-expert</td>
<td>Enter expert mode when connecting</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

Automatically enters **expert** mode when **cctrl** connects to the cluster:

```bash
shell> cctrl -expert
Tungsten Clustering (for MySQL) 6.1
alpha: session established, encryption=false, authentication=false
[LOGICAL:EXPERT] /alpha >
```

**Table 8.3. cctrl Command-line Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>-host</td>
<td>Host name of the service manager to use</td>
<td>string</td>
<td>localhost</td>
</tr>
</tbody>
</table>

Allows you to specify the host to connect to when looking for a manager. By default, **cctrl** will connect to the local host:

```bash
shell> cctrl -host host1
Tungsten Clustering (for MySQL) 6.1
alpha: session established, encryption=false, authentication=false
[LOGICAL] /alpha >
```

**Table 8.4. cctrl Command-line Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>-logical</td>
<td>Enter logical mode when connecting</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

Automatically enters **logical** mode when **cctrl** connects to the cluster. This mode is the default when connecting to a typical; using this option forces this mode:

```bash
shell> cctrl -expert
Tungsten Clustering (for MySQL) 6.1
alpha: session established, encryption=false, authentication=false
[LOGICAL:EXPERT] /alpha >
```

**Table 8.5. cctrl Command-line Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>-multi</td>
<td>Allow support for connecting to multiple services</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

Allow support for connecting to multiple services:

```bash
shell> cctrl -multi
Tungsten Clustering (for MySQL) 6.1
alpha: session established, encryption=false, authentication=false
[LOGICAL:EXPERT] /alpha >
```

**Table 8.6. cctrl Command-line Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>-no-history</td>
<td>Disable command history</td>
<td></td>
</tr>
</tbody>
</table>
Prevents `cctrl` from accessing or recording command history during interaction.

- `--physical` [285]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>-physical</td>
<td>Enter physical mode when connecting</td>
<td>string</td>
</tr>
</tbody>
</table>

Table 8.7. `cctrl` Command-line Options

Automatically enters logical mode when `cctrl` connects to the cluster. This mode is the default when connecting to a typical; using this option forces this mode:

```
shell> cctrl -physical
Tungsten Clustering (for MySQL) 6.1
alpha: session established, encryption=false, authentication=false
[PHYSICAL] resource://>
```

- `--port` [285]

Table 8.8. `cctrl` Command-line Options

Specify the TCP/IP port of the service manager

```
shell> cctrl -port
Tungsten Clustering (for MySQL) 6.1
alpha: session established, encryption=false, authentication=false
[PHYSICAL] resource://>
```

- `--proxy` [285]

Table 8.9. `cctrl` Command-line Options

Operate as a proxy service

```
shell> cctrl --proxy
Tungsten Clustering (for MySQL) 6.1
alpha: session established, encryption=false, authentication=false
[PHYSICAL] resource://>
```

- `--service` [285]

Table 8.10. `cctrl` Command-line Options

Enables the selection of a specific service when first connecting to the cluster. For example:

```
shell> cctrl --service east_from_west
Tungsten Clustering (for MySQL) 6.1
east: session established, encryption=false, authentication=false
east_from_west: session established, encryption=false, authentication=false
[LOGICAL] /east_from_west >
```

8.1.2. `cctrl` Modes

- Admin Mode
- Expert Mode
- Logical Mode
8.1.3. cctrl Commands

Table 8.11. cctrl Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>Change to admin mode</td>
</tr>
<tr>
<td>cd</td>
<td>Change to a specific site within a multisite service</td>
</tr>
<tr>
<td>cluster</td>
<td>Issue a command across the entire cluster</td>
</tr>
<tr>
<td>cluster topology validate</td>
<td>Check, validate and report on current cluster topology and health.</td>
</tr>
<tr>
<td>cluster validate</td>
<td>Validate the cluster quorum configuration</td>
</tr>
<tr>
<td>create composite</td>
<td>Create a composite dataservice</td>
</tr>
<tr>
<td>datasource</td>
<td>Issue a command on a single datasource</td>
</tr>
<tr>
<td>expert</td>
<td>Change to expert mode</td>
</tr>
<tr>
<td>failover</td>
<td>Perform a failover operation from a master to a slave</td>
</tr>
<tr>
<td>help</td>
<td>Display the help information</td>
</tr>
<tr>
<td>ls</td>
<td>Show cluster status</td>
</tr>
<tr>
<td>members</td>
<td>List the managers of the dataservice</td>
</tr>
<tr>
<td>physical</td>
<td>Enter physical mode</td>
</tr>
<tr>
<td>ping</td>
<td>Test host availability</td>
</tr>
<tr>
<td>quit, exit</td>
<td>Exit cctrl</td>
</tr>
<tr>
<td>recover master using</td>
<td>Recover the master within a datasource using the specified master</td>
</tr>
<tr>
<td>replicator</td>
<td>Issue a command on a specific replicator</td>
</tr>
<tr>
<td>router</td>
<td>Issue a command on a specific router (connector)</td>
</tr>
<tr>
<td>service</td>
<td>Run a service script</td>
</tr>
<tr>
<td>set</td>
<td>Set management options</td>
</tr>
<tr>
<td>set master</td>
<td>Set the master within a datasource</td>
</tr>
<tr>
<td>show topology</td>
<td>Shows the currently configured cluster topology</td>
</tr>
<tr>
<td>switch</td>
<td>Promote a slave to a master</td>
</tr>
</tbody>
</table>

8.1.3.1. 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.

8.1.3.2. cctrl cd Command

The `cd` command changes the data service being administered. Subsequent commands will only affect the given data service name.
Command-line Tools

Using `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.

### 8.1.3.3. `cctrl` Cluster Command

The `cctrl` command operates at the level of the full cluster.

#### 8.1.3.3.1. `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.

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.

#### 8.1.3.3.2. `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.

#### 8.1.3.3.3. `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.

#### 8.1.3.3.4. `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.

#### 8.1.3.3.5. `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.

#### 8.1.3.3.6. `cctrl` Cluster Validate Command

The `cluster validate` 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 host1/192.168.2.28: ALIVE
HOST host2/192.168.2.21: ALIVE
HOST host3/192.168.2.22: ALIVE
```

287
Command-line Tools

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: host1
REACHABLE CURRENT 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 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 host1/192.168.2.28: 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 3.6.2, “Adding Active Witnesses to an Existing Deployment”
- Add slave hosts to the dataservice, see Section 3.6.1, “Adding Datasources to an Existing Deployment”
- Add passive witnesses to the dataservice, see Section 3.6.3, “Adding Passive Witnesses to an Existing Deployment”

8.1.3.3.7. cctrl cluster topology validate Command

The cluster topology validate will check and validate a cluster topology and, in the process, will report any issues that it finds. The purpose of this command is to provide a fast way to see, immediately, if there are any issues with any components of a cluster.

Here’s an example of the command when run in the context of a composite multi-master cluster:

[LOGICAL] /usa > cluster topology validate
Validating physical cluster 'east' for composite datasource 'east@usa'
Validating datasource 'db4@east'
Validating datasource 'db5@east'
Validating datasource 'db6@east'
Validating physical subservice 'east_from_west' of service 'east'
Validating datasource 'db4@east_from_west'
Validating datasource 'db5@east_from_west'
Validating datasource 'db6@east_from_west'
Physical cluster 'east_from_west' is VALID
Physical cluster 'east' is VALID

Validating physical cluster 'west' for composite datasource 'west@usa'
Validating datasource 'db1@west'
Validating datasource 'db2@west'
Validating datasource 'db3@west'
Validating physical subservice 'west_from_east' of service 'west'
Validating datasource 'db1@west_from_east'
Validating datasource 'db2@west_from_east'
Validating datasource 'db3@west_from_east'
Physical cluster 'west_from_east' is VALID
Physical cluster 'west' is VALID

Here’s an example of the command when run in the context of a composite multi-master cluster:
Validating datasource 'db4@east'
Validating datasource 'db5@east'
Validating datasource 'db6@east'
Validating physical subservice 'east_from_west' of service 'east'
Validating datasource 'db4@east_from_west'
Validating datasource 'db5@east_from_west'
Validating datasource 'db6@east_from_west'
Physical cluster 'east_from_west' is VALID
Physical cluster 'east' is VALID
Validating physical cluster 'east' for composite datasource 'west@usa'
Validating datasource 'db1@west'
Validating datasource 'db2@west'
Validating datasource 'db3@west'
Validating physical subservice 'west_from_east' of service 'west'
Validating datasource 'db1@west_from_east'
Validating datasource 'db2@west_from_east'
Validating datasource 'db3@west_from_east'
INVALID: db3@west_from_east: Primary datasource is 'db1' but replicator points at 'db2'
Physical cluster 'west_from_east' is INVALID
Physical cluster 'west' is INVALID
Composite cluster 'usa' is INVALID

In the above case you can see that there are two issues, shown with the word INVALID at the start of the line: In the cluster west, datasource db2's replicator points at db3 but should point at db1. In the cluster west_from_east, db3's replicator points at db2 but should point at db1.

8.1.3.4. 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 created 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

The following example creates a composite data service named ‘sj_nyc’

```bash
cctrl> 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

```bash
cctrl> cd sj_nyc
    cctrl> create composite datasource sj
```

8.1.3.5. cctrl datasource Command

The datasource command affects a single data source.

datasource
backup
fail
host
offline
online
recover
restore
shun
welcome

Table 8.12. cctrl datasource Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup</td>
<td>Backup a datasource</td>
</tr>
<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>
</tbody>
</table>
### 8.1.3.5.1. `cctrl 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.

```
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
```

### 8.1.3.5.2. `cctrl datasource fail` Command

The `datasource fail` allows you to place a host into the failed state.

To place a node back into an online state, you must issue the `datasource recover` command.

If the cluster is in automatic policy mode, the cluster will attempt to recover the host automatically if the replicator and database are online.

In order to maintain the failed state, switch the cluster to maintenance and/or stop the underlying database and replicator.

The following example changes the state of the node venus, to failed:

```
cctrl> datasource venus fail
```

### 8.1.3.5.3. `cctrl datasource offline` Command

The `datasource offline` allows you to place a host into an offline state. It has no effect if the datasource is already in an offline state.

To place a node back into an online state, you must issue the `datasource online` command.

If the cluster is in automatic policy mode, the cluster will return the host to online automatically.

In order to maintain the offline state, switch the cluster to maintenance and/or stop the underlying database and replicator.

The following example changes the state of the node mercury, to failed:

```
cctrl> datasource mercury offline
```

### 8.1.3.5.4. `cctrl datasource online` Command

The `datasource online` allows you to place a host into an online state. It has no effect if the datasource is already in an online state.

To place a node back into an online state, you must issue the `datasource online` command.

If the node is in a SHUNNED or FAIL state, this command will fail with an error. Instead, you should use the `datasource recover` command.

The following example changes the state of the node mercury, to failed:

```
cctrl> datasource mercury online
```

### 8.1.3.5.5. `cctrl datasource recover` Command

The `datasource recover` reconfigures a shunned data source and returns it to the cluster as a slave. This command can be used with failed master as well as slave data sources.

For slave data sources, the recover command attempts to restart the DBMS server followed by replication. If successful, the data source joins the cluster as an online slave.
For master data sources, the recover command first reconfigures the master as a slave. It then performs the same recovery process as for a failed slave.

If datasource recover is unsuccessful, the next step is typically to restore the data source from a backup. This should enable it to rejoin the cluster as a normal slave.

The following example recovers host mercury following a failure. The command is identical for master and slave data sources.

```
cctrl> datasource mercury recover
```

#### 8.1.3.5.6. cctrl datasource restore Command

The `datasource restore` command reloads a backup generated with the `datasource backup` command.

The following command options are supported:

- **uri** — The URI of a specific backup to restore
- **timeout** — Number of seconds to wait before the command times out.

To restore a data source you must first put the data source and its associated replicator offline.

The following example restores host saturn from the latest backup. The preceding commands place the datasource and replicator offline.

```
cctrl> datasource saturn shun
cctrl> datasource saturn offline
cctrl> replicator saturn offline
cctrl> datasource saturn restore
cctrl> datasource saturn welcome
cctrl> cluster online
```

The following example restores host mercury from an existing backup, which is explicitly named. The datasource and replicator must be offline.

```
cctrl> datasource mercury restore storage://file-system/store-0000000004.properties
```

#### 8.1.3.5.7. cctrl datasource shun Command

The `datasource shun` command removes the data source from the cluster and makes it unavailable to applications. It will remain in the shunned state without further changes until you issue a `datasource welcome` or `datasource recover` command.

The `datasource shun` command is most commonly used to perform maintenance on a data source. It allows you to reboot servers and replicators without triggering automated policy mode rules.

The following example shuns the data source on host venus.

```
cctrl> datasource venus shun
```

#### 8.1.3.5.8. cctrl 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:

```
|host3(slave:ONLINE, progress=157454, latency=1.000) | |  |
|STATUS [OK] [2013/05/14 05:12:54 PM BST] | |
| | |
|MANAGER(state=ONLINE) | |
|REPLICATOR(role=slave, master=host2, state=ONLINE) | |
|DATASERVER(state=ONLINE) | |
|CONNECTIONS(created=0, active=0) | |

```

```
[LOGICAL:EXPERT] /alpha > datasource host3 shun
Datasource 'host3' set to SHUNNED
```

To switch the datasource back to the online state, the welcome is used:

```
[LOGICAL:EXPERT] /alpha > datasource host3 welcome
```
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 8.1.3.14, "**cctrl recover Command**".

### 8.1.3.6. cctrl expert Command

The **expert** command enables expert mode in cctrl. This suppresses prompts for commands that can cause damage to data. It is provided as a convenience for fast administration of the system.

**Warning**

This mode should be used with care, and only be used by experienced operators who fully understand the implications of the subsequent commands issued. Missuse of this feature may cause irreparable damage to a cluster.

### 8.1.3.7. cctrl failover Command

The **failover** command performs a failover to promote an existing slave to master after the current master has failed. The master data source must be in a failed state to use **failover**. If the master data source is not failed, you should instead use **switch**.

If there is no argument the **failover** command selects the most caught up slave and promotes it as the master. You can also specify a particular host, in which case **failover** will ensure that the chosen slave is fully up-to-date and promote it.

Failover ensures that the slave has applied all transactions present in its log, then promotes the slave to master. It does not attempt to retrieve transactions from the old master, as this is by definition already failed. After promoting the chosen slave to master, **failover** reconfigures other slaves to point to it and ensures all data sources are online.

To recover a failed master you should use the **datasource recover** command.

Failover to any up-to-date slave in the cluster. If no slave is available the operation fails:

```bash
cctrl> failover
```

Failover from a broken master to a specific node:

```bash
cctrl> failover to mercury
```

### 8.1.3.8. 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 > help
........
Description: Overview of Tungsten cctrl Commands

Commands:
admin                          - Enter admin mode
cl <name>                      - Change to the specified SOR cluster element
cluster <command>              - Issue a command on the entire cluster
create composite <type> <name>- Create SOR cluster components
datasource <host> <cmd>        - Issue a command on a datasource
datasource recover            - Ensure the replicator, connector and operation of the datasource are correct within the current cluster configuration
expert                         - Enter expert mode
failover                       - Failover from failed master to slave
help                           - Show help
ls [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 <host> <cmd>        - Issue a command on a replicator
```
Command-line Tools

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.

8.1.3.9. 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 the 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
  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, issue

  cctrl> ls host1

- resources
  The resources option generates a list of the configured resources and their current status.

- services
  The services option generates a list of the configured services known to the manager.

- sessions
  The sessions outputs statistics for the cluster. Statistics will only be presented when SMARTSCALE is enabled for the connectors

Without any further options, the output of ls looks similar to the following:

```
[LOGICAL] /alpha > ls
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)                                                  |
+----------------------------------------------------------------------------+
```
8.1.3.10. cctrl members Command

The members command outputs a list of the currently identified managers within the dataservice.

For example:

```
[LOGICAL] /alpha > members
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.

8.1.3.11. cctrl physical Command

The members command enables physical mode commands and displays. This is a specialized mode used to examine interfaces of resources managed by the cluster. It is not recommended for normal administrative use.

8.1.3.12. cctrl ping Command

The ping command checks to see whether a host is alive. If the host name is omitted, it tests all hosts in the cluster including witness hosts.

Ping uses the host ping timeout and methods specified in the manager.properties file. By default output is parsimonious.

The following shows and example of the output:

```
[LOGICAL] /nyc > ping
NETWORK CONNECTIVITY: PING TIMEOUT=2
NETWORK CONNECTIVITY: CHECKING MY OWN ('db2') CONNECTIVITY
NETWORK CONNECTIVITY: CHECKING CLUSTER MEMBER 'db1'
NETWORK CONNECTIVITY: CHECKING CLUSTER MEMBER 'db3'
```
8.1.3.13. cctrl quit Command

Exits cctrl and returns the user to the shell. For example:

8.1.3.14. cctrl recover Command

The recover will attempt to recover and bring online all nodes and services that are not in an ONLINE state.

Any previous failed Master nodes will be reconfigured as Slaves, and all associated replicator services will be reconciled to connect to correct Master

If recovery is unsuccessful, the next step is typically to restore any failed data source from a backup.

8.1.3.15. cctrl recover master using Command

8.1.3.16. cctrl recover relay using Command

8.1.3.17. cctrl recover using Command

8.1.3.18. cctrl replicator Command

8.1.3.19. cctrl rm Command

8.1.3.20. cctrl router Command

8.1.3.21. cctrl service Command

The service command executes a command on the operating system according to standard Linux/Unix service script conventions. The service command may apply to a single host or may be executed on all hosts using the * operator. This latter form is also known as a broadcast command. You can enter service commands from any manager.

Commonly defined services include the following. User-defined services may also be invoked using the service command provided they are listed in the service configuration files for the cluster.

- connector: Tungsten Connector service
- mysql: MySQL service
- replicator: Tungsten Replicator service

The standard service commands are:

- restart: Stop and then start the service
- start: Start the service if it is not already running
- status: Show the current process status
- stop: Stop the service if it is running
- tail: Show the end of the process log (useful for diagnostics)

To start all mysqld processes in the cluster. This should be done in. maintenance mode to avoid triggering a failover.

```
cctrl> service */mysql restart
```

Stop the replicator process on host mercury.

```
cctrl> service mercury/replicator tail
```

Show the end of the log belonging to the connector process on host jupiter.

```
cctrl> service jupiter/connector tail
```
Warning

[Re-]starting master DBMS servers can cause failover when operating in automatic policy mode. Always set policy mode to maintenance before restarting a master DBMS server.

8.1.3.22. cctrl set Command

The `set` command sets a management option. The following options are available.

- set policy - Set policy for cluster automation
- set output - Set logging level in cctrl

8.1.3.23. cctrl set force Command

8.1.3.24. cctrl show topology Command

The `show topology` command shows the topology for the currently selected cluster, or cluster composite.

For example, below is sample output for a Composite Primary/DR (active/passive) cluster:

```
[LOGICAL] /east > show topology
clustered_master_slave
```

For example, below is sample output for a Composite Multi-Master (active/active) cluster:

```
[LOGICAL] /usa > show topology
clustered_multi_master
```

When selecting a cluster within the composite:

```
[LOGICAL] /usa > use east
[LOGICAL] /east > show topology
clustered_primary
```

The following values are output according to the cluster selected:

- Primary service returns `clustered-primary`
- Sub-service returns `clustered-sub-service`
- Composite master-slave returns `composite-master-slave`
- Composite multi-master returns `composite-multi-master`

8.1.3.25. cctrl set master Command

8.1.3.26. cctrl switch Command

The `switch` command performs a planned failover to promote an existing slave to master and reconfigure the current master as a slave.

The most common reason for a switch operation is to perform maintenance on the master.

If there is no argument the switch command selects the most caught up slave and promotes it as the master. You can also specify a particular host, in which case switch will ensure that the chosen slave is fully up-to-date and promote it.

Switch is a complex operation.

- First, we ensure that all transactions to the master through SQL router or connector processes complete before initiating the switch.
- It submits a flush transaction through the replicator to ensure that the chosen slave is fully caught up with the master.
- It then reconfigures the master and slave to reverse their roles.
- Finally, it puts the master and slave back online.

In the event that switch does not complete, We attempt to revert to the old master. If a switch fails, you should check the cluster using 'ls' to ensure that things rolled back correctly.
Command-line Tools

Examples:

Switch to any up-to-date slave in the cluster. If no slave is available the operation fails.

```
cctrl> switch
```

Switch the master to host mercury.

```
cctrl> switch to mercury
```

The `switch` command can also be used to switch between Composite Master and Composite Slave clusters in a Composite DR Topology.

In this scenario, the switch will promote the RELAY node in the remote cluster to be the master, and revert the Master in the local cluster to be the Relay.

To initiate the switch in a composite cluster, issue the command from the composite cluster, for example if you have cluster service east and west in a composite cluster called global, and east is the current composite master:

```
cctrl> use global
cctrl> switch
```

8.2. The `check_tungsten_latency` Command

The `check_tungsten_latency` command reports warning or critical status information depending on whether the latency across the nodes in the cluster is above a specific level.

### Table 8.13. `check_tungsten_latency` Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-c</td>
<td>Report a critical status if the latency is above this level</td>
</tr>
<tr>
<td>--perfdata</td>
<td>Show the latency performance information</td>
</tr>
<tr>
<td>--perslave-perfdata</td>
<td>Show the latency performance information on a per-slave basis</td>
</tr>
<tr>
<td>-w</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) | max_latency=0.506:1;1;
```
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;;
```

8.3. The check_tungsten_online Command

The `check_tungsten_online` command checks whether all the hosts in a given service are online and running.

Within a Tungsten Clustering service, the replicator, manager and connector services are checked. All must be online for an OK response.

Table 8.14. `check_tungsten_online` Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h</td>
<td>Display the help text</td>
</tr>
<tr>
<td>-port</td>
<td>RMI port for the replicator being checked</td>
</tr>
<tr>
<td>-s</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
```

8.4. The check_tungsten_policy Command

The `check_tungsten_policy` command checks whether the policy is in **AUTOMATIC** mode or not.

Table 8.15. `check_tungsten_policy` Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h</td>
<td>Display the help text</td>
</tr>
</tbody>
</table>

This command only needs to be run on one node within the service; the command returns the policy mode 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 the policy is not *AUTOMATIC*.
  
  For example:
  
  ```text
  CRITICAL: Policy is MAINTENANCE
  ```

- **WARNING** — status requires attention. This indicates that one service within the system is not online.

- **OK** — status is OK.
  
  For example:
  
  ```text
  OK: Policy is AUTOMATIC
  ```

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:

```text
shell> check_tungsten_policy
OK: Policy is AUTOMATIC
```

### 8.5. 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 8.16. `check_tungsten_progress` Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-s</code></td>
<td>Service name to check (optional)</td>
</tr>
<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:

```text
LEVEL: DETAIL
```

Where DETAIL includes detailed information about the status report, and LEVEL is:

- **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:
  
  ```text
  CRITICAL: Replicator is not ONLINE
  ```

- **OK** — replicator is making progress.
  
  For example:
  
  ```text
  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, especially if you see errors like this:

```text
CRITICAL: Replicator is not ONLINE
```

For example, to wait 5 seconds to ensure the replicator is progressing:

```text
shell> check_tungsten_progress -t 5
OK: Replicator is making progress
```

Version 6.0.4 Optionally, specify the replication service name using the `-s` option. This is normally only needed for Composite Multimaster (CMM) deployments where there is no single default replication service.

```text
shell> check_tungsten_progress -t 5 -s east_from_west
OK: Replicator is making progress
```
8.6. The check_tungsten_services Command

The `check_tungsten_services` command provides a simple check to confirm whether configured services are currently running. The command must be executed with a command-line option specifying which services should be checked and confirmed.

**Table 8.17. check_tungsten_services Options**

<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:
  
  OK: All services (Replicator) are online

This output is easily parseable by various monitoring tools, including Nagios NRPE, and can be used to monitor the status of your services quickly without restoring to using the full `trepctl` output.

**Note**

The `check_tungsten_services` only confirms that the services and processes are running; their state is not confirmed. To check state with a similar interface, use the `check_tungsten_online` command.

To check the services:

- To check the replicator services:
  
  shell> `check_tungsten_services -r`
  
  OK: All services (Replicator) are online

- 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

8.7. The clean_release_directory Command

The `clean_release_directory` is located in the `tools` directory removes older releases of the installed product from the installation directory. Over time, as `tpm` update the configuration or new releases of the product, new directories with the full release information are created, but old ones are not removed in case you need to go back to a previous release.

The `clean_release_directory` command removes all but the five most recent installs and the current release. For example, with the following directory:

```
shell> `ls -l /opt/continuent/releases`
drwxrwxr-x 17 mc mc 4096 Jul  7 15:36 ./
drwxr-xr-x  9 mc mc 4096 Jul  7 15:36 ../
drwxrwxr-x  2 mc mc 4096 Jul  7 15:36 install/
drwxr-xr-x  5 mc mc 4096 Jul  7 15:36 tungsten-replicator-5.2.0-218_pid16197/
drwxr-xr-x  5 mc mc 4096 Jul  7 15:36 tungsten-replicator-5.2.0-219_pid10303/
drwxr-xr-x  5 mc mc 4096 Jul  7 15:36 tungsten-replicator-5.2.0-219_pid1393/
```
Command-line Tools

8.8. 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.

```
class_cluster_backup [ --agent ] [ --datasource ] [ --directory ] [ --help, -h ] [ --info, -i ] [ --json ] [ --net-ssh-option ] [ --notice, -n ] [ --offline-backup String ] [ --quiet, -q ] [ --require-automatic-mode String ] [ --require-slave-backup String ] [ --validate ] [ --verbose, -v ]
```

Where:

Table 8.18. `cluster_backup` Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--agent</code></td>
<td>The replicator backup agent to use when running the backup, as configured in cctrl</td>
</tr>
<tr>
<td><code>--datasource</code></td>
<td>Execute the backup against the specified datasource</td>
</tr>
<tr>
<td><code>--directory</code></td>
<td>Use this installed Tungsten directory as the base for all operations</td>
</tr>
<tr>
<td><code>--help, -h</code></td>
<td>Display this message</td>
</tr>
<tr>
<td><code>--info, -i</code></td>
<td>Provide return code and logging messages as a JSON object after the script finishes</td>
</tr>
<tr>
<td><code>--json</code></td>
<td>Set the Net:SSH option for remote system calls</td>
</tr>
<tr>
<td><code>--net-ssh-option</code></td>
<td></td>
</tr>
</tbody>
</table>
Option | Description
--- | ---
--notice, -n | 
--offline-backup String | Put the datasource OFFLINE prior to taking the backup.
--quiet, -q | 
--require-automatic-mode String | The script will fail if the cluster is not in the AUTOMATIC policy mode.
--require-slave-backup String | 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.
--validate | Only run the script validation.
--verbose, -v | 

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.

The `cluster_backup` command may also be configured for use in Composite clusters. In this use case, one backup per cluster will be created. See Section 5.10.2, “Automating Backups” for more information.

For example:
```
shells> crontab -l
00 00 * * * /opt/continuent/tungsten/cluster-home/bin/cluster_backup >>/opt/continuent/service_logs/cluster_backup.log 2>&1
```

All output will be sent to /opt/continuent/service_logs/cluster_backup.log.

### 8.9. The connector Command

The `connector` is the wrapper script that handles the execution of the connector service.

Table 8.19. connector Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>client-list</code>[303]</td>
<td>Return a list of the current client connections through this connector.</td>
</tr>
<tr>
<td><code>cluster-status</code>[303]</td>
<td>Return the cluster status, as the connector currently understands it. This is the command-line alternative to the inline cluster status command.</td>
</tr>
<tr>
<td><code>condrestart</code>[303]</td>
<td>Restart only if already running</td>
</tr>
<tr>
<td><code>console</code>[303]</td>
<td>Launch in the current console (instead of a daemon)</td>
</tr>
<tr>
<td><code>dump</code>[303]</td>
<td>Request a Java thread dump (if connector is running)</td>
</tr>
<tr>
<td><code>graceful-stop {seconds}</code>[303]</td>
<td>Stops the connector gracefully, allowing outstanding open connections to finish and close before the connector process is stopped. {seconds} is an integer of zero or more, and is required. The connector will shut down immediately if there are no active connections. Specifying zero (0) will cause a shutdown without waiting for connections to terminate.</td>
</tr>
<tr>
<td><code>install</code>[303]</td>
<td>Install the service to automatically start when the system boots</td>
</tr>
<tr>
<td><code>reconfigure</code>[303]</td>
<td>Reconfigure the connector by forcing the connector to reread the configuration, including the configuration files and <code>user.map</code>.</td>
</tr>
<tr>
<td><code>remove</code>[304]</td>
<td>Remove the service from starting during boot</td>
</tr>
<tr>
<td><code>restart</code>[304]</td>
<td>Stop connector if already running and then start</td>
</tr>
<tr>
<td><code>start</code>[304]</td>
<td>Start in the background as a daemon process</td>
</tr>
<tr>
<td><code>status</code>[304]</td>
<td>Query the current status</td>
</tr>
<tr>
<td><code>stop</code>[304]</td>
<td>Stop if running (whether as a daemon or in another console)</td>
</tr>
</tbody>
</table>

These commands and options are described below:
- **client-list [303]**

Lists all ongoing connections with origin, local and remote [mysql data source] IPs/ports

```
shell> connector client-list
Executing Tungsten Connector Service --client-list ...
+----------------------+---------------------+----------------------+------------------------+
| Client | Connector Inbound | Connector Outbound | Data Source |
+----------------------+---------------------+---------------------+------------------------+
+----------------------+---------------------+---------------------+------------------------+
Done Tungsten Connector Service --client-list
```

- **cluster-status [303]**

Returns the cluster status, in the same form as the inline tungsten cluster status command. In addition, the same information can be generated encapsulated in JSON format by adding the `--json` option. See Section 6.10.1.1, "Connector connector cluster status on the Command-line", for more information.

```
shell> connector cluster-status
Executing Tungsten Connector Service --cluster-status ...
+--------------+--------------------+-------------+--------------+--------+--------+------------------------------------------+------------------+-----------------+------------------+--------------------+---------------------+
| Data service | Data service state | Data source | Is composite | Role | State | High water | Last shun reason | Applied latency | Relative latency | Active connections | Connections created |
+--------------+--------------------+-------------+--------------+--------+--------+------------------------------------------+------------------+-----------------+------------------+--------------------+---------------------+
| europe | ONLINE | c1 | false | master | ONLINE | 0(c1-bin.000002:0000000000001219;113912) | 1.0 | 113.0 | 1 | 0 |
| europe | ONLINE | c2 | false | slave | ONLINE | 0(c1-bin.000002:0000000000001219;113912) | 1.0 | 113.0 | 1 | 0 |
| europe | ONLINE | c3 | false | slave | ONLINE | 0(c1-bin.000002:0000000000001219;113912) | 1.0 | 113.0 | 1 | 0 |
+--------------+--------------------+-------------+--------------+--------+--------+------------------------------------------+------------------+-----------------+------------------+--------------------+---------------------+
Done Tungsten Connector Service --cluster-status
```

- **condrestart [303]**

Restart the connector, only if it is already running. This can be useful to use when changing configuration or performing database management within automated scripts, as the connector will be only be restart if it was previously running.

For example, if the connector is running, `connector condrestart [303]` operates as `connector restart`:

```
shell> connector condrestart
Stopping Tungsten Connector Service...
Waiting for Tungsten Connector Service to exit...
Starting Tungsten Connector Service...
Waiting for Tungsten Connector Service......
Running: PID:26646
```

However, if not already running, the operation does nothing:

```
shell> connector condrestart
Stopping Tungsten Connector Service...
Tungsten Connector Service was not running.
```

- **console [303]**

Launch in the current console (instead of a daemon)

- **dump [303]**

Request a Java thread dump (if connector is running)

- **graceful-stop {seconds} [303]**

Stops the connector gracefully, allowing outstanding open connections to finish and close before the connector process is stopped. `{seconds}` is an integer of zero or more, and is required. The connector will shut down immediately if there are no active connections. Specifying zero (0) will cause a shutdown without waiting for connections to terminate.

- **install [303]**

Installs the startup scripts for running the connector at boot. For an alternative method of deploying these start-up scripts, see `deployall`.

- **reconfigure [303]**

Reloads the configuration without touching existing connections. New configuration only applies to new connections. Note that this command won't work for the following settings: `connection.keepAlive.interval`, `connection.keepAlive.timeout`, `server.listen.address`, `server.port`, `server.port.binding.timeout`, `server.port.binding.retry.delay` (when reaching `server.max_connections` only)
- remove [304]

Removes the startup scripts for running the connector at boot. For an alternative method of removing these start-up scripts, see undeployall.

- restart [304]

Stops the connector, if it is already running, and then restarts it:

```shell
shell> connector restart
Stopping Tungsten Connector Service...
Stopped Tungsten Connector Service.
Starting Tungsten Connector Service.
Waiting for Tungsten Connector Service......
running: PID:26248
```

- start [304]

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

```shell
shell> connector start
Starting Tungsten Connector Service...
```

- status [304]

Checks the execution status of the connector:

```shell
shell> connector status
Tungsten Connector Service is running: PID:27015, Wrapper:STARTED, Java:STARTED
```

If the connector is not running:

```shell
shell> connector status
Tungsten Connector Service is not running.
```

This only provides the execution state of the connector, not the actual state of replication. To get detailed information on the status of replication use trepcrt status.

- stop [304]

Stops the connector if it is already running:

```shell
shell> connector stop
Stopping Tungsten Connector Service...
Waiting for Tungsten Connector Service to exit...
Stopped Tungsten Connector Service.
```

8.10. The ddlscan Command

The `ddlscan` command scans the existing schema for a database or table and then generates a schema or file in a target database environment. For example, `ddlscan` is used in MySQL to Oracle heterogeneous deployments to translate the schema definitions within MySQL to the Oracle format. For more information on heterogeneous deployments, see Understanding Heterogeneous Deployments (in [Tungsten Replicator 6.1 Manual]).

For example, to generate Oracle DDL from an existing MySQL database:

```shell
shell> ddlscan -user tungsten -url "jdbc:mysql:thin://tr-hadoop1:13306/test" -pass password \
        -template ddl-mysql-oracle.vm -db test
SQL generated on Thu Sep 11 15:39:06 BST 2014 by ./ddlscan utility of Tungsten

url = jdbc:mysql:thin://host1:13306/test
user = tungsten
dbName = test
*/
DROP TABLE test.sales;
CREATE TABLE test.sales
(
    id NUMBER(10, 0) NOT NULL,
    salesman CHAR,
    planet CHAR,
    value FLOAT,
    PRIMARY KEY (id)
);
```

The format of the command is:
Command-line Tools

```
```

The available options are as follows:

**Table 8.20. ddlscan Command-line Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-conf path [305]</td>
<td>Path to a static-{svc}.properties file to read JDBC connection address and credentials</td>
</tr>
<tr>
<td>-db db [305]</td>
<td>Database to use (will substitute $DBNAME in the URL, if needed)</td>
</tr>
<tr>
<td>-opt opt val [306]</td>
<td>Option(s) to pass to template, try: -opt help me</td>
</tr>
<tr>
<td>-out file [306]</td>
<td>Render to file (print to stdout if not specified)</td>
</tr>
<tr>
<td>-pass secret [305]</td>
<td>JDBC password</td>
</tr>
<tr>
<td>-path path [306]</td>
<td>Add additional search path for loading Velocity templates</td>
</tr>
<tr>
<td>-rename file [305]</td>
<td>Definitions file for renaming schemas, tables and columns</td>
</tr>
<tr>
<td>-service name [305]</td>
<td>Name of a replication service instead of path to config</td>
</tr>
<tr>
<td>-tableFile file [305]</td>
<td>New-line separated definitions file of tables to find</td>
</tr>
<tr>
<td>-tables regex [305]</td>
<td>Comma-separated list of tables to find</td>
</tr>
<tr>
<td>-template file [305]</td>
<td>Specify template file to render</td>
</tr>
<tr>
<td>-url jdbcUrl [305]</td>
<td>JDBC connection string (use single quotes to escape)</td>
</tr>
<tr>
<td>-user user [305]</td>
<td>JDBC username</td>
</tr>
</tbody>
</table>

**ddlscan** supports three different methods for execution:

- Using an explicit JDBC URL, username and password:

  ```shell```
  ddlscan -user tungsten -url 'jdbc:mysql:thin://tr-hadoop1:13306/test' -user user -pass password ...
  ```shell```

  This is useful when a deployment has not already been installed.

- By specifying an explicit configuration file:

  ```shell```
  ddlscan -conf /opt/continuent/tungsten/tungsten-replicator/conf/static-alpha.properties ...
  ```shell```

- When an existing deployment has been installed, by specifying one of the active services:

  ```shell```
  ddlscan -service alpha ...
  ```shell```

In addition, the following two options must be specified on the command-line:

- The template to be used (using the -template [305] option) for the DDL translation must be specified on the command-line. A list of the support templates and their operation are available in Table 8.21, "ddlscan Supported Templates".

- The -db [305] parameter, which defines the database or schema that should be scanned. All tables are translated unless an explicit list, regex, or table file has been specified.

For example, to translate MySQL DDL to Oracle for all tables within the schema `test` using the connection to MySQL defined in the service `alpha`:

```shell```
ddlscan -service alpha -template ddl-mysql-oracle.vm -db test
```shell```

**ddlscan** provides a series of additional command-line options, and a full list of the available templates.

### 8.10.1. Optional Arguments

The following arguments are optional:

- -tables [305]
- -rename [305]

A list of table renames which will be taken into account when generating target DDL. The format of the table matches the format of the `rename` filter.
• **-path**

  The path to additional Velocity templates to be searched when specifying the template name.

• **-opt**

  An additional option (and variable) which are supplied to be used within the template file. Different template files may support additional options for specifying alternative information, such as schema names, file locations and other values.

```shell
ddlscan --service alpha --template ddl-mysql-oracle.vm --db test --opt schemaPrefix mysql_
```

• **-out**

  Sends the generated DDL output to a file, in place of sending it to standard output.

• **-help**

  Generates the help text of arguments.

### 8.10.2. Supported Templates and Usage

#### Table 8.21. *ddlscan* Supported Templates

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ddl-check-pkeys.vm</td>
<td>Reports which tables are without primary key definitions</td>
</tr>
<tr>
<td>ddl-mysql-cassandra.vm</td>
<td>Generates DDL from a MySQL host suitable for the base tables in Apache Cassandra</td>
</tr>
<tr>
<td>ddl-mysql-cassandra-staging.vm</td>
<td>Generates DDL from a MySQL host suitable for the base tables in Apache Cassandra</td>
</tr>
<tr>
<td>ddl-mysql-hive-8.10.vm</td>
<td>Generates DDL from a MySQL host suitable for the base tables in a Hive Environment</td>
</tr>
<tr>
<td>ddl-mysql-hive-8.10-staging.vm</td>
<td>Generates DDL from a MySQL host suitable for the staging tables in a Hive Environment</td>
</tr>
<tr>
<td>ddl-mysql-hive-metadata.vm</td>
<td>Generates metadata as JSON to be used within a Hive Environment</td>
</tr>
<tr>
<td>ddl-mysql-oracle.vm</td>
<td>Generates Oracle schema from a MySQL schema</td>
</tr>
<tr>
<td>ddl-mysql-oracle-cdc.vm</td>
<td>Generates Oracle tables with CDC capture information from a MySQL schema</td>
</tr>
<tr>
<td>ddl-mysql-redshift.vm</td>
<td>Generates DDL from a MySQL host suitable for the base tables in Amazon Redshift</td>
</tr>
<tr>
<td>ddl-mysql-redshift-staging.vm</td>
<td>Generates DDL from a MySQL host suitable for the staging tables in Amazon Redshift</td>
</tr>
<tr>
<td>ddl-mysql-vertica.vm</td>
<td>Generates DDL suitable for the base tables in HP Vertica</td>
</tr>
<tr>
<td>ddl-mysql-vertica-staging.vm</td>
<td>Generates DDL suitable for the staging tables in HP Vertica</td>
</tr>
<tr>
<td>ddl-oracle-mysql.vm</td>
<td>Generates DDL for MySQL tables from an Oracle schema</td>
</tr>
<tr>
<td>ddl-oracle-mysql-pk-only.vm</td>
<td>Generates Primary Key DDL statements from an Oracle database for MySQL</td>
</tr>
</tbody>
</table>

#### 8.10.2.1. *ddl-check-pkeys.vm*

The *ddl-check-pkeys.vm* template can be used to check whether specific tables within a schema do not have a primary key:

```shell
ddlscan -template ddl-check-pkeys.vm \
-user tungsten -pass password -db sales \
-url jdbc:mysql://localhost:13306/sales
```

SQL generated on Thu Sep 04 18:23:52 BST 2014 by ./ddlscan utility of Tungsten

```sql
url = jdbc:mysql://localhost:13306/sales
user = tungsten
dbname = sales
*/
```
For certain environments, particularly heterogeneous replication, the lack of primary keys can lead to inefficient replication, or even fail to replicate data at all.

8.10.2.2. ddl-mysql-cassandra.vm

The **ddl-mysql-cassandra.vm** template generates DDL for Apache Cassandra tables from MySQL schemas. For example:

```sql
CREATE TABLE "test"."all_mysql_types"
(
    "my_id" INT ,
    "my_bit" BOOLEAN /* BIT(1) */ ,
    "my_tinyint" INT ,
    "my_boolean" INT ,
    "my_smallint" INT ,
    "my_mediumint" INT ,
    "my_int" INT ,
    "my_bigint" BIGINT ,
    "my_decimal_18_5" DECIMAL ,
    "my_float" FLOAT ,
    "my_double" DOUBLE ,
    "my_date" DATE ,
    "my_datetime" TIMESTAMP ,
    "my_timestamp" TIMESTAMP ,
    "my_time" TIME ,
    "my_year" DECIMAL ,
    "my_char_10" TEXT ,
    "my_varchar_10" TEXT ,
    "my_tinytext" TEXT ,
    "my_text" TEXT ,
    "my_mediumtext" TEXT ,
    "my_longtext" TEXT ,
    "my_enum_abc" TEXT ,
    "my_set_def" TEXT ,
    primary key ("my_id")
);```

Columns are translated as follows:

<table>
<thead>
<tr>
<th>Oracle Datatype</th>
<th>Cassandra Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT</td>
<td>BIGINT</td>
</tr>
<tr>
<td>BIT</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>INT</td>
</tr>
<tr>
<td>CHAR</td>
<td>TEXT</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
</tr>
<tr>
<td>DATETIME</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>ENUM</td>
<td>TEXT</td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
</tr>
<tr>
<td>INT</td>
<td>INT</td>
</tr>
<tr>
<td>LONGTEXT</td>
<td>TEXT</td>
</tr>
<tr>
<td>MEDIUMINT</td>
<td>INT</td>
</tr>
<tr>
<td>MEDIUMTEXT</td>
<td>TEXT</td>
</tr>
</tbody>
</table>
The following additional considerations are taken into account:

- The primary key generated is based on the primary key for the table and the opcode column used during processing. This provides optimal performance during loading.
- All column names are quoted (using double quotes) to prevent reserved word issues.
- Cassandra has significantly fewer types and so data and time values are often translated to simpler types.

8.10.2.3. **ddl-mysql-cassandra-staging.vm**

Creates a table suitable for use within Cassandra as a staging table for data from MySQL.

```sql
CREATE TABLE "test"."stage_xxx_all_mysql_types"
(
    "tungsten_opcode" TEXT ,
    "tungsten_seqno" INT ,
    "tungsten_row_id" INT ,
    "tungsten_commit_timestamp" TIMESTAMP ,
    "my_id" INT ,
    "my_bit" BOOLEAN /* BIT(1) */ ,
    "my_tinyint" INT ,
    "my_boolean" INT ,
    "my_smallint" INT ,
    "my_mediumint" INT ,
    "my_int" INT ,
    "my_bigint" BIGINT ,
    "my_decimal_10_5" DECIMAL ,
    "my_float" FLOAT ,
    "my_double" DOUBLE ,
    "my_date" DATE ,
    "my_datetime" TIMESTAMP ,
    "my_timestamp" TIMESTAMP ,
    "my_time" TIME ,
    "my_year" DECIMAL ,
    "my_char_10" TEXT ,
    "my_varchar_10" TEXT ,
    "my_tinytext" TEXT ,
    "my_text" TEXT ,
    "my_mediumtext" TEXT ,
    "my_longtext" TEXT ,
    "my_enum_abc" TEXT ,
    "my_set_def" TEXT ,
    primary key (tungsten_opcode,"my_id")
) ;
```

All datatype translates are identical to those used in **ddl-mysql-cassandra.vm**.

8.10.2.4. **ddl-mysql-hive-0.10.vm**

Generates DDL suitable for a carbon-copy form of the table from the MySQL host:

```bash
shell> ddiscan -user tungsten -url "jdbc:mysql://tr-hadoop1:13306/test" -pass password \ -template ddl-mysql-hive-0.10.vm -db test
```

---

SQL generated on Thu Sep 11 12:57:11 BST 2014 by Tungsten ddiscan utility

- url = jdbc:mysql://tr-hadoop1:13306/test
- user = tungsten
- dbName = test
DROP TABLE IF EXISTS test.sales;
CREATE TABLE test.sales
(
  id INT,
  salesman STRING,
  planet STRING,
  value DOUBLE
);

Wherever possible, the closest Hive equivalent datatype is used for each source datatype, as follows:

<table>
<thead>
<tr>
<th>MySQL Datatype</th>
<th>Hive Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATETIME</td>
<td>STRING</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>DATE</td>
<td>STRING</td>
</tr>
<tr>
<td>YEAR</td>
<td>INT</td>
</tr>
<tr>
<td>TIME</td>
<td>STRING</td>
</tr>
<tr>
<td>TINYINT</td>
<td>TINYINT</td>
</tr>
<tr>
<td>TINYINT UNSIGNED</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>SMALLINT UNSIGNED</td>
<td>INT</td>
</tr>
<tr>
<td>MEDIUMINT</td>
<td>INT</td>
</tr>
<tr>
<td>INT</td>
<td>INT</td>
</tr>
<tr>
<td>INT UNSIGNED</td>
<td>BIGINT</td>
</tr>
<tr>
<td>BIGINT</td>
<td>BIGINT</td>
</tr>
<tr>
<td>BIGINT UNSIGNED</td>
<td>STRING</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>STRING</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>STRING</td>
</tr>
<tr>
<td>CHAR</td>
<td>STRING</td>
</tr>
<tr>
<td>BINARY</td>
<td>BINARY</td>
</tr>
<tr>
<td>VARBINARY</td>
<td>BINARY</td>
</tr>
<tr>
<td>TEXT</td>
<td>STRING</td>
</tr>
<tr>
<td>BLOB</td>
<td>BINARY</td>
</tr>
<tr>
<td>FLOAT</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>ENUM</td>
<td>STRING</td>
</tr>
<tr>
<td>SET</td>
<td>STRING</td>
</tr>
<tr>
<td>BIT</td>
<td>STRING</td>
</tr>
</tbody>
</table>

The template supports the following optional parameters to change behavior:

- `--opt schemaPrefix`

  A prefix to be placed in front of all schemas. For example, if called with `schemaPrefix` set to `mysql_`:

  ```shell```
  shell> ddlscan ... -opt schemaPrefix mysql_
  ```shell```

  The schema name will be prefixed, translating the schema name from `sales` into `mysql_sales`.

- `--opt tablePrefix`

  A prefix to be placed in front of all schemas. For example, if called with `tablePrefix` set to `mysql_`:

  ```shell```
  shell> ddlscan ... -opt tablePrefix mysql_
  ```shell```

  The table name will be prefixed, translating the tablenname from `sales` into `mysql_sales`. 

309
8.10.2.5. ddl-mysql-hive-0.10-staging.vm

Staging tables within Hive define the original table columns with additional columns to track the operation type, sequence number, timestamp and unique key for each row. For example, the table `sales` in MySQL:

```
mysql> describe sales;
+----------+----------+------+-----+---------+----------------+
| Field    | Type     | Null | Key | Default | Extra          |
+----------+----------+------+-----+---------+----------------+
| id       | int(11)  | NO   | PRI | NULL    | auto_increment |
| salesman  | char(20) | YES  |     | NULL    |                |
| planet    | char(20) | YES  |     | NULL    |                |
| value     | float    | YES  |     | NULL    |                |
+----------+----------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Generates the following Hive-compatible DDL when using this template:

```
shell> ddlscan -user tungsten -url 'jdbc:mysql://tr-hadoop1:13306/test' -pass password \
-template ddl-mysql-hive-0.10-staging.vm -db test
-- SQL generated on Thu Sep 11 12:31:45 BST 2014 by Tungsten ddlscan utility
-- url = jdbc:mysql://tr-hadoop1:13306/test
-- user = tungsten
-- dbName = test

DROP TABLE IF EXISTS test.stage_xxx_sales;
CREATE EXTERNAL TABLE test.stage_xxx_sales
(
  tungsten_opcode STRING,
  tungsten_seqno INT,
  tungsten_row_id INT,
  tungsten_commit_timestamp TIMESTAMP,
  id INT,
  salesman STRING,
  planet STRING,
  value DOUBLE
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ' ' ESCAPED BY '\'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE LOCATION '/user/tungsten/staging/test/sales';
```

Wherever possible, the closest Hive equivalent datatype is used for each source datatype, see `ddl-mysql-hive-0.10.vm` for more information.

8.10.2.6. ddl-mysql-hive-metadata.vm

The Hadoop tools require information about the schema in JSON format so that the table names and primary key information can be used when materializing data from the staging tables into the base tables. This template generates that information in JSON format:

```
shell> ddlscan -user tungsten -url 'jdbc:mysql://tr-hadoop1:13306/test' -pass password \n-template ddl-mysql-hive-metadata.vm -db test

{
  "tables": [
    {"schema": "test", "name": "sales", "keys": ["id"], "columns": [
      {"name": "id", "type": "INT"},
      {"name": "salesman", "type": "STRING"},
      {"name": "planet", "type": "STRING"},
      {"name": "value", "type": "DOUBLE"}
    ]
  ]
}
```

8.10.2.7. ddl-mysql-oracle.vm

When translating MySQL tables to Oracle compatible schema, the following datatypes are migrated to their closest Oracle equivalent:

<table>
<thead>
<tr>
<th>MySQL Datatype</th>
<th>Oracle Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>NUMBER(10, 0)</td>
</tr>
</tbody>
</table>
The following additional transformations happen automatically:

- Table names are translated to uppercase.
- Column names are translated to uppercase.
- If a column name is a reserved word in Oracle, then the column name has an underscore character appended (for example, `TABLE` becomes `TABLE_`).

In addition to the above translations, errors will be raised for the following conditions:

- If the table name starts with a number.
- If the table name exceeds 30 characters in length.
- If the table name is a reserved word in Oracle.

Warnings will be raised for the following conditions:

- If the column or column name started with a number.
- If the column name exceeds 30 characters in length, the column name will be truncated.
- If the column name is a reserved word in Oracle.

8.10.2.8. `ddl-mysql-oracle-cdc.vm`

The `ddl-mysql-oracle-cdc.vm` template generates identical tables in Oracle, from their MySQL equivalent, but with additional columns for CDC capture. For example:

```shell
ddlscan -user tungsten -url 'jdbc:mysql://tr-hadoop1:13306/test' -pass password \   -template ddl-mysql-oracle-cdc.vm -db test \\ SQL generated on Thu Sep 11 13:17:05 BST 2014 by ./ddlscan utility of Tungsten url = jdbc:mysql://tr-hadoop1:13306/test user = tungsten dbName = test */
```

```sql
DROP TABLE test.sales;
CREATE TABLE test.sales
(
    id NUMBER(10, 0) NOT NULL,
    salesman CHAR,
    planet CHAR,
    value FLOAT,
    `AuditDate` DATE
)
```

<table>
<thead>
<tr>
<th>MySQL Datatype</th>
<th>Oracle Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT</td>
<td>NUMBER(19, 0)</td>
</tr>
<tr>
<td>TINYINT</td>
<td>NUMBER(3, 0)</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>NUMBER(5, 0)</td>
</tr>
<tr>
<td>MEDIUMINT</td>
<td>NUMBER(7, 0)</td>
</tr>
<tr>
<td>DECIMAL(x,y)</td>
<td>NUMBER(x, y)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
</tr>
<tr>
<td>CHAR(n)</td>
<td>CHAR(n)</td>
</tr>
<tr>
<td>VARCHAR(n)</td>
<td>VARCHAR(n) (n &lt; 2000), CLOB n &gt; 2000)</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
</tr>
<tr>
<td>DATETIME</td>
<td>DATE</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>DATE</td>
</tr>
<tr>
<td>TEXT</td>
<td>CLOB</td>
</tr>
<tr>
<td>BLOB</td>
<td>BLOB</td>
</tr>
<tr>
<td>ENUM(...)</td>
<td>VARCHAR(255)</td>
</tr>
<tr>
<td>ENUM(...)</td>
<td>VARCHAR(4000)</td>
</tr>
<tr>
<td>BIT(1)</td>
<td>NUMBER(1)</td>
</tr>
</tbody>
</table>
For information on the datatypes translated, see `ddl-mysql-oracle.vn`.

8.10.2.9. `ddl-mysql-redshift.vn`

The `ddl-mysql-redshift.vn` template generates DDL for Amazon Redshift tables from MySQL schemas. For example:

```sql
CREATE TABLE test.all_mysql_types
(
  my_id INT,
  my_bit BOOLEAN /* BIT(1) */, my_smallint SMALLINT /* TINYINT(4) */, my_boolean SMALLINT /* TINYINT(1) */, my_mediumint INT /* MEDIUMINT(9) */, my_int INT,
  my_bigint BIGINT,
  my_decimal_10_5 DECIMAL(10,5),
  my_float FLOAT,
  my_double DOUBLE PRECISION /* DOUBLE */, my_date DATE,
  my_datetime DATETIME,
  my_timestamp TIMESTAMP, my_year YEAR(4) /* ERROR: unrecognized (type=0, length=0) */, my_char_10 CHAR(10), my_varchar_10 VARCHAR(40) /* VARCHAR(10) */,
  my_tinytext VARCHAR(65535) /* WARN: MySQL TINYTEXT translated to max VARCHAR */, my_text VARCHAR(65535) /* WARN: MySQL TEXT translated to max VARCHAR */, my_longtext VARCHAR(65535) /* WARN: MySQL LONGTEXT translated to max VARCHAR */, my_enum_abc VARCHAR(1) /* ENUM('A','B','C') */, my_set_def VARCHAR(65535) /* SET('D','E','F') */,
  PRIMARY KEY (my_id)
);
```

Columns are translated as follows:

<table>
<thead>
<tr>
<th>Oracle Datatype</th>
<th>Redshift Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT</td>
<td>BIGINT</td>
</tr>
<tr>
<td>BINARY</td>
<td>BINARY, CHAR in 5.2.1 and later</td>
</tr>
<tr>
<td>BIT(1)</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>BIT</td>
<td>CHAR</td>
</tr>
<tr>
<td>BLOB</td>
<td>VARBINARY, VARCHAR in 5.2.1 and later</td>
</tr>
<tr>
<td>CHAR</td>
<td>CHAR</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
</tr>
<tr>
<td>DATETIME</td>
<td>DATETIME</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>DOUBLE PRECISION</td>
</tr>
<tr>
<td>ENUM</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
</tr>
<tr>
<td>INT</td>
<td>INT</td>
</tr>
<tr>
<td>LONGBLOB</td>
<td>VARBINARY, VARCHAR in 5.2.1 and later</td>
</tr>
<tr>
<td>LONGTEXT</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>MEDIUMBLOB</td>
<td>VARBINARY, VARCHAR in 5.2.1 and later</td>
</tr>
<tr>
<td>MEDIUMINT</td>
<td>INT</td>
</tr>
<tr>
<td>MEDIUMTEXT</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>SET</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>TEXT</td>
<td>VARCHAR</td>
</tr>
</tbody>
</table>
In addition to these explicit changes, the following other considerations are taken into account:

- When translating the DDL for `CHAR` and `VARCHAR` columns, the actual column size is increased by a factor of four. This is because Redshift tables always stored data using 32-bit UTF characters and column sizes are in bytes, not characters. Therefore a `CHAR(20)` column is created as `CHAR(80)` within Redshift.
- `TEXT` columns are converted to a Redshift `VARCHAR` of 65535 in length (the maximum allowed).
- `BLOB` columns are converted to a Redshift `VARBINARY` of 65000 in length (the maximum allowed).
- `BIT` columns with a size of 1 are converted to Redshift `BOOLEAN` columns, larger sizes are converted to `CHAR` columns of 64 bytes in length.
- `TIME` columns are converted to a Redshift `VARCHAR` of 17 bytes in length since no explicit `TIME` type exists.

### 8.10.2.10. `ddl-mysql-redshift-staging.vm`

The `ddl-mysql-redshift-staging.vm` template generates DDL for Amazon Redshift tables from MySQL schemas. For example:

```sql
CREATE TABLE test.stage_xxx_all_mysql_types
(
  tungsten_opcode CHAR(2),
  tungsten_seqno INT,
  tungsten_row_id INT,
  tungsten_commit_timestamp TIMESTAMP,
  my_id INT,
  my_bit BOOLEAN /* BIT(1) */,
  my_tinyint SMALLINT /* TINYINT(4) */,
  my_boolean SMALLINT /* TINYINT(1) */,
  my_smallint SMALLINT,
  my_mediumint INT /* MEDIUMINT(9) */,
  my_int INT,
  my_bigint BIGINT,
  my_decimal_10_5 DECIMAL(10,5),
  my_float FLOAT,
  my_double DOUBLE PRECISION /* DOUBLE */,
  my_date DATE,
  my_datetime DATETIME,
  my_timestamp TIMESTAMP,
  my_time VARCHAR(17) /* WARN: no pure TIME type in Redshift */,  
  my_year YEAR(4) /* ERROR: unrecognized (type=0, length=0) */,
  PRIMARY KEY (tungsten_opcode, tungsten_seqno, tungsten_row_id)
);```

The actual translation of datatypes is identical to that found in `ddl-mysql-redshift.vm`.

### 8.10.2.11. `ddl-mysql-vertica.vm`

The `ddl-mysql-vertica.vm` template generates DDL for generating tables within an HP Vertica database from an existing MySQL database schema. For example:

```shell
dlsc -user tungsten -url 'jdbc:mysql://tr-hadoop1:13306/test' -pass password \
-template ddl-mysql-vertica.vm -db test
```

SQL generated on Thu Sep 11 14:20:14 BST 2014 by `dlsc` utility of Tungsten

url = jdbc:mysql://tr-hadoop1:13306/test
Because Vertica does not explicitly support primary keys, a default projection for the key order is created based on the primary key of the source table.

The templates translate different datatypes as follows:

<table>
<thead>
<tr>
<th>MySQL Datatype</th>
<th>Vertica Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATETIME</td>
<td>DATETIME</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
</tr>
<tr>
<td>TIME</td>
<td>TIME</td>
</tr>
<tr>
<td>TINYINT</td>
<td>TINYINT</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>MEDIUMINT</td>
<td>INT</td>
</tr>
<tr>
<td>INT</td>
<td>INT</td>
</tr>
<tr>
<td>BIGINT</td>
<td>INT</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>CHAR</td>
<td>CHAR</td>
</tr>
<tr>
<td>BINARY</td>
<td>BINARY</td>
</tr>
<tr>
<td>VARBINARY</td>
<td>VARBINARY</td>
</tr>
<tr>
<td>TEXT, TINYTEXT, MEDIUMTEXT, LONGTEXT</td>
<td>VARCHAR(65000)</td>
</tr>
<tr>
<td>BLOB, TINYBLOB, MEDIUMBLOB, LONGBLOB</td>
<td>VARBINARY(65000)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>DOUBLE PRECISION</td>
</tr>
<tr>
<td>ENUM</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>SET</td>
<td>VARCHAR(4000)</td>
</tr>
<tr>
<td>BIT(1)</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>BIT</td>
<td>CHAR(64)</td>
</tr>
</tbody>
</table>

In addition, the following considerations should be taken into account:

- **DECIMAL**: MySQL type is not supported.
- **TEXT**: Types in MySQL are converted to a VARCHAR in Vertica of the maximum supported size.
- **BLOB**: Types in MySQL are converted to a VARBINARY in Vertica of the maximum supported size.
- **SET**: Types in MySQL are converted to a VARCHAR in Vertica of 4000 characters, designed to work in tandem with the `settostring` filter.
- **ENUM**: Types in MySQL are converted to a VARCHAR in Vertica of the size of the longest ENUM value, designed to work in tandem with the `enum-tostring` filter.

8.10.2.12. **ddl-mysql-vertica-staging.vm**

The `ddl-mysql-vertica-staging.vm` template generates DDL for HP Vertica staging tables. These include the full table definition, in addition to three columns used to define the staging data, including the operation code, sequence number and unique row ID. For example:

```bash
shell> ddlscan -user tungsten -url 'jdbc:mysql://tr-hadoop1:13306/test' -pass password \   -template ddl-mysql-vertica-staging.vm -db test
```
The `ddl-oracle-mysql.vm` template generates the DDL required to create a schema within MySQL based on the existing Oracle schema. For example:

```
shell> ddlscan -service sales -template ddl-oracle-mysql.vm -db sales
```

Columns are translated as follows:

<table>
<thead>
<tr>
<th>Oracle Datatype</th>
<th>MySQL Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>DATETIME</td>
</tr>
<tr>
<td>NUMBER(8)</td>
<td>NUMERIC</td>
</tr>
<tr>
<td>NUMBER(n) where n &lt; 19</td>
<td>INT</td>
</tr>
<tr>
<td>NUMBER(n) where n &gt; 19</td>
<td>BIGINT</td>
</tr>
<tr>
<td>NUMBER(n) where n &lt; 3</td>
<td>TINYINT</td>
</tr>
<tr>
<td>NUMBER(n) where n &lt; 5</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>NUMBER(n) where n &lt; 7</td>
<td>MEDIUMINT</td>
</tr>
<tr>
<td>NUMBER(n) where n &lt; 10</td>
<td>INT</td>
</tr>
<tr>
<td>NUMBER(n) where n &lt; 19</td>
<td>BIGINT</td>
</tr>
<tr>
<td>NUMBER</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>LONG</td>
<td>LONGTEXT</td>
</tr>
<tr>
<td>BFILE</td>
<td>VARCHAR(1024)</td>
</tr>
</tbody>
</table>
The following additional transformations happen automatically:

- If a column name is a reserved word in MySQL, then the column name has an underscore character appended (for example, `TABLE` becomes `TABLE_`).

An error is raised in the following conditions:

- If the size of a `FLOAT` is larger than 53 points of precision.

8.10.2.14. `ddl-oracle-mysql-pk-only.vm`

The `ddl-oracle-mysql-pk-only.vm` template generates alter table statements to add the primary key, as determined from the Oracle primary key or index information. For example:

```sql
ALTER TABLE hadoop.sample ADD PRIMARY KEY (ID);
```

Note that it does not generate table DDL, only statements to alter existing tables with primary key information.

8.11. The `deployall` Command

The `deployall` tool installs the required startup scripts into the correct location so that all required services can be automatically started and stopped during the startup and shutdown of your server.

To use, the tool should be executed with superuser privileges, either directly using `sudo`, or by logging in as the superuser and running the command directly:

The script will automatically detect the initialization system in use (systemd or initd) and prefer systemd when both are available.

```sh
deployall
```

The startup scripts are added to the correct run levels to enable operation during standard startup and shutdown levels.

**Note**

For systemd configurations only:

For continuity of service reasons, the `deployall` script does NOT restart individual components when called, it will only install systemd scripts. This implies that, right after a call to deployall and before host restart, the system will stay in a mixed mode where systemd scripts are in place but components were started without systemd, so won’t be controllable by it.

In order to align the configuration, you will need to run:

```sh
component stop sysd
component stop tcomponent
```

For example:

```sh
component stop sysd
component stop tcomponent
component stop sysd
component stop tcomponent
```

See Section 4.3, "Configuring Startup on Boot".
8.12. The dsctl Command

The dsctl command provides a simplified interface into controlling the datasource within a replication scenario to set the current replication position. Because dsctl uses the built-in datasource connectivity of the replicator, differences in the storage and configuration of the current replicator metadata and position can be controlled without resorting to updating the corresponding database directly.

The command is driven by a number of command-specific instructions to get or set the datasource position.

### Table 8.22. dsctl Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>get</td>
<td>Return the available position information</td>
</tr>
<tr>
<td>help</td>
<td>Print the help display</td>
</tr>
<tr>
<td>reset</td>
<td>Clear the datasource position information</td>
</tr>
<tr>
<td>set</td>
<td>Set the position</td>
</tr>
</tbody>
</table>

These must be used in conjunction with one of the following options to select the required datasources or service:

### Table 8.23. dsctl Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-conf</td>
<td>Path to the static services properties file</td>
</tr>
<tr>
<td>-ds</td>
<td>Name of the datasource</td>
</tr>
<tr>
<td>-service</td>
<td>Name of the replication service to get information from</td>
</tr>
</tbody>
</table>

If more than one service or datasource has been configured, one of these options must be used to select the service. Otherwise, by default dsctl will use the corresponding configured service.

8.12.1. dsctl get Command

### Table 8.24. dsctl Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-ascmd</td>
<td>Generates the command required to set the datasource to the current position</td>
</tr>
</tbody>
</table>

Returns the current datasource status and position, returning the information as a JSON string. The example below has been formatted for clarity:

```shell
dsctl
[
  {
    "last_frag" : true,
    "applied_latency" : 1,
    "extract_timestamp" : "2015-01-21 21:46:57.0",
    "eventid" : "mysql-bin.88000014:8800000000005645;1",
    "source_id" : "tr-11",
    "epoch_number" : 22,
    "update_timestamp" : "2015-01-21 21:46:58.0",
    "task_id" : 0,
    "shard_id" : "tungsten_alpha",
    "seqno" : 22,
    "fragno" : 0
  }
]
```

When the `-ascmd` option is used, the information is output in form of a command:

```shell
dsctl get -ascmd
dsctl set "seqno 17 epoch 11 event-id 'mysql-bin.8800002:88000008140315777;1' source-id 'ubuntu'
```

If the `-reset` is used, then the generated command also includes the option. For example:

```shell
dsctl get -ascmd -reset
ndsctl set "seqno 17 epoch 11 event-id 'mysql-bin.8800002:88000008140315777;1' source-id 'ubuntu' -reset
```
8.12.2. dsctl set Command

Table 8.25. dsctl Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-epoch</td>
<td>Epoch Number</td>
</tr>
<tr>
<td>-event-id</td>
<td>Source Event ID</td>
</tr>
<tr>
<td>-reset</td>
<td>Resets the datasources before performing set operation</td>
</tr>
<tr>
<td>-seqno</td>
<td>Sequence number</td>
</tr>
<tr>
<td>-source-id</td>
<td>Source ID</td>
</tr>
</tbody>
</table>

Sets the current replicator position. When using this option, the -seqno, -epoch, -event-id, and -source-id options must be specified to set the corresponding values in the replicator.

For example:

```
shell> dsctl set -seqno 22 -epoch 22 -event-id "mysql-bin.000014:0000000000005645;-1" -source-id tr-11
Service "alpha" datasource "global" position was set to: seqno=22 epoch_number=22 » eventid=mysql-bin.000014:0000000000005645;-1 source_id=tr-11
```

When used with the -reset, the datasource is reset before the set operation:

```
shell> dsctl set -seqno 17 -epoch 11 -event-id "mysql-bin.000082:000000014031577;1" -source-id "ubuntu" -reset
Service "alpha" datasource "global" catalog information cleared
Service "alpha" datasource "global" position was set to: seqno=17 epoch_number=11 » eventid=mysql-bin.000082:000000014031577;1 source_id=ubuntu
```

Adding the -reset option to the dsctl get -ascmd command also adds the option to the generated command:

```
shell> dsctl get -ascmd -reset
```

```
shell> dsctl set -seqno 17 -epoch 11 -event-id "mysql-bin.000082:000000014031577;1" -source-id "ubuntu" -reset
```

8.12.3. dsctl reset Command

Clears the current replicator status and position information:

```
shell> dsctl reset
Service "alpha" datasource "global" catalog information cleared
```

8.12.4. dsctl help Command

Displays the current help text:

```
shell> dsctl help
Datasource Utility
Syntax: dsctl [conf|service] [-ds name] [operation]
Configuration (required if there’s more than one service):
-conf path       - Path to a static-<svc>.properties file
OR
-service name    - Name of a replication service to get datasource configuration from
[ -ds name]      - Name of the datasource (default: global)
Options:
get              - Return all available position information
set -seqno ###   - Set position (all four parameters required)
-epoch ###       - Source Event ID
-event-id AAAAAAAAA.######:#######
-source-id AAA.AAA.AAA
[ -reset]        - Optionally reset the information first
reset            - Clear datasource position information
help             - Print this help display
```

8.13. env.sh Script

After installation, the env.sh can be used to setup the local environment, such as appending to the local $PATH.

If --profile-script is set during installation, then the local profile script will also be updated to ensure the env.sh file is loaded at login of the OS user.

```
shell> cat .bash_profile
```

318
8.14. The manager Command

The manager is the wrapper script that handles the execution of the manager service.

Table 8.26. manager Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condrestart [319]</td>
<td>Restart only if already running</td>
</tr>
<tr>
<td>console [319]</td>
<td>Launch in the current console (instead of a daemon)</td>
</tr>
<tr>
<td>dump [320]</td>
<td>Request a Java thread dump (if manager is running)</td>
</tr>
<tr>
<td>install [320]</td>
<td>Install the service to automatically start when the system boots</td>
</tr>
<tr>
<td>remove [320]</td>
<td>Remove the service from starting during boot</td>
</tr>
<tr>
<td>restart [320]</td>
<td>Stop manager if already running and then start</td>
</tr>
<tr>
<td>start [320]</td>
<td>Start in the background as a daemon process</td>
</tr>
<tr>
<td>status [320]</td>
<td>Query the current status</td>
</tr>
<tr>
<td>stop [320]</td>
<td>Stop if running (whether as a daemon or in another console)</td>
</tr>
</tbody>
</table>

These commands and options are described below:

- **condrestart [319]**

Restart the manager, only if it is already running. This can be useful to use when changing configuration or performing database management within automated scripts, as the manager will be only be restart if it was previously running.

For example, if the manager is running, manager condrestart operates as manager restart:

```
shell> manager condrestart
Stopping Tungsten Manager Service...
Waiting for Tungsten Manager Service to exit...
Stopped Tungsten Manager Service.
Starting Tungsten Manager Service...
Waiting for Tungsten Manager Service......
running: PID: 26646
```

However, if not already running, the operation does nothing:

```
shell> manager condrestart
Stopping Tungsten Manager Service...
Tungsten Manager Service was not running.
```

- **console [319]**

Launch in the current console (instead of a daemon)
8.15. The multi_trepctl Command

The multi_trepctl command provides unified status and operation support across your Tungsten Clustering installation across multiple hosts without the need to run the trepctl command across multiple hosts and/or services individually.

```
multi_trepctl [-by-service] [ --fieldsappliedLastSeqNoappliedLatencyhostroleservicenamestate ] [ --host, --hostself ]
list [ --outputjsonlistyaml ] [ --path, --paths ] [ --role, --roles ]
run [ --service, --servicesself ] [ --skip-headers ] [ --sort-by ]
```

The default operation, with no further command-line commands or arguments displays the status of all the hosts and services identified as related to the current host. In a typical single-service deployment, the command outputs the status of all services by determining the relationship between hosts connected to the default service:
On a server with multiple services, information is output for each service and host:

```
shell> multi_trepctl
| host   | servicename | role   | state         | appliedlastseqno | appliedlatency |
| tr-ms1 | alpha       | master | ONLINE        | 54               | 0.867         |
| tr-ms2 | alpha       | slave  | ONLINE        | 54               | 1.945         |
| tr-ms3 | alpha       | slave  | ONLINE        | 54               | 42.891        |
```

```
On a server with multiple services, information is output for each service and host:

shell> multi_trepctl
| host  | servicename | role   | state         | appliedlastseqno | appliedlatency |
| east1 | east        | master | ONLINE        | 53               | 0.000         |
| east1 | west        | slave  | OFFLINE:ERROR | -1               | -1.000        |
| west1 | east        | master | ONLINE        | 294328           | 0.319         |
| west1 | west        | slave  | ONLINE        | 53               | 119.834       |
| west2 | west        | master | ONLINE        | 231595           | 0.316         |
| west2 | east        | slave  | ONLINE        | 53               | 181.128       |
| west3 | east        | slave  | ONLINE        | 53               | 204.790       |
| west3 | west        | slave  | ONLINE        | 231595           | 22.895        |
```

### 8.15.1. `multi_trepctl` Options

The `multi_trepctl` tool provides a number of options that control the information and detail output when the command is executed.

#### Table 8.27. `multi_trepctl` Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--by-service [321]</td>
<td>Sort the output by the service name</td>
</tr>
<tr>
<td>--fields [321]</td>
<td>Fields to be output during summary</td>
</tr>
<tr>
<td>--host [321], --hosts [321]</td>
<td>Host or hosts on which to limit output</td>
</tr>
<tr>
<td>--output [322]</td>
<td>Specify the output format</td>
</tr>
<tr>
<td>--paths [322], --path [322]</td>
<td>Directory or directories to check when looking for tools</td>
</tr>
<tr>
<td>--role [322], --roles [322]</td>
<td>Role or roles on which to limit output</td>
</tr>
<tr>
<td>--service [323], --services [323]</td>
<td>Service or services on which to limit output</td>
</tr>
<tr>
<td>--skip-headers [323]</td>
<td>Skip the headers</td>
</tr>
<tr>
<td>--sort-by [323]</td>
<td>Sort by a specified field</td>
</tr>
</tbody>
</table>

Where:

- `--by-service [321]`

Order the output according to the service name and role within the service:

```
shell> multi_trepctl --by-service
| host   | servicename | role   | state         | appliedlastseqno | appliedlatency |
| east1 | east        | master | ONLINE        | 64               | 59.388        |
| west1 | east        | master | ONLINE        | 64               | 60.889        |
| west2 | east        | slave  | ONLINE        | 64               | 68.970        |
| west3 | east        | slave  | ONLINE        | 64               | 61.097        |
| west1 | west        | master | ONLINE        | 294328           | 0.319         |
| west2 | west        | master | ONLINE        | 231595           | 0.316         |
| east1 | west        | slave  | OFFLINE:ERROR | 1                | -1.000        |
| west3 | west        | slave  | ONLINE        | 231595           | 22.895        |
```

- `--fields [321]`

Limited the output to the specified list of fields from the output of fields output by `trepctl`. For example, to limit the output to the host, role, and `appliedlatency`:

```
shell> multi_trepctl --fields=host,role,appliedlatency
| host   | role   | appliedlatency |
| tr-ms1 | master | 0.324         |
| tr-ms2 | slave  | -1.000        |
```

- `--host [321], --hosts [321]`

Limit the output to the host, or a comma-separated list of hosts specified. For example:

```
shell> multi_trepctl --hosts=tr-ms1,tr-ms3
| host   | servicename | role   | state         | appliedlastseqno | appliedlatency |
| tr-ms1 | alpha       | master | ONLINE        | 2022             | 0.324         |
```

321
Command-line Tools

<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>db1</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>3</td>
<td>0.419</td>
</tr>
<tr>
<td>db2</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>3</td>
<td>0.481</td>
</tr>
<tr>
<td>db3</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>3</td>
<td>0.484</td>
</tr>
<tr>
<td>db4</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>4</td>
<td>0.466</td>
</tr>
<tr>
<td>db5</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>4</td>
<td>0.451</td>
</tr>
<tr>
<td>db6</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>4</td>
<td>0.496</td>
</tr>
</tbody>
</table>

- **--output [322]**
  
  Specify the output format.

<table>
<thead>
<tr>
<th>Option</th>
<th>--output [322]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Specify the output format</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Default</td>
<td>info</td>
</tr>
<tr>
<td>Valid Values</td>
<td>json</td>
</tr>
<tr>
<td></td>
<td>list</td>
</tr>
<tr>
<td></td>
<td>name</td>
</tr>
<tr>
<td></td>
<td>tab</td>
</tr>
<tr>
<td></td>
<td>yaml</td>
</tr>
</tbody>
</table>

For example, to output the current status in JSON format:

```shell
shell> multi_trepctl --output json
{
    "appliedlastseqno": 2322,
    "appliedlatency": 0.524,
    "host": "tr-ms1",
    "role": "master",
    "servicename": "alpha",
    "state": "ONLINE"
},
{
    "appliedlastseqno": 2322,
    "appliedlatency": 0.0,
    "host": "tr-ms2",
    "role": "slave",
    "servicename": "alpha",
    "state": "ONLINE"
},
{
    "appliedlastseqno": -1,
    "appliedlatency": -1.0,
    "host": "tr-ms3",
    "role": "slave",
    "servicename": "alpha",
    "state": "OFFLINE:ERROR"
}
```

- **--path [322], --paths [322]**
  
  Limit the search for trepctl to the specified path or comma-separated list of paths. On a deployment with multiple services, the output will be limited by the services installed within the specified directories:

```shell
shell> multi_trepctl --path /opt/replicator
<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>db1</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>3</td>
<td>0.419</td>
</tr>
<tr>
<td>db2</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>3</td>
<td>0.481</td>
</tr>
<tr>
<td>db3</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>3</td>
<td>0.484</td>
</tr>
<tr>
<td>db4</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>4</td>
<td>0.466</td>
</tr>
<tr>
<td>db5</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>4</td>
<td>0.451</td>
</tr>
<tr>
<td>db6</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>4</td>
<td>0.496</td>
</tr>
</tbody>
</table>
```

This is also useful when control of cross-site replicators is desired in MSMM topologies prior to v6.0.0.

For example, take all cross-site replicators offline:

```shell
shell> multi_trepctl --path /opt/replicator offline
```

To bring all cross-site replicators online:

```shell
shell> multi_trepctl --path /opt/replicator online
```

- **--role [322], --roles [322]**
Command-line Tools

Limit the output to show only the specified role or comma-separated list of roles:

```
shell> multi_trepctl --roles=slave
```

```
<table>
<thead>
<tr>
<th>host</th>
<th>servicename</th>
<th>role</th>
<th>state</th>
<th>appliedlastseqno</th>
<th>appliedlatency</th>
</tr>
</thead>
<tbody>
<tr>
<td>tr-ms2</td>
<td>alpha</td>
<td>slave</td>
<td>ONLINE</td>
<td>2322</td>
<td>8.000</td>
</tr>
<tr>
<td>tr-ms3</td>
<td>alpha</td>
<td>slave</td>
<td>ONLINE:ERROR</td>
<td>-1</td>
<td>-1.000</td>
</tr>
</tbody>
</table>
```

- `--service` [323], `--services` [323]

Limit the output to the specified service or comma-separated list of services:

```
shell> multi_trepctl --service=east
```

```
<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>8.000</td>
</tr>
<tr>
<td>west1</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>53</td>
<td>119.914</td>
</tr>
<tr>
<td>west2</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>53</td>
<td>181.128</td>
</tr>
<tr>
<td>west3</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>53</td>
<td>284.799</td>
</tr>
</tbody>
</table>
```

- `--skip-headers` [323]

Prevents the generation of the headers when generating the list output format:

```
shell> multi_trepctl --skip-headers
```

```
| tr-ms1 | alpha | master | ONLINE        | 2322 | 0.524 |
| tr-ms2 | alpha | slave  | ONLINE        | 2322 | 0.000 |
| tr-ms3 | alpha | slave  | ONLINE:ERROR  | -1   | -1.000 |
```

- `--sort-by` [323]

Sort by the specified fieldname. For example, to sort the output by the latency:

```
shell> multi_trepctl --sort-by appliedlatency
```

```
<table>
<thead>
<tr>
<th>host</th>
<th>servicename</th>
<th>role</th>
<th>state</th>
<th>appliedlastseqno</th>
<th>appliedlatency</th>
</tr>
</thead>
<tbody>
<tr>
<td>tr-ms3</td>
<td>alpha</td>
<td>slave</td>
<td>ONLINE:ERROR</td>
<td>-1</td>
<td>-1.000</td>
</tr>
<tr>
<td>tr-ms2</td>
<td>alpha</td>
<td>slave</td>
<td>ONLINE</td>
<td>2322</td>
<td>8.000</td>
</tr>
<tr>
<td>tr-ms1</td>
<td>alpha</td>
<td>master</td>
<td>ONLINE</td>
<td>2322</td>
<td>8.000</td>
</tr>
</tbody>
</table>
```

8.15.2. multi_trepctl Commands

The default operational mode is for `multi_trepctl list` to output the status. A specific mode can be also be specified on the command-line.

Table 8.29. `multi_trepctl` Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>List the information about each service</td>
</tr>
<tr>
<td>run</td>
<td>Run the specified trepctl command on all hosts/services</td>
</tr>
</tbody>
</table>

In addition to the two primary commands, `multi_trepctl` can execute commands that would normally be applied to `trepctl`, running them on each selected host, service or directory according to the options. The output format and expectation is controlled through the `list` and `run` commands.

For example:

```
shell> multi_trepctl status
```

Outputs the long form of the status information [as per `trepctl status`] for each identified host.

8.15.2.1. `multi_trepctl` list Command

The `multi_trepctl list` mode is the default mode for `multi_trepctl` and outputs the current status across all hosts and services as a table:

```
shell> multi_trepctl
```

```
<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>host1</td>
<td>firstrep</td>
<td>master</td>
<td>ONLINE:ERROR</td>
<td>-1</td>
<td>-1.008</td>
</tr>
<tr>
<td>host2</td>
<td>firstrep</td>
<td>slave</td>
<td>GOING-ONLINE:SYNCHRONIZING</td>
<td>5271</td>
<td>4656.264</td>
</tr>
<tr>
<td>host3</td>
<td>firstrep</td>
<td>slave</td>
<td>ONLINE:ERROR</td>
<td>-1</td>
<td>-1.008</td>
</tr>
<tr>
<td>host4</td>
<td>firstrep</td>
<td>slave</td>
<td>ONLINE:ERROR</td>
<td>-1</td>
<td>-1.008</td>
</tr>
</tbody>
</table>
```

Or selected hosts and services if options are specified. For example, to get the status only for `host1` and `host2`:

```
shell> multi_trepctl --hosts=host1,host2
```

```
<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>host1</td>
<td>firstrep</td>
<td>master</td>
<td>ONLINE</td>
<td>5277</td>
<td>0.476</td>
</tr>
</tbody>
</table>
```
The `multi_trepctl` command implies that the status or information is being output from each of the commands executed on the remote hosts and services.

### 8.15.2.2. multi_trepctl run Command

The `multi_trepctl run` command can be used where the output of the corresponding `trepctl` command cannot be formatted into a convenient list. For example, to execute a backup on every host within a deployment:

```bash
cshell> multi_trepctl run backup
```

The same filters and host or service selection can also be made:

```bash
cshell> multi_trepctl run backup --hosts=host1,host2,host3
```

Return from the command will only take place when remote commands on each host have completed and returned.

### 8.16. The query Command

#### Table 8.30. `query` Common Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-conf PATH</code></td>
<td>Configuration file that contains values for connection properties (url, user and password)</td>
</tr>
<tr>
<td><code>-file PATH</code></td>
<td>File containing the SQL commands to run. If missing, read SQL commands from STDIN</td>
</tr>
<tr>
<td><code>-password</code></td>
<td>Prompt for password</td>
</tr>
<tr>
<td><code>-url JDBCURL</code></td>
<td>JDBC url of the database to connect to</td>
</tr>
<tr>
<td><code>-user USER</code></td>
<td>User used to connect to the database</td>
</tr>
</tbody>
</table>

The `query` command line tool can be used to issue SQL statements against a database.

The queries can either be entered via STDIN, or read in from a text file.

The following example shows a SELECT statement issued via STDIN:

```bash
cshell> query -url "jdbc:mysql:thin://db2:13306/" -user tungsten -password Enter password: ********
select * from tungsten_nyc.trep_commit_seqno;

[ {
  "statement":"select * from tungsten_nyc.trep_commit_seqno;","rc":0,"results":
  [{
    "task_id":0,
    "seqno":1,
    "fragno":0,
    "last_frag":1,
    "source_id":"db1",
    "epoch_number":1,
    "eventid":"mysql-bin.000002:0000000000000879;-1",
    "applied_latency":0,
    "update_timestamp":"2019-06-28 10:44:20.0",
    "shard_id":"tungsten_nyc",
    "extract_timestamp":"2019-06-28 10:44:19.0"
  }]
}]
```

324
8.17. The replicator Command

The replicator is the wrapper script that handles the execution of the replicator service.

Table 8.31. replicator Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condrestart [325]</td>
<td>Restart only if already running</td>
</tr>
<tr>
<td>console [325]</td>
<td>Launch in the current console (instead of a daemon)</td>
</tr>
<tr>
<td>dump [325]</td>
<td>Request a Java thread dump (if replicator is running)</td>
</tr>
<tr>
<td>install [325]</td>
<td>Install the service to automatically start when the system boots</td>
</tr>
<tr>
<td>remove [326]</td>
<td>Remove the service from starting during boot</td>
</tr>
<tr>
<td>restart [326]</td>
<td>Stop replicator if already running and then start</td>
</tr>
<tr>
<td>start [326]</td>
<td>Start in the background as a daemon process</td>
</tr>
<tr>
<td>status [326]</td>
<td>Query the current status</td>
</tr>
<tr>
<td>stop [326]</td>
<td>Stop if running (whether as a daemon or in another console)</td>
</tr>
</tbody>
</table>

These commands and options are described below:

- **condrestart [325]**

Table 8.32. replicator Commands Options for condrestart [325]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>offline</td>
<td>Start in OFFLINE state</td>
</tr>
</tbody>
</table>

Restart the replicator, only if it is already running. This can be useful to use when changing configuration or performing database management within automated scripts, as the replicator will be only be restart if it was previously running.

For example, if the replicator is running, replicator condrestart operates as replicator restart:

```
shell> replicator condrestart
Stopping Tungsten Replicator Service...
Waiting for Tungsten Replicator Service to exit...
Stopped Tungsten Replicator Service.
Starting Tungsten Replicator Service...
Waiting for Tungsten Replicator Service......
running: PID:26646
```

However, if not already running, the operation does nothing:

```
shell> replicator condrestart
Stopping Tungsten Replicator Service...
Tungsten Replicator Service was not running.
```

- **console [325]**

Table 8.33. replicator Commands Options for console [325]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>offline</td>
<td>Start in OFFLINE state</td>
</tr>
</tbody>
</table>

Launch in the current console (instead of a daemon)

- **dump [325]**

Request a Java thread dump (if replicator is running)

- **install [325]**
Installs the startup scripts for running the replicator at boot. For an alternative method of deploying these start-up scripts, see `deployall`.

`- remove` [326]

Removes the startup scripts for running the replicator at boot. For an alternative method of removing these start-up scripts, see `undeployall`.

`- restart` [326]

Table 8.34. `replicator` Commands Options for `restart` [326]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>offline</td>
<td>Stop and restart in OFFLINE state</td>
</tr>
</tbody>
</table>

**Warning**

Restarts a running replicator temporarily stops and restarts replication.

Stops the replicator, if it is already running, and then restarts it:

```bash
shell> replicator restart
Stopping Tungsten Replicator Service...
Stopped Tungsten Replicator Service.
Starting Tungsten Replicator Service...
Waiting for Tungsten Replicator Service......
running: PID:26248
```

`- start` [326]

Table 8.35. `replicator` Commands Options for `start` [326]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>offline</td>
<td>Start in OFFLINE state</td>
</tr>
</tbody>
</table>

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

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

`- status` [326]

Checks the execution status of the replicator:

```bash
shell> replicator status
Tungsten Replicator Service is running: PID:27015, Wrapper:STARTED, Java:STARTED
```

If the replicator is not running:

```bash
shell> replicator status
Tungsten Replicator Service is not running.
```

This only provides the execution state of the replicator, not the actual state of replication. To get detailed information on the status of replication use `trepctl status`.

`- stop` [326]

Stops the replicator if it is already running:

```bash
shell> replicator stop
Stopping Tungsten Replicator Service...
Waiting for Tungsten Replicator Service to exit...
Stopped Tungsten Replicator Service.
```

If the cluster was configured with `auto-enable=false` [437] then you will need to put each node online individually.

### 8.18. The `setupCDC.sh` Command

The `setupCDC.sh` script configures an Oracle database with the necessary CDC tables to enable heterogeneous replication from Oracle to MySQL.

The script accepts one argument, the filename of the configuration file that will define the CDC configuration. The file accepts the parameters as listed in Table 8.36, "setupCDC.conf Configuration Options".
Table 8.36. **setupCDC.conf** Configuration Options

<table>
<thead>
<tr>
<th>CommandLine Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdc_type [327]</td>
<td>cdc_type [327]</td>
<td>The CDC type to be used to extract data, either synchronous (using triggers) or asynchronous (using log processing).</td>
</tr>
<tr>
<td>delete_publisher [327]</td>
<td>delete_publication [327]</td>
<td>Whether the publisher user should be deleted.</td>
</tr>
<tr>
<td>delete_subscriber [327]</td>
<td>delete_subscriber [327]</td>
<td>Whether the subscriber user should be deleted.</td>
</tr>
<tr>
<td>pub_password [328]</td>
<td>pub_password [328]</td>
<td>The publisher password that will be created for the CDC service.</td>
</tr>
<tr>
<td>pub_user [328]</td>
<td>pub_user [328]</td>
<td>The publisher user that will be created for this CDC service.</td>
</tr>
<tr>
<td>service [328]</td>
<td>service [328]</td>
<td>The service name of the Tungsten Replicator service that will be created.</td>
</tr>
<tr>
<td>source_user [328]</td>
<td>source_user [328]</td>
<td>The source schema user with rights to access the database.</td>
</tr>
<tr>
<td>specific_path [328]</td>
<td>specific_path [328]</td>
<td>The path where the tungsten.tables file is located; the file must be in a shared location accessible by Tungsten Replicator.</td>
</tr>
<tr>
<td>specific_tables [328]</td>
<td>specific_tables [328]</td>
<td>If enabled, extract only the tables defined within a tungsten.tables file.</td>
</tr>
<tr>
<td>sys_pass [329]</td>
<td>sys_pass [329]</td>
<td>The system password to connect to Oracle as SYSDBA.</td>
</tr>
<tr>
<td>sys_user [329]</td>
<td>sys_user [329]</td>
<td>The system user to connect to Oracle as SYSDBA.</td>
</tr>
<tr>
<td>tungsten_user [329]</td>
<td>tungsten_user [329]</td>
<td>The subscriber (Tungsten user) that will subscribe to the changes and read the information from the CDC tables.</td>
</tr>
</tbody>
</table>

Where:

- **cdc_type [327]**

<table>
<thead>
<tr>
<th>Option</th>
<th>cdc_type [327]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Option</td>
<td>cdc_type [327]</td>
</tr>
<tr>
<td>Description</td>
<td>The CDC type to be used to extract data, either synchronous (using triggers) or asynchronous (using log processing).</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td>CDCASYNC</td>
</tr>
<tr>
<td></td>
<td>CDCSYNC</td>
</tr>
</tbody>
</table>

The CDC type to be used to extract data, either synchronous (using triggers) or asynchronous (using log processing).

- **delete_publisher [327]**

<table>
<thead>
<tr>
<th>Option</th>
<th>delete_publisher [327]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Option</td>
<td>delete_publisher [327]</td>
</tr>
<tr>
<td>Description</td>
<td>Whether the publisher user should be deleted.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Whether the publisher user should be deleted.

- **delete_subscriber [327]**

<table>
<thead>
<tr>
<th>Option</th>
<th>delete_subscriber [327]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Option</td>
<td>delete_subscriber [327]</td>
</tr>
<tr>
<td>Description</td>
<td>Whether the subscriber user should be deleted.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td>0</td>
</tr>
</tbody>
</table>
Whether the subscriber user should be deleted.

- **pub_password** [328]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>pub_password [328]</td>
<td>The publisher password that will be created for the CDC service.</td>
<td>string</td>
</tr>
</tbody>
</table>

The publisher password that will be created for the CDC service.

- **pub_user** [328]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>pub_user [328]</td>
<td>The publisher user that will be created for this CDC service.</td>
<td>string</td>
</tr>
</tbody>
</table>

The publisher user that will be created for this CDC service.

- **service** [328]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>service [328]</td>
<td>The service name of the Tungsten Replicator service that will be created.</td>
<td>string</td>
</tr>
</tbody>
</table>

The service name of the Tungsten Replicator service that will be created.

- **source_user** [328]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>source_user [328]</td>
<td>The source schema user with rights to access the database.</td>
<td>string</td>
</tr>
</tbody>
</table>

The source schema user with rights to access the database.

- **specific_path** [328]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>specific_path [328]</td>
<td>The path where the tungsten.tables file is located; the file must be in a shared location accessible by Tungsten Replicator.</td>
<td>string</td>
</tr>
</tbody>
</table>

The path where the tungsten.tables file is located; the file must be in a shared location accessible by Tungsten Replicator.

- **specific_tables** [328]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>specific_tables [328]</td>
<td>If enabled, extract only the tables defined within a tungsten.tables file.</td>
<td>string</td>
</tr>
</tbody>
</table>

Valid Values

<table>
<thead>
<tr>
<th>Valid Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Extract all tables</td>
</tr>
<tr>
<td>1</td>
<td>Use a tables file to select tables</td>
</tr>
</tbody>
</table>
If enabled, extract only the tables defined within a tungsten.tables file.

- **sys_pass** [329]

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>sys_pass [329]</td>
</tr>
<tr>
<td>Description</td>
<td>The system password to connect to Oracle as SYSDBA.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

The system password to connect to Oracle as SYSDBA.

- **sys_user** [329]

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>sys_user [329]</td>
</tr>
<tr>
<td>Description</td>
<td>The system user to connect to Oracle as SYSDBA.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

The system user to connect to Oracle as SYSDBA.

- **tungsten_pwd** [329]

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>tungsten_pwd [329]</td>
</tr>
<tr>
<td>Description</td>
<td>The password for the subscriber user.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

The password for the subscriber user.

- **tungsten_user** [329]

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>tungsten_user [329]</td>
</tr>
<tr>
<td>Description</td>
<td>The subscriber (Tungsten user) that will subscribe to the changes and read the information from the CDC tables.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

The subscriber (Tungsten user) that will subscribe to the changes and read the information from the CDC tables.

To use, supply the name of the configuration file to `setupCDC.sh`:

```
shell> ./setupCDC.sh sample.conf
```

### 8.19. The startall Command

The `startall` will start all configured services within the configured directory:

```
shell> startall
Starting Tungsten Replicator Service...
Waiting for Tungsten Replicator Service....... running: PID:2378
Starting Tungsten Manager Service...
Waiting for Tungsten Manager Service......... running: PID:2722
Starting Tungsten Connector Service...
Waiting for Tungsten Connector Service...... running: PID:2817
```

If a service is already running, then a notification of the current state will be provided:

```
Starting Tungsten Replicator Service...
Tungsten Replicator Service is already running.
```

Note that if any service is not running, and a suitable PID is found, the file will be deleted and the services started, for example:

```
Removed stale pid file:
```
8.20. The stopall Command

The `stopall` command stops all running services if they are already running:

```shell
$ stopall
Stopping Tungsten Connector Service...
Waiting for Tungsten Connector Service to exit...
Stopped Tungsten Connector Service.
Stopping Tungsten Manager Service...
Stopped Tungsten Manager Service.
Stopping Tungsten Replicator Service...
Waiting for Tungsten Replicator Service to exit...
Stopped Tungsten Replicator Service.
```

8.21. The thl Command

The `thl` command provides an interface to the Tungsten High-Level (THL) data, including the ability to view the list of available files, details of the enclosed event information, and the ability to purge Tungsten files to reclaim space on disk beyond the configured log retention policy.

The command supports two command-line options that are applicable to all operations, as shown in Table 8.37, "thl Options".

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-conf path</code></td>
<td>Path to the configuration file containing the required replicator service configuration</td>
</tr>
<tr>
<td><code>-service servicename</code></td>
<td>Name of the service to be used when looking for THL information</td>
</tr>
</tbody>
</table>

For example, to execute a command on a specific service:

```shell
$ thl index -service firstrep
```

Individual operations are selected by use of a specific command to the `thl` command. Supported commands are:

- `index` — obtain a list of available THL files.
- `info` — obtain summary information about the available THL data.
- `list` — list one or more THL events.
- `purge` — purge THL data.
- `help` — get the command help text.

Further information on each of these operations is provided in the following sections.

8.21.1. thl Position Commands

The `thl` command supports a number of position and selection command-line options that can be used to select an individual THL event, or a range of events, to be displayed.

- `-seqno # [330]`

Valid for: `thl list`

Output the THL sequence for the specific sequence number. When reviewing or searching for a specific sequence number, for example when the application of a sequence on a slave has failed, the replication data for that sequence number can be individually viewed. For example:

```shell
$ thl list -seqno 15
SEQ# = 15 / FRAG# = 0 (last frag)
- FILE = thl.data.0000000001
- TIME = 2013-05-02 11:37:00.0
- EPOCH = 7
- EVENTID = mysql-bin.000004:0000000000003345:0
- SOURCEID = host1
- METADATA = [mysql_server_id=1687011;unsafe_for_block_commit;dbms_type=mysql;service=firstrep;shard=cheffy]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [##charset = UTF-8, autocommit = 1, sql_auto_is_null = 0, foreign_key_checks = 0, unique_checks = 0, sql_mode = 'NO_AUTO_VALUE_ON_ZERO', character_set_client = 33, ]
```
collation_connection = 33, collation_server = 8

SCHEMA = cheffy

SQL(0) = CREATE TABLE `access_log` (  
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,  
  `userid` int(10) unsigned DEFAULT NULL,  
  `datetime` int(10) unsigned NOT NULL DEFAULT '0',  
...

If the sequence number selected contains multiple fragments, each fragment will be output. Depending on the content of the sequence number information, the information can be output containing only the header/metadata information or only the table data (row or SQL) that was contained within the fragment. See -headers and -sql for more information.

**Note**

Unsigned integers are displayed and stored in the THL as their negative equivalents, and translated to the correct unsigned type when the data is applied to the target database.

- **-low # [331]** and/or **-high # [331]**
  - **-from # [331]** and/or **-to # [331]**

Valid for: `thl list`, `thl purge`

Specify the start (-low [331]) or end ( -high [331]) of the range of sequence numbers to be output. If only -low [331] is specified, then all sequence numbers from that number to the end of the THL are output. If -high [331] is specified, all sequence numbers from the start of the available log file to the specified sequence number are output. If both numbers are specified, output all the sequence numbers within the specified range.

5.1.0. Starting in 5.1.0, the options -from [331] and -to [331] can be used in place of -low [331] and -high [331] respectively.

For example:

```shell
shell> thl list -low 320
```

Or:

```shell
shell> thl list -from 320
```

Will output all the sequence number fragments from number 320.

```shell
shell> thl list -high 540
```

Or:

```shell
shell> thl list -high 540
```

Will output all the sequence number fragments up to and including 540.

```shell
shell> thl list -low 320 -high 540
```

Or:

```shell
shell> thl list -from 320 -to 540
```

Will output all the sequence number fragments from number 320 up to, and including, sequence number 540.

- **-first [331]**

Valid for: `thl list`, `thl purge`

5.2.0. Starting in 5.2.0, the options -first [331], -first # [331], -last [332], and -last # [332], can be used to select THL events.

The -first [331] selects only the first stored THL event. For example:

```shell
shell> thl list -first
SEQ# = 0 / FRAG# = 0 (last frag)
- TIME = 2017-06-28 13:12:38.0
- EPOCH# = 0
...
```

- **-first # [331]**

Valid for: `thl list`, `thl purge`

5.2.0. Starting in 5.2.0, the options -first [331], -first # [331], -last [332], and -last # [332], can be used to select THL events.
The `-first #` selects the specified number of events, starting from the first event. For example:

```sh
thl list -first 5
```

Would display the first five events from the stored THL.

- **-last [332]**
  - Valid for: `thl list, thl purge`

5.2.0. Starting in 5.2.0, the options `-first [331]`, `-first # [331]`, `-last [332]`, and `-last # [332]`, can be used to select THL events.

The `-last [332]` selects only the last stored THL event. For example:

```sh
thl list -last
```

When the THL index contains events from 1558-1601, would display events 1597 through to 1601.

### 8.21.2. `thl list` Command

The `list` command to the `thl` command outputs a list of the sequence number information from the THL. By default, the entire THL as stored on disk is output. Command-line options enable you to select individual sequence numbers, sequence number ranges, or all the sequence information from a single file.

```sh
thl list
[-seqno # ]
[-low # ] [-from # ] [-high # ] [-to # ]
[-last # ] [-last ] [-first # ] [-first ]
```

- **-file filename [332]**
  - 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 [332]**
  - 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 [332]**
  - 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 [332]**
  - 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:

<table>
<thead>
<tr>
<th>Sequence No</th>
<th>Epoch</th>
<th>Fragment</th>
<th>Last</th>
<th>Fragment</th>
<th>Date/Time</th>
<th>EventID</th>
<th>SourceID</th>
<th>Comments</th>
</tr>
</thead>
</table>

For more information on the fields displayed, see Section E.1.1, "THL Format".

- **-json**

Only valid with the **-headers** option, the header information is output for the selected sequence numbers from the THL in JSON format. The field contents are identical, with each fragment of each THL sequence being contained in a JSON object, with the output consisting of an array of these sequence objects. For example:

```
[{
    "lastFrag": false,
    "epoch": 7,
    "seqno": 320,
    "time": "2013-05-02 11:41:19.0",
    "frag": 0,
    "comments": "",
    "sourceId": "host1",
    "eventId": "mysql-bin.000004:0000000244490614;0"
},
{
    "lastFrag": true,
    "epoch": 7,
    "seqno": 320,
    "time": "2013-05-02 11:41:19.0",
    "frag": 1,
    "comments": "",
    "sourceId": "host1",
    "eventId": "mysql-bin.000004:0000000244490685;0"
}]
```

For more information on the fields displayed, see THL SEQNO [605].

- **-sizes**

Shows the size information for a given THL event, describing either the size of the SQL, or the number of rows within the given event. For example:

```
shell> thl list -sizes
<table>
<thead>
<tr>
<th>SEQ# Frag Time</th>
<th>Event total:_1 chunks 73 bytes in SQL statements _8 rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 0 06-28 13:21:11.0</td>
<td>Event total: 1 chunks 73 bytes in SQL statements 8 rows</td>
</tr>
<tr>
<td>13 0 06-28 13:21:10.0</td>
<td>Event total: 1645 chunks 0 bytes in SQL statements 1645 rows</td>
</tr>
<tr>
<td>14 0 06-28 13:21:11.0</td>
<td>Event total: 1 chunks 36 bytes in SQL statements 8 rows</td>
</tr>
<tr>
<td>15 0 06-28 13:21:11.0</td>
<td>Event total: 1 chunks 61 bytes in SQL statements 8 rows</td>
</tr>
<tr>
<td>16 0 06-28 13:21:11.0</td>
<td>Event total: 1 chunks 73 bytes in SQL statements 8 rows</td>
</tr>
<tr>
<td>17 0 06-28 13:21:12.0</td>
<td>Event total: 1 chunks 36 bytes in SQL statements 8 rows</td>
</tr>
<tr>
<td>18 0 06-28 13:21:12.0</td>
<td>Event total: 1 chunks 61 bytes in SQL statements 8 rows</td>
</tr>
<tr>
<td>19 0 06-28 13:21:10.0</td>
<td>Event total: 1784 chunks 0 bytes in SQL statements 1784 rows</td>
</tr>
<tr>
<td>20 0 06-28 13:21:12.0</td>
<td>Event total: 1 chunks 73 bytes in SQL statements 8 rows</td>
</tr>
<tr>
<td>21 0 06-28 13:21:11.0</td>
<td>Event total: 1576 chunks 0 bytes in SQL statements 1576 rows</td>
</tr>
<tr>
<td>22 0 06-28 13:21:12.0</td>
<td>Event total: 1 chunks 36 bytes in SQL statements 8 rows</td>
</tr>
<tr>
<td>23 0 06-28 13:21:12.0</td>
<td>Event total: 1 chunks 61 bytes in SQL statements 8 rows</td>
</tr>
</tbody>
</table>
```

Summary information is also output identifying an overall count of the changes. For example:
This information can be useful when viewing or monitoring the replication progress as it can help to indicate and identify the size of a specific transaction, particularly if the transaction is large. This can be particularly useful in combination with the `-first` and/or `-last`.

For more detailed information on individual fragments within a sequence (and for large transactions there will be multiple fragments), use the `thl list -sizesdetail` command.

- `-sizesdetail`

Shows detailed size information for a given THL event, describing either the size of the SQL, or the number of rows within the given event per fragment within each event, and with a summary for each event total. For very large THL event sizes this provide more detailed information about the size and makeup of the event. For example:

```shell
  thl list -sizes -last
```

Summary information is also output identifying an overall count of the changes. For example:

```shell
  thl list -sizessummary
```

This information can be useful when viewing or monitoring the replication progress as it can help to indicate and identify the size of a specific transaction, particularly if the transaction is large. This can be particularly useful in combination with the `-first` and/or `-last`.

- `-sizessummary`

Outputs only the size summary information for the requested THL:

```shell
  thl list -sizessummary
```

- `-specs`

Shows the column specifications, such as identified type, length, and additional settings, when viewing events within row-based replication deployments. This can be helpful when examining THL data in heterogeneous replication deployments.

For example:

```shell
  thl list -low 5282 -specs
```
When identifying the different data types, the following effects should be noted:

- **CHAR** and **VARCHAR** are identified as type 12, **VARCHAR**
- **SET** is identified as an **INTEGER**
- When the value is either **NULL** or **0** (Zero), date and time fields are shown as type 0, **NULL**
- **ENUM** is identified as an **OTHER**
- **BLOB** and **TEXT** are identified as type 2004, **BLOB**
- `-timezone`

Specify the timezone to use when display date or time values. When not specified, times are displayed using UTC.

### 8.21.3. `thl index` Command

The `index` command to `thl` provides a list of all the available THL files and the sequence number range stored within each file:

```
shell> thl index
LogIndexEntry thl.data.0000000001(0:113)
LogIndexEntry thl.data.0000000002(114:278)
LogIndexEntry thl.data.0000000003(279:375)
LogIndexEntry thl.data.0000000004(376:472)
LogIndexEntry thl.data.0000000005(473:569)
LogIndexEntry thl.data.0000000006(570:941)
LogIndexEntry thl.data.0000000007(942:1494)
LogIndexEntry thl.data.0000000008(1495:1658)
LogIndexEntry thl.data.0000000009(1659:1755)
LogIndexEntry thl.data.0000000010(1756:1852)
LogIndexEntry thl.data.0000000011(1853:1949)
LogIndexEntry thl.data.0000000012(1950:2046)
LogIndexEntry thl.data.0000000013(2047:2563)
```

The optional argument `-no-checksum` ignores the checksum information on events in the event that the checksum is corrupt.

### 8.21.4. `thl purge` Command

The `purge` command to the `thl` command deletes sequence number information from the THL files.

```
thl purge [-low # ] [-high # ] [-y ] [-no-checksum ]
```

The `purge` command deletes the THL data according to the following rules:

- **Warning**

  Purging all data requires that the THL information either be recreated from the source table, or reloaded from the master replicator.

  Without any specification, a `purge` command will delete all of the stored THL information.

- When only `-high` is specified, delete all the THL data up to and including the specified sequence number.

- When only `-low` is specified, delete all the THL data from and including the specified sequence number.

- With a range specification, using one or both of the `-low` and `-high` options, the range of sequences will be purged. The rules are the same as for the `list` command, enabling purge from the start to a sequence, from a sequence to the end, or all the sequences within a given range. The ranges must be on the boundary of one or more log files. It is not possible to delete THL data from the middle of a given file.

For example, consider the following list of THL files provided by `thl index`:

```
shell> thl index
LogIndexEntry thl.data.0000000377(5802:5821)
LogIndexEntry thl.data.0000000378(5822:5841)
LogIndexEntry thl.data.0000000379(5842:5861)
LogIndexEntry thl.data.0000000380(5862:5881)
LogIndexEntry thl.data.0000000381(5882:5901)
```

335
The above shows a range of THL sequences from 5802 to 6266.

To delete all of the THL from the start of the list, sequence no 5802, to 6021 (inclusive), use the \texttt{-high} to specify the highest number to be removed (6021):

\begin{verbatim}
shell> thl purge -high 6021
WARNING: The purge command will break replication if you delete all events or delete events that have not reached all slaves.
Are you sure you wish to delete these events \[y/N\]?

y
Deleting events where SEQ# <=6021
2017-02-10 16:31:36,235 [- main] INFO  thl.THLManagerCtrl Transactions deleted
\end{verbatim}

Running a \texttt{thl index}, sequence numbers from 6022 to 6266 are still available:

\begin{verbatim}
shell> thl index
LogIndexEntry thl.data.0000000388(6022:6041)
LogIndexEntry thl.data.0000000389(6042:6061)
LogIndexEntry thl.data.0000000390(6062:6081)
LogIndexEntry thl.data.0000000391(6082:6101)
LogIndexEntry thl.data.0000000392(6102:6121)
LogIndexEntry thl.data.0000000393(6122:6141)
LogIndexEntry thl.data.0000000394(6142:6161)
LogIndexEntry thl.data.0000000395(6162:6181)
LogIndexEntry thl.data.0000000396(6182:6201)
LogIndexEntry thl.data.0000000397(6202:6221)
LogIndexEntry thl.data.0000000398(6222:6241)
LogIndexEntry thl.data.0000000399(6242:6261)
LogIndexEntry thl.data.0000000400(6262:6266)
\end{verbatim}

To delete the last two THL files, specify the sequence number at the start of the file, 6242 to the \texttt{-low} to specify the sequence number:

\begin{verbatim}
shell> thl purge -low 6242 -y
WARNING: The purge command will break replication if you delete all events or delete events that have not reached all slaves.
Deleting events where SEQ# >= 6242
2017-02-10 16:40:42,463 [- main] INFO  thl.THLManagerCtrl Transactions deleted
\end{verbatim}

A \texttt{thl index} shows the sequence as removed:

\begin{verbatim}
shell> thl index
LogIndexEntry thl.data.0000000388(6022:6041)
LogIndexEntry thl.data.0000000389(6042:6061)
LogIndexEntry thl.data.0000000390(6062:6081)
LogIndexEntry thl.data.0000000391(6082:6101)
LogIndexEntry thl.data.0000000392(6102:6121)
LogIndexEntry thl.data.0000000393(6122:6141)
LogIndexEntry thl.data.0000000394(6142:6161)
LogIndexEntry thl.data.0000000395(6162:6181)
LogIndexEntry thl.data.0000000396(6182:6201)
LogIndexEntry thl.data.0000000397(6202:6221)
LogIndexEntry thl.data.0000000398(6222:6241)
LogIndexEntry thl.data.0000000399(6242:6261)
LogIndexEntry thl.data.0000000400(6262:6266)
\end{verbatim}

The confirmation message can be bypassed by using the \texttt{-y} option, which implies that the operation should proceed without further confirmation.

The optional argument \texttt{-no-checksum} ignores the checksum information on events in the event that the checksum is corrupt.

When purging, the THL files must be writable; the replicator must either be offline or stopped when the purge operation is completed.

A \texttt{purge} operation may fail for the following reasons:
Command-line Tools

- Fatal error: The disk log is not writable and cannot be purged.
  The replicator is currently running and not in the offline state. Use `trepctl offline` to release the write lock n the THL files.

- Fatal error: Deletion range invalid; must include one or both log end points: low seqno=0 high seqno=1000
  An invalid sequence number or range was provided. The purge operation will refuse to purge events that do not exist in the THL files and do not match a valid file boundary, i.e. the low figure must match the start of one file and the high the end of a file. Use `thl index` to determine the valid ranges.

8.21.5. thl info Command

The optional argument `-no-checksum` ignores the checksum information on events in the event that the checksum is corrupt.

8.21.6. thl help Command

The `help` command to the `thl` command outputs the current help message text.

8.22. The trepctl Command

The `trepctl` command provides the main status and management interface to Tungsten Replicator. The `trepctl` command is responsible for:

- Putting the replicator online or offline
- Performing backup and restore operations
- Skipping events in the THL in the event of an issue
- Getting status and active configuration information

The operation and control of the command is defined through a series of command-line options which specify general options, replicator wide commands, and service specific commands that provide status and control over specific services.

The `trepctl` command by default operates on the current host and configured service. For installations where there are multiple services and hosts in the deployment. Explicit selection of services and hosts is handled through the use of command-line options, for more information see Section 8.22.1, “trepctl Options”.

```
trepctl
  backup [ -backup agent ] [ -limit s ] [ -storage agent ]
capabilities
clear
clients [-json]
flush [ -limit s ]
heartbeat [ -name ] [ -tz s ] [ -host name ]
kill [ -y ]
load
offline [ -all-services ]
offline-deferred [ -at-event event ] [ -at-heartbeat [ heartbeat ] ] [ -at-seqno seqno ] [ -at-time YYYY-MM-DD.hh:mm:ss ] [ -immediate ]
online [ -all-services ] [ -base-seqno s ] [ -force ] [ -from-event event ] [ -no-checksum ] [ -skip-seqno seqdef ] [ -until-event event ] [ -until-heartbeat [ name ] ] [ -until-seqno seqno ] [ -until-time YYYY-MM-DD.hh:mm:ss ]
perf [ -r ] [ -port number ]
properties [ -filter name ] [ -values ]
purge [ -limit s ] [ -y ]
diff [ -r ]
reset [ -all ] [ -db ] [ -relay ] [ -thl ] [ -y ] [ -retry N ] [ -service name ]
services [ -full ] [ -json ] [ -r ]
servicetable [ -r ]
setrole [ -role master | relay | slave ] [ -uri ]
shard [ -delete shard ] [ -insert shard ] [ -list ] [ -update shard ]
status [ -json ] [ -namechannel-assignments-services-shards-stages-store-tasks-watches ] [ -r ]
unload [ -y ] [ -verbose ]
version
wait [ -applied seqno ] [ -limit s ] [ -state st ]
```

For individual operations, `trepctl` uses a sub-command structure on the command-line that specifies which operation is to be performed. There are two classifications of commands, global commands, which operate across all replicator services, and service-specific commands that perform operations on a specific service and/or host. For information on the global commands available, see Section 8.22.2, “trepctl Global Commands”. Information on individual commands can be found in Section 8.22.3, “trepctl Service Commands”.

337
8.22.1. trepctl Options

Table 8.38. trepctl Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-host name</td>
<td>Host name of the replicator</td>
</tr>
<tr>
<td>-port number</td>
<td>Port number of the replicator</td>
</tr>
<tr>
<td>-retry N</td>
<td>Number of times to retry the connection</td>
</tr>
<tr>
<td>-service name</td>
<td>Name of the replicator service</td>
</tr>
<tr>
<td>-verbose</td>
<td>Enable verbose messages for operations</td>
</tr>
</tbody>
</table>

Global command-line options enable you to select specific hosts and services. If available, trepctl will read the active configuration to determining the host, service, and port information. If this is unavailable or inaccessible, the following rules are used to determine which host or service to operate upon:

- If no host is specified, then trepctl defaults to the host on which the command is being executed.
- If no service is specified:
  - If only one service has been configured, then trepctl defaults to showing information for the configured service.
  - If multiple services are configured, then trepctl returns an error, and requests a specific service be selected.

To use the global options:

- **-host**
  Specify the host for the operation. The replicator service must be running on the remote host for this operation to work.

- **-port**
  Specify the base TCP/IP port used for administration. The default is port 10000; port 10001 is also used. When using different ports, port and port+1 is used, i.e. if port 4996 is specified, then port 4997 will be used as well. When multiple replicators are installed on the same host, different numbers may be used.

- **-service**
  The servicename to be used for the requested status or control operation. When multiple services have been configured, the servicename must be specified.

    ```
    shell> trepctl status
    Processing status command...
    Operation failed: You must specify a service name with the -service flag
    ```

  Starting in 6.0.4, if multiple services are configured but not specified, then a list of available services is provided:

    ```
    shell> trepctl status
    Processing status command...
    Operation failed: You must specify a service name with the -service flag because there is more than one service available. The currently available status commands are:
    trepctl -service north status
    trepctl -service north_from_east status
    trepctl -service north_from_west status
    ```

- **-verbose**
  Turns on verbose reporting of the individual operations. This includes connectivity to the replicator service and individual operation steps. This can be useful when diagnosing an issue and identifying the location of a particular problem, such as timeouts when access a remote replicator.

- **-retry**
  Retry the request operation the specified number of times. The default is 10.

8.22.2. trepctl Global Commands

The trepctl command supports a number of commands that are global, or which work across the replicator regardless of the configuration or selection of individual services.
### Table 8.39. 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>servicetable</td>
<td>List all the currently configured services in a tabular format</td>
</tr>
<tr>
<td>version</td>
<td>Show the replicator version number and build</td>
</tr>
</tbody>
</table>

These commands can be executed on the current or a specified host. Because these commands operate for replicators irrespective of the service configuration, selecting or specifying a service is not required.

#### 8.22.2.1. trepctl kill Command

The `trepctl kill` command terminates the replicator without performing any cleanup of the replicator service, THL or sequence number information stored in the database. Using this option may cause problems when the replicator service is restarted.

```
trepctl kill [-y]
```

When executed, `trepctl` will ask for confirmation:

```
shell> trepctl kill
Do you really want to kill the replicator process? [yes/NO]
```

The default is no. To kill the service, ignoring the interactive check, use the `-y` option:

```
shell> trepctl kill -y
Sending kill command to replicator
Replicator appears to be stopped
```

#### 8.22.2.2. 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
```

For more information on the fields displayed, see Section E.2, “Generated Field Reference”.

For a replicator with multiple services, the information is output for each configured service:

```
shell> trepctl services
Processing services command...
NAME    VALUE
appliedLastSeqno: 44
appliedLatency  : 0.692
role      : master
serviceName : alpha
serviceType : local
started   : true
state     : ONLINE
NAME    VALUE
appliedLastSeqno: 49
appliedLatency  : 0.57
role      : slave
serviceName : beta
serviceType : remote
started   : true
state     : ONLINE
```
The information can be reported in JSON format by using the `-json` option to the command:

```
shell> trepctl services -json
```

```
{
    "serviceType" : "local",
    "appliedLatency" : "0.48",
    "serviceName" : "alpha",
    "appliedLastSeqno" : "2541",
    "started" : "true",
    "role" : "master",
    "state" : "ONLINE"
}
```

The information is output as an array of objects, one object for each service identified.

If the `-full` option is added, the JSON output includes full details of the service, similar to that output by the `trepctl status` command, but for each configured service:

```
shell> trepctl services -json -full
```

```
{
    "masterConnectUri" : ",",
    "rmiPort" : "18880",
    "clusterName" : "default",
    "currentTimeMillis" : "1370256230198",
    "state" : "ONLINE",
    "maximumStoredSeqNo" : "2541",
    "minimumStoredSeqNo" : "8",
    "pendingErrorCode" : "NONE",
    "masterListenUri" : "thl://host1:2112/",
    "pendingErrorSeqno" : "-1",
    "pipelineSource" : "jdbc:mysql:thin://host1:3306/",
    "serviceName" : "alpha",
    "pendingErrorEventId" : "NONE",
    "appliedLatency" : "0.48",
    "transitioningTo" : "",
    "relativeLatency" : "245804.198",
    "role" : "master",
    "siteName" : "default",
    "pendingError" : "NONE",
    "uptimeSeconds" : "246023.627",
    "latestEpochNumber" : "2537",
    "extensions" : "",
    "dataServerHost" : "host1",
    "resourcePrecedence" : "99",
    "pendingExceptionMessage" : "NONE",
    "simpleServiceName" : "alpha",
    "sourceId" : "host1",
    "offlineRequests" : "NONE",
    "channels" : "1",
    "version" : "Tungsten Replicator 6.1.3 build 41",
    "seqnoType" : "java.lang.Long",
    "serviceType" : "local",
    "currentEventId" : "mysql-bin.48886:000000000000003:0",
    "appliedLastEventId" : "mysql-bin.48886:000000000000003:0",
    "timeInStateSeconds" : "245803.753",
    "appliedLastSeqno" : "2541",
    "started" : "true"
}
```

Auto-refresh support added in 6.0.1. Starting with Tungsten Clustering 6.0.1, `trepctl services` supports the `-r` option to support auto-refresh.

For more information on the fields displayed, see Section E.2, "Generated Field Reference".

### 8.22.2.3. trepctl servicetable Command

The `trepctl servicetable` command outputs a list of all the current services and current status information in a tabular format to make it easy to determine multi-service installations.
trepctl servicetable

For example:

```
shell> trepctl servicetable
Processing servicetable command...
Service | Status | Role       | MasterConnectUri               | SeqNo | Latency
--------|--------|------------|-------------------------------|-------|--------
alpha   | ONLINE | slave      | thl://trfiltera:2112/         | 322   | 0.00
beta    | ONLINE | slave      | thl://ubuntuheterosrc:2112/   | 12    | 4658.59
Finished servicetable command...
```

The command also supports the auto-refresh option, `-r`:

```
shell> trepctl servicetable -r 5
```

For more information on the fields displayed, see Section E.2, “Generated Field Reference”.

### 8.22.2.4. trepctl version Command

The `trepctl version` command outputs the version number of the specified replicator service.

```
trepctl version
```

```
Tungsten Replicator 6.1.3 build 43
```

The system can also be used to obtain remote version:

```
shell> trepctl -host host2 version
Tungsten Replicator 6.1.3 build 43
```

Version numbers consist of two parts, the main version number which denotes the product release, and the build number. Updates and fixes to a version may use updated build numbers as part of the same product release.

### 8.22.3. trepctl Service Commands

The `trepctl` service commands operate per-service, that is, when there are multiple services in a configuration, the service name on which the command operates must be explicitly stated. For example, when a backup is executed, the backup executes on an explicit, specified service.

The individuality of different services is critical when dealing with the replicator commands. Services can be placed into online or offline states independently of each other, since each service will be replicating information between different hosts and environments.

#### Table 8.40. trepctl Service Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup</td>
<td>Backup database</td>
</tr>
<tr>
<td>capabilities</td>
<td>List the configured replicator capabilities</td>
</tr>
<tr>
<td>check</td>
<td>Generate consistency check</td>
</tr>
<tr>
<td>clear</td>
<td>Clear one or all dynamic variables</td>
</tr>
<tr>
<td>clients</td>
<td>List clients connected to this replicator</td>
</tr>
<tr>
<td>flush</td>
<td>Synchronize transaction history log to database</td>
</tr>
<tr>
<td>heartbeat</td>
<td>Insert a heartbeat event with optional name</td>
</tr>
<tr>
<td>load</td>
<td>Load the replication service</td>
</tr>
<tr>
<td>offline</td>
<td>Set replicator to OFFLINE state</td>
</tr>
<tr>
<td>offline-deferred</td>
<td>Set replicator OFFLINE at a future point in the replication stream</td>
</tr>
<tr>
<td>online</td>
<td>Set Replicator to ONLINE with start and stop points</td>
</tr>
<tr>
<td>perf</td>
<td>Print detailed performance information</td>
</tr>
<tr>
<td>properties</td>
<td>Display a list of all internal properties</td>
</tr>
<tr>
<td>purge</td>
<td>Purge non-Tungsten logins on database</td>
</tr>
<tr>
<td>qs</td>
<td>Print a simplified quick replicator status</td>
</tr>
<tr>
<td>reset</td>
<td>Deletes the replicator service</td>
</tr>
</tbody>
</table>
Command-line Tools

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setrole</td>
<td>Set replicator role</td>
</tr>
<tr>
<td>shard</td>
<td>List, add, update, and delete shards</td>
</tr>
<tr>
<td>status</td>
<td>Print replicator status information</td>
</tr>
<tr>
<td>unload</td>
<td>Unload the replication service</td>
</tr>
<tr>
<td>wait</td>
<td>Wait for the replicator to reach a specific state, time or applied sequence number</td>
</tr>
</tbody>
</table>

The following sections detail each command individually, with specific options, operations and information.

### 8.22.3.1. trepctl backup Command

The `trepctl backup` command performs a backup of the corresponding database for the selected service.

```bash
trepctl backup [-backup agent] [-limit s] [-storage agent]```

Where:

**Table 8.41. 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:

```bash
shell> trepctl backup
Backup completed successfully; URI=storage://file-system/store-0000000002.properties
```

The return information gives the URI of the backup properties file. This information can be used when performing a restore operation as the source of the backup. See Section 8.22.3.17, "trepctl restore Command". Different backup solutions may require that the replicator be placed into the `OFFLINE` state before the backup is performed.

A log of the backup operation will be stored in the replicator log directory, if a file corresponding to the backup tool used (e.g. `mysqldump.log`).

If multiple backup agents have been configured, the backup agent can be selected on the command-line:

```bash
shell> trepctl backup -backup mysqldump
```

If multiple storage agents have been configured, the storage agent can be selected using the `-storage [342]` option:

```bash
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 [342]` 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 [342]` option specifies how long `trepctl` should wait before returning.

For example, to wait five seconds before returning:

```bash
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:

```bash
... 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.

```bash
... backup.BackupTask Storing backup result...
... backup.FileSystemStorageAgent Allocated backup location: »
... backup.FileSystemStorageAgent Stored backup storage file: »
... backup.FileSystemStorageAgent Stored backup storage properties: »
```
The URI can be used during a restore.

### 8.22.3.2. `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:

```
Roles:             [master, slave]
Replication Model: push
Consistency Check: true
Heartbeat:         true
Flush:             true
```

The fields output are as follows:

- **Roles**
  Indicates whether the replicator can be a **master** or **slave**, or both.

- **Replication Model**
  The model used by the replication system. The default model for MySQL for example is push, where information is extracted from the binary log and pushed to slaves that apply the transactions. The pull model is used for heterogeneous deployments.

- **Consistency Check**
  Indicates whether the internal consistency check is supported. For more information see Section 8.22.3.3, "trepctl check Command".

- **Heartbeat**
  Indicates whether the heartbeat service is supported. For more information see Section 8.22.3.7, "trepctl heartbeat Command".

- **Flush**
  Indicates whether the `trepctl flush` operation is supported.

### 8.22.3.3. `trepctl check` Command

The `check` command operates by running a CRC check on the schema or table specified, creating a temporary table containing the check data and values during the process. The data collected during this process is then written to a consistency table within the replication configuration schema and is used to verify the table data consistency on the master and the slave.

**Warning**

Because the check operation is creating a temporary table containing a CRC of each row within the specified schema or specific table, the size of the temporary table created can be quite large as it consists of CRC and row count information for each row of each table (within the specified row limits). The configured directory used by MySQL for temporary table creation will need a suitable amount of space to hold the temporary data.

### 8.22.3.4. `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:
8.22.3.5. trepctl clients Command

Outputs a list of the clients that have been connected to the master service since it went online. If a slave service goes offline or is stopped, it will still be reported by this command.

`trepctl clients [-json ]`

Where:

Table 8.42. trepctl clients Command Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-json</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"
  },
  {
    "rmiPort": "10000",
    "rmiHost": "host2"
  },
  {
    "rmiPort": "10000",
    "rmiHost": "host3"
  }
]
```

The information is divided first by host, and then by the RMI management port.

8.22.3.6. 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
Master log is synchronized with database at log sequence number: 3622
```

The flush operation is always initiated, and by default `trepctl` will wait until the operation completes. Using the `-limit` option, the amount of time the command-line waits before returning can be specified:

```
shell> trepctl flush -limit 1
```

8.22.3.7. trepctl heartbeat Command

Inserts a heartbeat into the replication stream, which can be used to identify replication points.

`trepctl heartbeat [-name ] [-tz s ]`

The heartbeat system is a way of inserting an identifiable event into the THL that is independent of the data being replicated. This can be useful when performing different operations on the data where specific checkpoints must be identified.

To insert a standard heartbeat:
**Command-line Tools**

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
   appliedLastEventId        : mysql-bin.000009:0000000000009310;0
   appliedLastSeqno          : 3631
   ```

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:
   ```
   Processing status command....
   NAME                      VALUE
   appliedLastEventId        : mysql-bin.000009:0000000000008271;0
   appliedLastSeqno          : 3627
   appliedLatency            : 8.704
   ...
   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.

8.22.3.7.1. `trepctl heartbeat` Time Zone Handling

When the Replicator inserts a heartbeat there is an associated timezone. The default uses the Replicator host's timezone.

The `-tz` option to the `trepctl heartbeat` command may be used to force the use of a specific timezone.

For example, use GMT as the timezone when inserting a heartbeat:

```bash
shell> trepctl heartbeat -tz NONE
```

Use the Replicator host's timezone to insert the heartbeat:

```bash
shell> trepctl heartbeat -tz HOST
```

Use the given timezone to insert the heartbeat:

```bash
shell> trepctl heartbeat -tz (valid timezone id)
```

If the MySQL server timezone is different from the host timezone (which is strongly not recommended), then `-tz (valid timezone id)` should be used instead where `(valid timezone id)` should be the same as the MySQL server timezone.

8.22.3.7.2. `trepctl heartbeat` Internal Implementation

Internally, the heartbeat system operates through a tag added to the metadata of the THL entry and through a dedicated heartbeat table within the schema created for the replicator service. The table contains the sequence number, event ID, timestamp and heartbeat name. The heartbeat information is written into a special record within the transaction history log. A sample THL entry can be seen in the output below:

```
SEQ# = 3629 / FRAG# = 0 (last frag)
  - TIME = 2013-07-19 12:14:57.0
  - EPOCH# = 3614
  - EVENTID = mysql-bin.000009:0000000000008681:0
  - SOURCEID = host1
  - METADATA = [mysql_server_id=1687011;dbms_type=mysql;is_metadata=true;service=alpha;
    shard=tungsten_alpha;heartbeat=dataload]
  - TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
  - OPTIONS = ['##charset = UTF-8, autocommit = 1, sql_auto_is_null = 0,
    foreign_key_checks = 1, unique_checks = 1, sql_mode = 'IGNORE_SPACE',
    character_set_client = 33, collation_connection = 33, collation_server = 8']
  - SCHEMA = tungsten_alpha
  - SQL(0) = UPDATE tungsten_alpha.heartbeat SET source_tstamp= '2013-07-19 12:14:57',
    salt= 9, name= 'dataload'  WHERE id= 1
```

During replication, slaves identify the heartbeat and record this information into their own heartbeat table. Because the heartbeat is recorded into the transaction history log, the specific sequence number of the transaction, and the event itself can be easily identified.

8.22.3.8. `trepctl load` Command

Load the replicator service.

```bash
trepctl load
```

Load the replicator service. The service name must be specified on the command-line, even when only one service is configured:

```bash
shell> trepctl load
Operation failed: You must specify a service name using -service
```

The service name can be specified using the `-service` option:

```bash
shell> trepctl -service alpha load
Service loaded successfully: name=alpha
```

8.22.3.9. `trepctl offline` Command

The `trepctl offline` command puts the replicator into the offline state, stopping replication.

```bash
trepctl offline [-all-services] [+ -immediate]
```

To put the replicator offline:

```bash
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 \texttt{-immediate} option:

\begin{verbatim}
shell> trepctl offline -immediate
\end{verbatim}

\section*{8.22.3.10. \texttt{trepctl offline-deferred} Command}

The \texttt{trepctl offline-deferred} sets a future sequence, event or heartbeat as the trigger to put the replicator in the offline state.

\begin{verbatim}
trepctl offline-deferred [-at-event event] [-at-heartbeat [heartbeat]] [-at-seqno seqno] [-at-time YYYY-MM-DD_hh:mm:ss]
\end{verbatim}

Where:

\begin{table}[h]
\centering
\begin{tabular}{|c|l|}
\hline
Option & Description \\
\hline
\texttt{-at-event event} & Go offline at the specified event \\
\texttt{-at-heartbeat [heartbeat]} & Go offline when the specified heartbeat is identified \\
\texttt{-at-seqno seqno} & Go offline at the specified sequence number \\
\texttt{-at-time YYYY-MM-DD_hh:mm:ss} & Go offline at the specified time \\
\hline
\end{tabular}
\caption{\texttt{trepctl offline-deferred} Command Options}
\end{table}

The \texttt{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 \texttt{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 \texttt{trepctl offline}. See Section 8.22.3.9, "\texttt{trepctl offline} Command".

The supported triggers are:

• \texttt{-at-seqno} [347]
  Specifies a transaction sequence number (GTID) where the replication will be stopped. For example:

  \begin{verbatim}
  shell> trepctl offline-deferred -at-seqno 3800
  \end{verbatim}

  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.

• \texttt{-at-event} [347]
  Specifies the event where replication should stop:

  \begin{verbatim}
  shell> trepctl offline-deferred -at-event 'mysql-bin.000009:0000000000088140;3'
  \end{verbatim}

  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.

• \texttt{-at-heartbeat} [347]
  Specifies the name of a specific heartbeat to look for when replication should be stopped.

• \texttt{-at-time} [347]
  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).

  \begin{verbatim}
  shell> trepctl offline-deferred -at-time 2013-09-01_00:00:00
  \end{verbatim}

  The transaction being executed at the time specified completes, then the replicator goes offline.
If any specified deferred point has already been reached, then the replicator will go offline anyway. For example, if the current sequence number is 3800 and the deferred sequence number specified is 3700, then the replicator will go offline immediately just as if the `trepctl offline` command has been used.

When a trigger is reached, for example if a sequence number is given, that sequence will be applied and then the replicator will go offline.

The status of the pending `trepctl offline-deferred` setting can be identified within the status output within the `offlineRequests` field:

```
shell> trepctl status
...
offlineRequests : Offline at sequence number: 3810
```

Multiple `trepctl offline-deferred` commands can be given for each corresponding trigger type. For example, below three different triggers have been specified, sequence number, time and heartbeat event, with the status showing each deferred event separated by a semicolon:

```
shell> trepctl status
...
offlineRequests : Offline at heartbeat event: dataloaded;Offline at »
sequence number: 3640;Offline at time: 2013-09-01 00:00:00 EDT
```

Offline deferred settings are cleared when the replicator is put into the offline state, either manually or automatically.

### 8.22.3.11. trepctl online Command

The `trepctl online` command puts the replicator into the online state. During the state change from offline to online various options can be used to control how the replicator goes back on line. For example, the replicator can be placed online, skipping one or more faulty transactions or disabling specific configurations.

```
```

Where:

**Table 8.44. trepctl online Command Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-all-services</td>
<td>Place online all available services</td>
</tr>
<tr>
<td>-base-seqno x</td>
<td>On a master, restart replication using the specified sequence number</td>
</tr>
<tr>
<td>-force</td>
<td>Force the online state</td>
</tr>
<tr>
<td>-from-event event</td>
<td>Start replication from the specified event</td>
</tr>
<tr>
<td>-no-checksum</td>
<td>Disable checksums for all events when going online</td>
</tr>
<tr>
<td>-skip-seqno seqdef</td>
<td>Skip one, multiple, or ranges of sequence numbers before going online</td>
</tr>
<tr>
<td>-until-event event [name]</td>
<td>Define an event when replication will stop</td>
</tr>
<tr>
<td>-until-heartbeat [name]</td>
<td>Define a heartbeat when replication will stop</td>
</tr>
<tr>
<td>-until-seqno seqno</td>
<td>Define a sequence no when replication will stop</td>
</tr>
<tr>
<td>-until-time YYYY-MM-DD_hh:mm:ss</td>
<td>Define a time when replication will stop</td>
</tr>
</tbody>
</table>

The `trepctl online` command attempts to switch replicator into the online state. The replicator may need to be put online because it has been placed offline for maintenance, or due to a failure.

To put the replicator online use the standard form of the command:

```
shell> trepctl online
```

Going online may fail if the reason for going offline was due to a fault in processing the THL, or in applying changes to the dataserver. The replicator will refuse to go online if there is a fault, but certain failures can be explicitly bypassed.

### 8.22.3.11.1. 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=5230 fragno=0 »
message=java.sql.SQLException: Statement failed on slave but succeeded on master
```

848
To skip the single sequence number, 5250, shown:

```shell
$ trepctl online -skip-seqno 5250
```

The sequence number specification can be specified according to the following rules:

- A single sequence number:
  ```shell
  $ trepctl online -skip-seqno 5250
  ```
- A sequence range:
  ```shell
  $ trepctl online -skip-seqno 5250-5260
  ```
- A comma-separated list of individual sequence numbers and/or ranges:
  ```shell
  $ trepctl online -skip-seqno 5250,5251,5253-5260
  ```

### 8.22.3.11.2. Going Online from a Base Sequence Number

**Note**

To set the position of the replicator, the `dsctl` command can also be used.

Alternatively, the base sequence number, the transaction ID where replication should start, can be specified explicitly:

```shell
$ trepctl online -base-seqno 5260
```

**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.

**Note**

If issuing `-base-seqno` and `-from-event` together, you need to also issue the `-force` option for it to work, otherwise a warning will be displayed.

### 8.22.3.11.3. Going Online from a Specific Event

**Note**

To set the position of the replicator, the `dsctl` command can also be used.

If the source event (for example, the MySQL binlog position) is known, this can be used as the reference point when going online and restarting replication:

```shell
$ trepctl online -from-event 'mysql-bin.000011:0000000000002552;0'
```

When used, replication will start from the next event within the THL. The event ID provided must be valid. The event cannot be found in the THL, the operation will fail.

**Note**

If issuing `-base-seqno` and `-from-event` together, you need to also issue the `-force` option for it to work, otherwise a warning will be displayed.

### 8.22.3.11.4. 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
offlineRequests : 0
...
8.22.3.11.4.1. Going Online Until Specified Sequence Number

To go online until a specific transaction ID, use `-until-seqno`:

```shell
shell> trepctl online -until-seqno 5260
```

This will process all transactions up to, and including, sequence 5260, at which point the replicator will go offline.

8.22.3.11.4.2. Going Online Until Specified Event

To go online until a specific event ID:

```shell
shell> trepctl online -until-event 'mysql-bin.000011:0000000000003057:0'
```

Replication will go offline when the event ID up to the specified event has been processed.

8.22.3.11.4.3. Going Online Until Heartbeat

To go online until a heartbeat event:

```shell
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
shell> trepctl online -until-heartbeat load-finished
```

8.22.3.11.4.4. Going Online Until Specified Time

To go online until a specific date and time:

```shell
shell> trepctl online -until-time 2014-01-01_00:00:00
```

Replication will go offline once the transaction being processed at the time specified has completed.

8.22.3.11.5. 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
shell> trepctl online -force
```

8.22.3.11.6. 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
shell> trepctl online -no-checksum
```

This will bring the replicator online without reading or writing checksum information.

**Important**

Use of the `-no-checksum` option disables both the reading and writing of checksums on log records. If starting the replicator without checksums to get past a checksum failure, the replicator should be taken offline again once the offending event has been replicated. This will avoid generating too many local records in the THL without checksums.

8.22.3.12. trepctl perf Command

Display a list of all the internal properties. The list can be filtered.

```bash
trepctl perf [ -r ]
```

The `perf` outputs performance information on a stage by stage basis from the current replicator. The information has been reformatted and extracted from the existing replicator status, task and stage information available through other commands and requests, but reformatted and with values calculated to make identifying specific performance metrics quicker.

For example, on a typical extraction replicator:
The individual statistics shown are as follows:

- All statistics within the replicator are reset when the replicator goes ONLINE. The statistics shown are therefore displayed relative to the current uptime for the replicator.

- For each stage, the following information is shown:
  - Stage name
  - Seqno — this is the current SEQNO[605] number for the specified stage. A difference in sequence numbers is possible (as seen in the applier example above) during startup or synchronisation.
  - Latency — the latency of this stage compared to the commit time of the original transaction.
  - Events — the number of THL events processed by this stage.

Statistics are then shown for each stage, two rows, first for the time to process all of the specified events, and then an average processing time for the events processed during that time within that stage. The individual statistics shown are as follows:

- Extraction — the time taken to extract the event from the current source. On an extractor, this is the source database (for example, the binary log in MySQL). On other stages this is the time to read from disk or the remote replicator the THL event.

- Filtering — the time taken to process the events through the filters configured in the specified stage.

- Applying — the time taken to apply the event to the end of the stage, whether that is to THL on disk, the next queue in preparation for the next stage, or the target database.

- Other — the time taken for other parts of the stage process, this includes waiting for thread management, updating internal structures, and recording information in the target datasource system, such as trep_commit_seqno.

- Filters in stage — The list of filters configured for this stage in the order in which they are applied to the event.

For convenience, the performance display can be set to refresh with a configured interval using the trepctl perf -r 5 command.

In the event that the replicator is currently offline, no statistics are displayed:

```
shell> trepctl perf
Currently not online; performance stats not available
State: Safely Offline for 6.491s
```

### 8.22.3.13. trepctl properties Command

Display a list of all the internal properties. The list can be filtered.

```
trepctl properties [-filter name] [-values]
```

The list of properties can be used to determine the current configuration:

```
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",
    "jdbc:mysql:thin://host1:3306/tungsten_alpha?createDB=true",
}```
Command-line Tools

...}

Note

Passwords are not displayed in the output.

The information is output as a JSON object with key/value pairs for each property and corresponding value.

The list can be filtered using the -filter option:

```
shell> trepctl properties -filter shard
{
  "replicator.filter.shardfilter": »
   com.continuent.tungsten.replicator.shard.ShardFilter",
  "replicator.filter.shardbyseqno": »
   com.continuent.tungsten.replicator.filter.JavaScriptFilter",
  "replicator.filter.shardbyseqno.shards": '1000',
  "replicator.filter.shardfilter.enforceHome": "false",
  "replicator.filter.shardfilter.unknownShardPolicy": "error",
  "replicator.filter.shardbyseqno.script": »
   ../../tungsten-replicator//samples/extensions/javascript/shardbyseqno.js",
  "replicator.filter.shardbytable.script": »
   ../../tungsten-replicator//samples/extensions/javascript/shardbytable.js",
  "replicator.filter.shardfilter.enabled": 'true',
  "replicator.filter.shardfilter.allowWhitelisted": "false",
  "replicator.shard.default.db": "stringent",
  "replicator.filter.shardbytable": »
   com.continuent.tungsten.replicator.filter.JavaScriptFilter",
  "replicator.filter.shardfilter.autoCreate": "false",
  "replicator.filter.shardfilter.unwantedShardPolicy": "error"
}
```

The value or values from filtered properties can be retrieved by using the -values option:

```
shell> trepctl properties -filter site.name -values
default
```

If a filter that would select multiple values is specified, all the values are listed without field names:

```
shell> trepctl properties -filter shard -values
com.continuent.tungsten.replicator.shard.ShardFilter
com.continuent.tungsten.replicator.filter.JavaScriptFilter
1800
false
../../tungsten-replicator//samples/extensions/javascript/shardbyseqno.js
error
../../tungsten-replicator//samples/extensions/javascript/shardbytable.js
true
false
stringent
com.continuent.tungsten.replicator.filter.JavaScriptFilter
false
error
```

8.22.3.14. trepctl purge Command

Forces all logins on the attached database, other than those directly related to Tungsten Clustering, to be disconnected. The command is only supported on master, and can be used to disconnect users before a switchover or taking a master offline to prevent further use of the system.

```
trepctl purge [-limit s] [-y]
```

Where:

| Table 8.45. trepctl purge Command Options
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>:-------------:</td>
<td>:-------------:</td>
</tr>
<tr>
<td>-limit s [353]</td>
<td>Specify the waiting time for the operation</td>
</tr>
<tr>
<td>-y [353]</td>
<td>Indicates that the command should continue without interactive confirmation</td>
</tr>
</tbody>
</table>

Warning

Use of the command will disconnect running users and queries and may leave the database in an unknown state. It should be used with care, and only when the dangers and potential results are understood.
To close the connections:

```
shell> trepctl purge
Do you really want to purge non-Tungsten DBMS sessions? [yes/NO]
```

You will be prompted to confirm the operation. To skip this confirmation and purge connections, use the `-y` option:

```
shell> trepctl purge -y
Directing replicator to purge non-Tungsten sessions
Number of sessions purged: 0
```

An optional parameter, `-wait` defines the period of time that the operation will wait before returning to the command-line.

An optional parameter, `-limit` defines the period of time that the operation will wait before returning to the command-line.

### 8.22.3.15. `trepctl qs` Command

The `trepctl qs` command provides a quicker, simpler, status display for the replicator showing only the critical information in a human-readable form. For example:

```
shell> trepctl qs
State: alpha Online for 4.21s, running for 1781.766s
Latency: 18.0s from source DB commit time on thl://ubuntuheterosrc.mcb:2112/ into target database
1216.315s since last source commit
Sequence: 4804 last applied, 0 transactions behind (0-4804 stored) estimate 0.00s before synchronization
```

The information presented is as follows:

- **State: alpha Online for 4.21s, running for 1781.766s**
  
  The top line shows the basic status information about the replicator:
  
  - The name of the service (`alpha`).
  - The replicator’s current state (`online`) and the time in that state.
  - The amount of time the replicator has been running.
  
  - **Latency: 18.0s from source DB commit time on thl://ubuntuheterosrc.mcb:2112/ into target database**
    
    The second line shows the latency information, the information shown is based on the role of the replicator. The above line is shown on an applier, where the latency information shows the write delay into the target database, where the information is coming from, and applying to the target database. For a master (extractor) the information shown describes the latency from extraction into the THL files:
    
    ```
    State: alpha Online for 169999.144s, running for 169999.138s
    Latency: 0.113s from DB commit time on ubuntuheterosrc into THL
    1679.354s since last database commit
    Sequence: 4859 last applied, 0 transactions behind (0-4859 stored) estimate 0.00s before synchronization
    ```
    
    - 1216.315s since last source commit
      
      The next line shows the interval since the last time there was a database commit. On a master (extractor) is the time between the last database commit to the binary log and the information being written to THL. On a slave, it’s the time between the last database commit on the source database and when the transaction was written to the target.
      
    - **Sequence: 4804 last applied, 0 transactions behind (0-4804 stored) estimate 0.00s before synchronization**
      
      The last line shows the sequence information:
      
      - The last applied sequence number (to THL on a master, or to the target database on a slave).
      - The number of transactions behind the current stored transaction list. This is an indication on a slave of how far behind in transactions (not latency) the slave is from the master.
      - The range of transactions currently stored (from minimum to maximum stored sequence number).
      - An estimate of how long it will take to apply the outstanding transactions. The calculation is made by determining the average rate transactions are being applied (either extraction or applying) against the number of outstanding transactions. It assumes all outstanding transactions are of an equal size. The actual THL transaction size is not taken into account. For information on THL sizes, try the `thl list -sizes` command.

If the replicator is offline due to being deliberately placed offline using `trepctl offline` then the basic information and status is shown:

```
shell> trepctl qs
State: Safely Offline for 352.773s
```
In the event of a replicator failure of some kind this will be reported in the output:

```
State: alpha Faulty (Offline) for 2.613s
Error Reason: SEQNO 4859 did not apply
Error: CSV loading failed: schema=test table=msg CSV file=/opt/continuent/tmp/staging/alpha/staging0/test-msg-4859.csv
» message=Wrapped com.continuent.tungsten.replicator.ReplicatorException: OS command failed: command=cqlsh --keyspace=test
   » --execute="copy stage_xxx_msg (tungsten_opcode,tungsten_seqno,tungsten_row_id,tungsten_commit_timestamp,id,msg) from
   » '/opt/continuent/tmp/staging/alpha/staging0/test-msg-4859.csv' with NULL='NULL';" rc=1 stdout= stderr=Connection
   » error: ('Unable to complete the operation against any hosts', {}) (/opt/continuent/tungsten/tungsten-replicator/appliers/batch/cassandra.js#81)
```

8.22.3.16. trepctl reset Command

The `trepctl reset` command resets an existing replicator service, performing the following operations:

- Deleting the local THL and relay directories
- Removes the Tungsten schema from the dataserver
- Removes any dynamic properties that have previously been set

The service name must be specified, using `-service`.

```
trepctl reset [-all] [-db] [-relay] [-thl] [-y]
```

Where:

**Table 8.46. trepctl reset Command Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-all [354]</td>
<td>Deletes the thl directory, relay logs directory and tungsten database for the service. Same as specifying -thl -relay -db</td>
</tr>
<tr>
<td>-db [354]</td>
<td>Deletes the tungsten_{service_name} database for the service</td>
</tr>
<tr>
<td>-relay [354]</td>
<td>Deletes the relay directory for the service</td>
</tr>
<tr>
<td>-thl [354]</td>
<td>Deletes the thl directory for the service</td>
</tr>
<tr>
<td>-y [354]</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 [354]` option:

```
shell> trepctl -service alpha reset -y
```

Then put the replicator back online again:

```
shell> trepctl online
```

You can also reset only part of the overall service by including one of the following options:

- Reset all components of the service.
- Reset the THL. This is equivalent to running `thl purge`.
- Reset the relay log contents.
- Reset the database, including emptying the `trep_commit_seqno` and other control tables.
- Reset the redo log contents of the service. Valid only for Oracle extraction deployments

8.22.3.17. 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`:

```
shell> trepctl online
```

Any outstanding events from the master will be processed and applied to the slave, which will catch up to the current master status over time.

### 8.22.3.18. `trepctl setrole` Command

The `trepctl setrole` command changes the role of the replicator service. This command can be used to change a configured host between slave and master roles, for example during switchover.

```
trepctl setrole [-role master|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 [355]</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/
```

### 8.22.3.19. `trepctl shard` Command

The `trepctl shard` command provides an interface to the replicator shard system definition system.

```
trepctl shard [-delete shard] [-insert shard] [-list] [-update shard]
```

Where:

<table>
<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 3.3, “Deploying Multisite/Multimaster Clustering” Section 3.2, “Deploying Composite Master/Slave Clustering”.

#### 8.22.3.19.1. Listing Current Shards

To obtain a list of the currently configured shards:

```
shell> trepctl shard -list
shard_id master critical
alpha sales true
```

The shard map information can also be captured and then edited to update existing configurations:

```
shell> trepctl shard -list>shard.map
```

#### 8.22.3.19.2. Inserting a New Shard Configuration

To add a new shard map definition, either enter the information interactively:

```
shell> trepctl shard -insert
Reading from standard input
```
1 new shard inserted

Or import from a file:

```
shell> trepctl shard -insert < shard.map
Reading from standard input
1 new shard inserted
```

### 8.22.3.19.3. Updating an Existing Shard Configuration

To update a definition:

```
shell> trepctl shard -update < shard.map
Reading from standard input
1 shard updated
```

### 8.22.3.19.4. Deleting a Shard Configuration

To delete a single shard definition, specify the shard name:

```
shell> trepctl shard -delete alpha
```

### 8.22.3.20. trepctl status Command

The `trepctl status` command provides status information about the selected data service. The status information by default is a generic status report containing the key fields of status information. More detailed service information can be obtained by specifying the status name with the `--name` parameter.

The format of the command is:

```
trepctl status 
  [-json] 
  [-name channel-assignments services shards stages stores tasks watches] 
  [-r]
```

Where:

<table>
<thead>
<tr>
<th>Table 8.49. <code>trepctl status</code> Command Options</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Option</strong></td>
</tr>
<tr>
<td><code>-json</code></td>
</tr>
<tr>
<td><code>-name</code></td>
</tr>
<tr>
<td><code>-r</code></td>
</tr>
</tbody>
</table>

For example, to get the basic status information:

```
shell> trepctl status
Processing status command...
NAME   VALUE
------- -----
appliedLastEventId : mysql-bin.000007:0000000000001273:0
appliedLastSeqno   : 215
appliedLatency     : 0.53
channels           : 1
clusterName        : default
currentEventId     : mysql-bin.000007:0000000000001273
currentTimeMillis : 1369233160014
dataServerHost     : host1
extensions         : 
latestEpochNumber  : 1000
masterConnectUri   : 
masterListenUri    : thl://host1:1045/
rmiPort            : 10000
role               : master
seqnoType          : java.lang.Long
serviceName        : alpha
serviceType        : local
```

356
Command-line Tools

- simpleServiceName : alpha
- serviceName : default
- sourceId : host1
- state : ONLINE
- timeInStateSeconds : 1874.512
- transitioningTo :
- uptimeSeconds : 1877.823
- version : Tungsten Replicator 6.1.3 build 43

Finished status command...

For more information on the field information output, see Section E.2, “Generated Field Reference”.

The -r option can be used to automatically refresh the output at the specified interval. For example, trepctl status -r 5 will refresh the output every 5 seconds.

8.22.3.20.1. Getting Detailed Status

More detailed information about selected areas of the replicator status can be obtained by using the -name option.

8.22.3.20.1.1. Detailed Status: Channel Assignments

When using a single threaded replicator service, the trepctl status -name channel-assignments will output an empty status. In parallel replication deployments, the trepctl status -name channel-assignments listing will output the list of schemas and their assigned channels within the configured channel quantity configuration. For example, in the output below, only two channels are shown, although five channels were configured for parallel apply:

```shell
$ trepctl status -name channel-assignments
Processing status command (channel-assignments)...
NAME     VALUE      
...       ....
channel : 0
shard_id: test
NAME     VALUE      
...       ....
channel : 0
shard_id: tungsten_alpha
```

Finished status command (channel-assignments)...

8.22.3.20.1.2. Detailed Status: Services

The trepctl status -name services status output shows a list of the currently configure internal services that are defined within the replicator:

```shell
$ trepctl status -name services
Processing status command (services)...
NAME              VALUE      
...              ....
accessFailures  : 0
active          : true
maxChannel      : -1
name            : channel-assignment
storeClass      : com.continuent.tungsten.replicator.channel.ChannelAssignmentService
totalAssignments: 0
```

Finished status command (services)...

8.22.3.20.1.3. Detailed Status: Shards

The trepctl status -name shards status output lists the individual shards in operation, most useful when parallel apply has been configured within the replicator, showing a summary of last applied sequences and the corresponding binlog references.

In an environment not configured with parallel apply, the shards output will just show a single entry

8.22.3.20.1.4. Detailed Status: Stages

The trepctl status -name stages status output lists the individual stages configured within the replicator, showing each stage, configuration, filters and other parameters applied at each replicator stage:

```shell
$ trepctl status -name stages
Processing status command (stages)...
NAME              VALUE      
applier.class      : com.continuent.tungsten.replicator.thl.THLStoreApplier
applier.name       : thl-applier
blockCommitRowCount: 1
committedMinSeqno  : 1
extractor.class    : com.continuent.tungsten.replicator.thl.RemoteTHLExtractor
extractor.name     : thl-remote
name               : remote-to-thl
processedMinSeqno  : -1
processedMinSegno  : -1
```

357
8.22.3.20.1.5. Detailed Status: Stores

The `trepctl status -name stores` status output lists the individual internal stores used for replicating THL data. This includes both physical (on disk) THL storage and in-memory storage. This includes the sequence number, file size and retention information.

For example, the information shown below is taken from a master service, showing the stages, `binlog-to-q` which reads the information from the binary log, and the in-memory `q-to-thl` that writes the information to THL.

```
shell> trepctl status -name stages
Processing status command (stages)...
NAME      VALUE
----      -----
applier.class : com.continuent.tungsten.replicator.storage.InMemoryQueueAdapter
applier.name   : queue
blockCommitRowCount : 1
committedMinSeqno : 224
extractor.class : com.continuent.tungsten.replicator.extractor.mysql.MySQLExtractor
extractor.name   : dbms
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-thl
processedMinSeqno : 224
taskCount       : 1

Finished status command (stages)...
```

When running parallel replication, the output shows the store name, sequence number and status information for each parallel replication channel:

```
shell> trepctl status -name stores
Processing status command (stores)...
NAME                      VALUE
----                      -----
activeSeqno             : 15
doChecksum              : false
flushIntervalMillis     : 0
fsyncOnFlush            : false
logConnectionTimeout    : 28800
logDir                  : /opt/continuent/thl/alpha
logFileRetainMillis     : 604800000
logFileSize             : 100000000
maximumStoredSeqNo      : 16
minimumStoredSeqNo      : 0
```

358
8.22.3.20.1.6. Detailed Status: Tasks

The `trepctl status -name tasks` command outputs the current list of active tasks within a given service, with one block for each stage within the replicator service.
8.22.3.20.1.7. Detailed Status: Watches

The `trepctl status -name watches` command outputs the current list of tasks the replicator is waiting on before a specific action.

For example, if you issue `trepctl offline-deferred -at-seqno x`, the output of `watches` will show the stages waiting on the specific seqno.

The following example show the use of `offline-deferred` and the subsequent resulting output from `watches`:

```
shell> trepctl offline-deferred -at-seqno 234
shell> trepctl status -name watches
Processing status command (watches)...
NAME          VALUE
----          -----  
action        cancel tasks
cancelled      false
committed      false
done          false
matched        [[0:false]]
predicate     SeqnoWatchPredicate seqno=234
stage          remote-to-thl
NAME          VALUE
----          -----  
action        cancel tasks
cancelled      false
committed      false
done          false
matched        [[0:false]]
predicate     SeqnoWatchPredicate seqno=234
stage          thl-to-q
NAME          VALUE
----          -----  
action        cancel tasks
cancelled      false
committed      false
done          false
matched        [[0:false]]
predicate     SeqnoWatchPredicate seqno=234
stage          q-to-dbms
```

8.22.3.20.2. Getting JSON Formatted Status

Status information can also be requested in JSON format. The content of the information is identical, only the representation of the information is different, formatted in a JSON wrapper object, with one key/value pair for each field in the standard status output.

Examples of the JSON output for each status output are provided below. For more information on the fields displayed, see Section E.2, "Generated Field Reference".

`trepctl status` JSON Output

```json
{
  "uptimeSeconds": "2128.682",
  "masterListenUri": "thl://host1:2112/",
  "clusterName": "default",
  "pendingExceptionMessage": "NONE",
  "appliedLastEventId": "mysql-bin.000007:0000000000001353:0",
  "pendingError": "NONE",
  "resourcePrecedence": "99",
  "transitioningTo": "",
  "offlineRequests": "NONE",
  "state": "ONLINE",
  "simpleServiceName": "alpha",
  "extensions": "",
  "pendingErrorEventId": "NONE",
  "sourceId": "host",
  "serviceName": "alpha",
  "version": "Tungsten Replicator 6.1.3 build 43",
  "role": "master",
  "currentTimeMillis": "1369233410874",
  "masterConnectUri": "",
  "rmiPort": "10000",
  "siteName": "default",
}
```
8.22.3.20.2.1. Detailed Status: Channel Assignments JSON Output

shell> trepctl status -name channel-assignments -json
{
  "channels": [
    {
      "channel": "0",
      "shard_id": "cheffy"
    },
    {
      "channel": "0",
      "shard_id": "tungsten_alpha"
    }
  ]
}

8.22.3.20.2.2. Detailed Status: Services JSON Output

shell> trepctl status -name services -json
{
  "totalAssignments": "2",
  "accessFailures": "0",
  "storeClass": "com.continuent.tungsten.replicator.channel.ChannelAssignmentService",
  "name": "channel-assignment",
  "maxChannel": "0"
}

8.22.3.20.2.3. Detailed Status: Shards JSON Output

shell> trepctl status -name shards -json
{
  "stage": "q-to-dbms",
  "appliedLastEventId": "mysql-bin.000007:000007224342;0",
  "appliedLatency": "63.099",
  "appliedLastSeqno": "2514",
  "eventCount": "16",
  "shardId": "cheffy"
}

8.22.3.20.2.4. Detailed Status: Stages JSON Output

shell> trepctl status -name stages -json
{
  "applier.name": "tthl-applier",
  "applier.class": "com.continuent.tungsten.replicator.thl.THLStoreApplier",
  "name": "remote-to-thl",
  "extractor.name": "thl-remote",
  "taskCount": "1",
  "committedMinSeqno": "2504",
  "blockCommitRowCount": "1",
  "processedMinSeqno": "-1",
  "extractor.class": "com.continuent.tungsten.replicator.thl.RemoteTHLExtractor"
},
{
  "applier.name": "parallel-q-applier",
  "applier.class": "com.continuent.tungsten.replicator.storage.InMemoryQueueAdapter",
  "name": "thl-to-q",
  "extractor.name": "thl-extractor",
  "taskCount": "1",
  "committedMinSeqno": "2504",
  "blockCommitRowCount": "10",
  "processedMinSeqno": "-1"
8.22.3.20.2.5. Detailed Status: Stores JSON Output

```
shell> trepctl status -name stores -json
{
"logConnectionTimeout" : "28800",
"doChecksum" : "false",
"name" : "thl",
"flushIntervalMillis" : "0",
"logFileSize" : "100000000",
"logDir" : "/opt/continuent/thl/alpha",
"activeSeqno" : "2561",
"readOnly" : "false",
"timeoutMillis" : "2147483647",
"storeClass" : "com.continuent.tungsten.replicator.thl.THL",
"logFileRetainMillis" : "604800000",
"maximumStoredSeqNo" : "2565",
"minimumStoredSeqNo" : "2047",
"fsyncOnFlush" : "false"
},
{
"storeClass" : "com.continuent.tungsten.replicator.storage.InMemoryQueueStore",
"maxSize" : "10",
"storeSize" : "7",
"name" : "parallel-queue",
"eventCount" : "219"
}
```

8.22.3.20.2.6. 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" : "119",
"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.000007:0000000111424440:0",
"appliedLatency" : "63.787",
"currentLastEventId" : "mysql-bin.000007:0000000111424440:0",
"eventCount" : "119",
"cancelled" : "false"
}
```
8.22.3.20.2.7. Detailed Status: Tasks JSON Output

```shell
trepctl status -name watches -json
```

8.22.3.21. trepctl unload Command

Unload the replicator service.

```
trepctl unload [-y]
```

Unload the replicator service entirely. An interactive prompt is provided to confirm the shutdown:

```
trepctl unload
Do you really want to unload replication service alpha? [yes/NO]
```

To disable the prompt, use the `-y` option:

```
trepctl unload -y
Service unloaded successfully: name=alpha
```

The name of the service unloaded is provided for confirmation.

8.22.3.22. 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>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-applied seqno</code>[363]</td>
<td>Specify the sequence number to be waited for</td>
</tr>
<tr>
<td><code>-limit s</code>[364]</td>
<td>Specify the number of seconds to wait for the operation to complete</td>
</tr>
<tr>
<td><code>-state st</code>[363]</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:

```
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:

```
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` 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
```

If the requested event does not take place before the specified time limit expires, then `trepctl` returns with the message 'Wait timed out!', and an exit status of 1.

## 8.23. The `tpasswd` Command

### Table 8.51. `tpasswd` Common Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--create</code>, <code>-c</code></td>
<td>Creates a new user/password</td>
</tr>
<tr>
<td><code>-delete</code>, <code>-d</code></td>
<td>Delete a user/password combination</td>
</tr>
<tr>
<td><code>-e</code>, <code>--encrypted.password</code></td>
<td>Encrypt the password</td>
</tr>
<tr>
<td><code>-f</code>, <code>--file</code></td>
<td>Specify the location of the security.properties file</td>
</tr>
<tr>
<td><code>-h</code>, <code>--help</code></td>
<td>Display help text</td>
</tr>
<tr>
<td><code>-p</code>, <code>--password.file.location</code></td>
<td>Specify the password file location</td>
</tr>
<tr>
<td><code>-t</code>, <code>--target</code></td>
<td>Specify the target application</td>
</tr>
<tr>
<td><code>--truststore.location</code></td>
<td>Specify the truststore location</td>
</tr>
</tbody>
</table>

## 8.24. The `tungsten_find_orphaned` Command

The `tungsten_find_orphaned` command was added in versions 5.4.1 and 6.1.1

The `tungsten_find_orphaned` command assists with locating orphaned events in both the MySQL binary logs as well as in the THL.

This program is designed to handle three failure scenarios:

1. Orphaned MySQL binary logs that were not extracted into THL before a failover
2. Orphaned THL on old master that did not make it to the new master
3. Orphaned slave THL on new master or online slave of the new master that did not get applied to the database before getting promoted to new master

The default action with no arguments is Scenario 1 to locate orphaned MySQL binary logs which have not been extracted into THL on an old master before recovery.

For Scenarios 1 and 2

By default, the `tungsten_find_orphaned` command expects to be run on a (possibly failed) master before recovery.

Once any new slave THL is appended to any existing extracted THL, the `tungsten_find_orphaned` command will be unable to find the delta between the binlogs and the THL.

If any events are located in the binlogs that do not exist in the THL on disk, they will be counted, and a summary displayed at the end.

Instructions on how to display any actual orphaned events in the binary logs will be provided as well.

For Scenario 2

If the latestEpochNumber from the new master `trepctl status` output is provided via the `-e {seqno}` option, `tungsten_find_orphaned` will check for THL differences from old to new masters.

For Scenario 3

Use `--new` on a NEW master to attempt to automatically locate the latest trepsvc.log file and determine if there is orphaned THL which exists on disk but had not yet been applied to the database.
If `tungsten_find_orphaned` is unable to auto-locate the `trepsvc.log` file, please specify the full path to the latest one using `--log` (or `-l`).

Table 8.52. `tungsten_find_orphaned` Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--debug, -d</code></td>
<td>Enable additional debug output.</td>
</tr>
<tr>
<td><code>--epoch, -e</code></td>
<td>Use the <code>--epoch</code> option on an OLD master to specify the latestEpochNumber from the new master in order to find orphaned THL which did not get copied to the new master. Note: Use <code>tungsten_find_orphaned -E</code> on the new master to obtain the latestEpochNumber (Scenario 2)</td>
</tr>
<tr>
<td><code>--epochget, -E</code></td>
<td>Use option <code>--epochget</code> on a NEW master to locate the latestEpochNumber for use on the OLD master with <code>--epoch</code> [sets quiet mode to true] (Scenario 2)</td>
</tr>
<tr>
<td><code>--help, -h</code></td>
<td>Show help text</td>
</tr>
<tr>
<td><code>--log, -l</code></td>
<td>Use option <code>--log</code> on a NEW master specify the full path to the latest trepsvc.log file. Used when <code>tungsten_find_orphaned</code> is unable to auto-locate trepsvc.log (Scenario 3)</td>
</tr>
<tr>
<td><code>--new, -n</code></td>
<td>Use option <code>--new</code> on a NEW master to attempt to automatically locate the latest trepsvc.log file and determine if there is orphaned THL which exists on disk but had not yet been applied to the database. [Scenario 3]</td>
</tr>
<tr>
<td><code>--path, -p</code></td>
<td>Specify the full path to the executable directory where the <code>thl</code> and <code>trepctl</code> commands are located.</td>
</tr>
<tr>
<td><code>--quiet, -q</code></td>
<td>Prevent final message from being output.</td>
</tr>
<tr>
<td><code>--service, -s</code></td>
<td>Specify the service name to use with the various commands.</td>
</tr>
<tr>
<td><code>--thl</code></td>
<td>Specify the full path to the <code>thl</code> command executable file.</td>
</tr>
<tr>
<td><code>--tpm</code></td>
<td>Specify the full path to the <code>tpm</code> command executable file.</td>
</tr>
<tr>
<td><code>--trepctl</code></td>
<td>Specify the full path to the <code>trepctl</code> command executable file.</td>
</tr>
<tr>
<td><code>--verbose, -v</code></td>
<td>Show verbose output</td>
</tr>
</tbody>
</table>

Examples:

Run on old master to find orphaned binary logs not extracted to thl before failover with verbose output (Scenario 1):

```bash
shell> tungsten_find_orphaned -v
```

Locate THL on an old master that was not transferred to the new master before failover (Scenario 2):

```bash
shell> tungsten_find_orphaned -v -e {latestEpochNumber from new master}
```

Run on new master to find unapplied THL [Scenario 3]:

```bash
shell> tungsten_find_orphaned -v -n
```

8.25. The `tungsten_find_position` Command

The `tungsten_find_position` command was added in versions 5.4.0 and 6.1.0

The `tungsten_find_position` assists with locating event information in the THL and producing a `dsctl set` command as output.

The `tungsten_find_position` command performs the following steps:

- Get the MySQL binary log position to search for from the CLI and validate
- Validate paths and commands
- Load all available service names and validate any specified service against that list
- Check if this Replicator is in the Master role
- Locate the supplied binary log position in the available THL

365
• Parse the THL found, if any
• Generate the dsctl command and display

Table 8.53. **tungsten_find_position** Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--debug</td>
<td>Enable additional debug output.</td>
</tr>
<tr>
<td>--debug</td>
<td>Pass the -file argument to the thl list command to make the location</td>
</tr>
<tr>
<td>--high</td>
<td>operation more efficient - the script will not have to parse ALL of the</td>
</tr>
<tr>
<td>--low</td>
<td>available THL.</td>
</tr>
<tr>
<td>--from</td>
<td>Pass specified seqno to the thl command to signify the ending position to</td>
</tr>
<tr>
<td>--high</td>
<td>scan to. Will scan from the start of all available thl unless the --low</td>
</tr>
<tr>
<td>--low</td>
<td>options are provided</td>
</tr>
<tr>
<td>--path</td>
<td>Pass specified seqno to the thl command to signify the starting position</td>
</tr>
<tr>
<td>--quiet</td>
<td>to scan from. Will continue to scan to the end of all available thl</td>
</tr>
<tr>
<td>--quiet</td>
<td>unless the --low options are provided</td>
</tr>
<tr>
<td>--path</td>
<td>Specify the full path to the executable directory where the thl and</td>
</tr>
<tr>
<td>--quiet</td>
<td>trepctl commands are located.</td>
</tr>
<tr>
<td>--quiet</td>
<td>Specify the service name to use with the various commands.</td>
</tr>
<tr>
<td>--test</td>
<td>Specify the thl command executable file.</td>
</tr>
<tr>
<td>--test</td>
<td>Specify the full path to the trepctl command executable file.</td>
</tr>
<tr>
<td>--verbose</td>
<td>Show verbose output</td>
</tr>
</tbody>
</table>

Below is a sample session:

```
shell> tungsten_find_position mysql-bin.000030:0000000000001981
dsctl set -reset -seqno 4 -epoch 2 -event-id "mysql-bin.000030:0000000000001981;-1" -source-id "db1"
```

### 8.26. The tungsten_find_seqno Command

The **tungsten_find_seqno** command was added in versions 5.4.0 and 6.1.0

The **tungsten_find_seqno** assists with locating event information in the THL and producing a `dsctl set` command as output.

The **tungsten_find_seqno** command performs the following steps:

- Get the Replicator sequence number to search for from the CLI and validate
- Validate paths and commands
- Load all available service names and validate any specified service against that list
- Check if this Replicator is in the Master role
- Locate the supplied seqno in the available THL
- Parse the THL found, if any
- Generate the dsctl command and display

Table 8.54. **tungsten_find_seqno** Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--debug</td>
<td>Enable additional debug output.</td>
</tr>
<tr>
<td>--help</td>
<td>Show help text</td>
</tr>
<tr>
<td>--path</td>
<td>Specify the full path to the executable directory where the thl and</td>
</tr>
<tr>
<td>--quiet</td>
<td>Prevent final message from being output.</td>
</tr>
</tbody>
</table>

366
Command-line Tools

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--service, -s</td>
<td>Specify the service name to use with the various commands.</td>
</tr>
<tr>
<td>--test, -t</td>
<td>Test mode - do not perform the actual update on the files</td>
</tr>
<tr>
<td>-t</td>
<td>Specify the full path to the thl command executable file.</td>
</tr>
<tr>
<td>-trepctl</td>
<td>Specify the full path to the trepctl command executable file.</td>
</tr>
<tr>
<td>-verbose, -v</td>
<td>Show verbose output</td>
</tr>
</tbody>
</table>

Below is a sample session:

```
shell> tungsten_find_seqno 4
dsctl set -reset -seqno 4 -epoch 2 -event-id "mysql-bin.000030:0000000000001981;-1" -source-id "db1"
```

8.27. 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 8.28, "The `tungsten_monitor` Script" to check the cluster against known best practices. It implements the Tungsten Script Interface as well as these additional options.

```
tungsten_health_check [ --dataservices ] [ --diagnostic-package ] [ --directory ] [ --email ] [ --force ] [ --from ] [ --help, -h ] [ --ignore ] [ --info, -i ] [ --json ] [ --lock-dir ] [ --lock-timeout ] [ --mail ] [ --net-ssh-option=key=value ] [ --notice, -n ] [ --show-differences ] [ --subject ] [ --test-failover ] [ --test-recover ] [ --test-switch ] [ --validate ] [ --verbose, -v ]
```

Where:

Table 8.55. `tungsten_health_check` Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--dataservices</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>--email</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</td>
<td>The from address for sending messages</td>
</tr>
<tr>
<td>--help, -h</td>
<td>Show help text</td>
</tr>
<tr>
<td>--ignore</td>
<td>Ignore notices that use this key</td>
</tr>
<tr>
<td>--info, -i</td>
<td>Display info, 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</td>
<td>Directory to store log and lock files in</td>
</tr>
<tr>
<td>--lock-timeout</td>
<td>The number of minutes to sleep a notice after sending it</td>
</tr>
<tr>
<td>--mail</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 SSH communication to other hosts.</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</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.
Command-line Tools

• Check for errors using `tpm validate`

• Check that all servers in the dataservice are running the same version of Continuent Tungsten

The script can be run manually:

```
shell> tungsten_health_check
```

All messages will be sent to `/opt/continuent/share/tungsten_health_check/lastrun.log`.

Sending results via email

The `tungsten_health_check` is able to send you an email when problems are found. It is suggested that you run the script as root so it is able to use the mail program without warnings.

Alerts are cached to prevent them from being sent multiple times and flooding your inbox. You may pass `--reset` to clear out the cache or `--lock-timeout` to adjust the amount of time this cache is kept. The default is 3 hours.

```
shell> tungsten_health_check --from=you@yourcompany.com --to=group@yourcompany.com
```

Showing manual configuration file changes

The `tpm validate` command will fail if you have manually changed a configuration file. The file differences may be added if you include the `--show-differences` argument.

Testing Continuent Tungsten functionality

Continuent Tungsten includes a testing infrastructure that you can use at any time. By adding the `--test-switch`, `--test-failover` or `--test-recover` arguments to the command, we will test these operations on each database server.

**Caution**

This will have an impact on dataservice availability. Limit this operation to maintenance windows or times when you can experience managed outages.

Compatibility

The script only works with MySQL at this time.

8.28. 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.

```
tungsten_monitor [ --check-log ] [ --connector-timeout ] [ --dataservices ] [ --diagnostic-package ] [ --directory ] [ --disk ] [ --elb-script ] [ --email ] [ --force ] [ --help, -h ] [ --ignore ] [ --info, -i ] [ --json ] [ --latency ] [ --lock-dir ] [ --lock-timeout ] [ --mail ] [ --max-backup-age ] [ --net-ssh-option ] [ --notice, -n ] [ --reset ] [ --subject ] [ --validate ] [ --verbose, -v ]
```

Where:

**Table 8.56. `tungsten_monitor` Command-line Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--check-log</code></td>
<td>Email any lines in the log file that match the egrep expression. <code>--check-log=tungsten-manager/log/tm_svc.log:OFFLINE</code></td>
</tr>
<tr>
<td><code>--connector-timeout</code></td>
<td>Number of seconds to wait for a connector response</td>
</tr>
<tr>
<td><code>--dataservices</code></td>
<td>This list of dataservices to monitoring to</td>
</tr>
<tr>
<td><code>--diagnostic-package</code></td>
<td>Create a diagnostic package if any issues are found</td>
</tr>
<tr>
<td><code>--directory</code></td>
<td>The <code>$CONTINUENT_ROOT</code> directory to use for running this command. It will default to the directory you use to run the script.</td>
</tr>
<tr>
<td><code>--disk</code></td>
<td>Display a warning if any disk usage is above this percentage</td>
</tr>
<tr>
<td><code>--elb-script</code></td>
<td>The xinetd script name that is responding to ELB liveness checks</td>
</tr>
<tr>
<td><code>--email</code></td>
<td>Email address to send to when mailing any notifications</td>
</tr>
<tr>
<td><code>--force</code></td>
<td>Continue operation even if script validation fails</td>
</tr>
</tbody>
</table>
Option | Description
--- | ---
--help, -h | Show help text
--ignore | Ignore notices that use this key
--info, -i | Display info, notice, warning, and error messages
--json | Output all messages and the return code as a JSON object
--latency | The maximum allowed latency for replicators
--lock-dir | Directory to store log and lock files in
--lock-timeout | The number of minutes to sleep a notice after sending it
--mail | Path to the mail program to use for sending messages
--max-backup-age | Maximum age in seconds of valid backups
--net-ssh-option | Provide custom SSH options to use for communication to other hosts.
--notice, -n | Display notice, warning, and error messages
--reset | Remove all entries from the lock directory
--subject | Email subject line
--validate | Only run script validation
--verbose, -v | Verbose

General Operation

Each time the tungsten_monitor runs, it will run a standard set of checks. Additional checks may be turned on using command line options.

- Check that all Tungsten services for this host are running
- Check that all replication services and datasources are ONLINE
- Check that replication latency does not exceed a specified amount
- Check that the local connector is responsive
- Check disk usage

An example of adding it to crontab:

```
shell> crontab -l
10 * * * * /opt/continuent/tungsten/cluster-home/bin/tungsten_monitor >/dev/null 2>/dev/null
```

All messages will be sent to `/opt/continuent/share/tungsten_monitor/lastrun.log`.

Sending results via email

The `tungsten_monitor` is able to send you an email when problems are found. It is suggested that you run the script as root so it is able to use the mail program without warnings.

Alerts are cached to prevent them from being sent multiple times and flooding your inbox. You may pass --reset to clear out the cache or --lock-timeout to adjust the amount of time this cache is kept. The default is 3 hours.

```
shell> crontab -l
10 * * * * /opt/continuent/tungsten/cluster-home/bin/tungsten_monitor --from=you@yourcompany.com --to=group@yourcompany.com >/dev/null 2>/dev/null
```

Monitoring log files

The `tungsten_monitor` can optionally monitor log files for certain keywords. This example will alert you to any lines in `trepsvc.log` that include `OFFLINE`.

```
shell> tungsten_monitor --check-log=tungsten-replicator/log/trepsvc.log:OFFLINE
```

Monitoring backup status

Knowing you have a recent backup is an important part any Tungsten deployment. The `tungsten_monitor` will look for the latest backup across all datasources and compare it to the value `--max-backup-age`. This example will let you know if a valid backup has not been taken in 3 days.

```
shell> tungsten_monitor --max-backup-age=259200
```
Compatibility

The script only works with MySQL at this time.

8.29. The tungsten_nagios_backups Command

8.30. The tungsten_nagios_online Command

8.31. The tungsten_post_process Command

The **tungsten_post_process** command was added in versions 5.4.0 and 6.0.5

The **tungsten_post_process** assists with the graceful maintenance of the static cross-site replicator configuration files on disk.

The **tungsten_post_process** command performs the following steps:

- Locates, reads and parses the various INI configuration files.

  Below is the list of possible INI files and file match patterns:

  - `{HOME}/tungsten.ini`
  - `/etc/tungsten/tungsten*.ini`
  
  **Note**

  Please note that all files in the `/etc/tungsten` directory starting with `tungsten` and ending in `.ini` will be used.

  - `/etc/tungsten.ini`

  - Identifies cross-site services that are local to the node, as well as the main local service.

  **Warning**

  Only Replicator options and properties for specific entry types are currently supported.

  The supported stanzas are as follows:

  - `{service}.replicator`
  - `{service}_from_{service}`

  Below is an example INI file showing section identifiers only:

  ```ini
  [defaults]
  [defaults.replicator]
  [east]
  [east.replicator]
  [west]
  [east_from_west]
  [west_from_east]
  ```

  Of the above example section identifiers, only the following would be used by **tungsten_post_process**:

  - `east.replicator`
  - `east_from_west`
  - `west_from_east`

  The remaining example section identifiers (`defaults`, `defaults.replicator`, `east` and `west`) would be ignored.

  - Gathers configuration options and properties defined for each identified local service.
  - Locates the associated configuration file on disk and gets the existing value for the option key.
  - Interactively prompts for confirmation. This may be bypassed using the `-y` option to **tungsten_post_process**.
  - Filter out any files that are not static replicator configurations.
• The `tungsten_post_process` command will then do one edit-in-place per ini option per local service.

• The script will check to make sure the value has been updated.

• Finally, there will be a summary message and helpful instructions about next steps. This may be bypassed using the `-q` option to `tungsten_post_process`.

### Table 8.57. `tungsten_post_process` Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--debug</code>, <code>-d</code></td>
<td>Enable additional debug output.</td>
</tr>
<tr>
<td><code>--help</code>, <code>-h</code></td>
<td>Show help text.</td>
</tr>
<tr>
<td><code>--quiet</code>, <code>-q</code></td>
<td>Prevent final message from being output.</td>
</tr>
<tr>
<td><code>--test</code>, <code>-t</code></td>
<td>Test mode - do not perform the actual update on the files</td>
</tr>
<tr>
<td><code>--verbose</code>, <code>-v</code></td>
<td>Show verbose output</td>
</tr>
<tr>
<td><code>--yes</code>, <code>-y</code></td>
<td>Bypass interactive confirmation prompt</td>
</tr>
</tbody>
</table>

Below is a sample session:

```
shell> tungsten_post_process
About to update the following config file(s):
/opt/continuent/tungsten/tungsten-replicator/conf/static-east_from_north.properties
/opt/continuent/tungsten/tungsten-replicator/conf/static-east_from_west.properties
Do you wish to continue? [y/N]
y
Done!
tungsten_post_process updated 8 configuration options successfully and skipped 6
```

Pick one node and set Maintenance mode so the manager does not try to bring the replicator online when you don't want it to:
```
echo "set policy maintenance" | cctrl
```

Then take the Replicator offline gracefully on one node:
```
trepctl -all-services offline
```

Restart the Replicator:
```
replicator restart
```

Bring all services online:
```
trepctl -all-services online
```

Check the status:
```
trepctl services
```

Repeat as needed.

When all nodes are done, pick one node and execute:
```
echo "set policy automatic" | cctrl
```

### 8.32. The `tungsten_prep_upgrade` Script

The script was added in Continuent Tungsten 6.0.0

The `tungsten_prep_upgrade` command is a utility script which assists in the upgrade process.

```
tungsten_prep_upgrade [--all|--service] {service} [--path {fullpath_to_replicator_dir}] [arguments]
```

Where:

### Table 8.58. `tungsten_prep_upgrade` Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--alldb</code></td>
<td>Loop through all services on this node [based on existing tungsten_* schemas].</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>--debug</td>
<td>Debug mode is VERY chatty, avoid it unless you really need it.</td>
</tr>
<tr>
<td>--dump, -d</td>
<td>Backup tracking databases using mysqldump [default: tungsten_{service}]</td>
</tr>
<tr>
<td>--dumpdb, -D</td>
<td>Specify database to dump</td>
</tr>
<tr>
<td>--force, -f</td>
<td>Force the operation.</td>
</tr>
<tr>
<td>--help, -h</td>
<td>Show help text</td>
</tr>
<tr>
<td>--host, -H</td>
<td>Specify the database hostname or IP address</td>
</tr>
<tr>
<td>--keep, --get, -g, -k</td>
<td>Get the current tracking position schema for the specified service and save it as json to a text file</td>
</tr>
<tr>
<td>--offline, -o</td>
<td>Take the specific service to the offline-deferred state at-heartbeat [default: offline_for_upg]</td>
</tr>
<tr>
<td>--offlinehb, -O</td>
<td>Specify the name of the heartbeat for the offline-deferred operation</td>
</tr>
<tr>
<td>--password, -w</td>
<td>Specify the database password</td>
</tr>
<tr>
<td>--path, -p</td>
<td>Full path to the cross-site replicator directory [default: /opt/replicator]</td>
</tr>
<tr>
<td>--port, -P</td>
<td>Specify the database listener port [default: 13306]</td>
</tr>
<tr>
<td>--restore, -r</td>
<td>Load the tracking database backup for the specified service.</td>
</tr>
<tr>
<td>--service, -s</td>
<td>Specify the service name to act upon.</td>
</tr>
<tr>
<td>--start</td>
<td>Bring up the cross-site replicator process</td>
</tr>
<tr>
<td>--stop</td>
<td>Gracefully shut down the cross-site replicator process</td>
</tr>
<tr>
<td>--targetdir, -T</td>
<td>Specify directory target for dump</td>
</tr>
<tr>
<td>--user, -u</td>
<td>Specify the database user</td>
</tr>
<tr>
<td>--verbose, -v</td>
<td>Show verbose output</td>
</tr>
</tbody>
</table>

If --{service|all} is not specified, `tungsten_prep_upgrade` will attempt to derive a list of one or more service names from `trepctl services`.

For operations that require MySQL access, `tungsten_prep_upgrade` will attempt to auto-locate the database user and password from `tpm reverse`.

Below are various examples:

- Take all replicator services offline automatically, or take a specific service offline:
  ```shell```
  ```
  shell> tungsten_prep_upgrade -o
  or:
  shell> tungsten_prep_upgrade --service london --offline
  shell> tungsten_prep_upgrade --service tokyo --offline
  ```
  ```
  ```

  Note
  To invoke the actual deferred offline set with --offline, use the below CLUSTER-specific `trepctl` command [i.e. from /opt/continuent, not /opt/replicator] on the master hosts within each cluster:
  ```shell```
  ```
  shell> trepctl heartbeat -name offline_for_upg
  ```

- Get (keep) the current tracking position schema for the specified service and save it as json to a text file [default: ~/position-{service}-YYYYMMDDHHMMSS.txt]
  ```shell```
  ```
  shell> tungsten_prep_upgrade -g
  or:
  shell> tungsten_prep_upgrade --service nyc --get
  shell> tungsten_prep_upgrade --service tokyo --get
  (NOTE: saves to ~/position-{tokyo-YYYYMMDDHHMMSS}.txt)
  ```

- Gracefully shut down the cross-site replicator process:
  ```shell```
  ```
  shell> tungsten_prep_upgrade --stop
  ```

- Backup (dump) tracking databases using mysqldump [default: tungsten_{service}]
  ```shell```
  ```
  shell> tungsten_prep_upgrade -d --alldb
  ```

372
Command-line Tools

- Load (restore) the tracking database backup for the specified service. --all is unavailable with --restore.

**Note**
A restore may take place after the cross-site replicator is uninstalled, and so certain information is required on the command line (i.e. service, user and password)

### 8.33. The `tungsten_provision_thl` Command

The `tungsten_provision_thl` command can be used to generate the THL required to provision a database with information from a MySQL master to a slave. Because of the way the tool works, the tool is most useful in heterogeneous deployments where the data must be formatted and processed by the replicator for effective loading into the target database.

The tool operates as follows:
1. A `mysqldump` of the current database is taken from the current master.
2. The generated SQL from `mysqldump` is then modified so that the data is loaded into tables using the `BLACKHOLE` engine type. These statements still generate information within the MySQL binary log, but do not create any data.
3. A sandbox MySQL server is started, using the MySQL Sandbox tool.
4. A duplicate replicator is started, pointing to the sandbox MySQL instance, but sharing the same THL port and THL directory.
5. The modified SQL from `mysqldump` is loaded, generating events in the binary log which are extracted by the sandbox replicator.

Because the sandbox replicator works on the same THL port as the standard master replicator, the slaves will read the THL from the sandbox replicator. Also, because it uses the same THL directory, the THL will be written into additional THL files. It doesn't matter whether there are existing THL data files, the new THL will be appended into files in the same directory.

The tool has the following pre-requisites, in addition to the main Appendix B, Prerequisites for Tungsten Replicator:
- A tarball of the Tungsten Replicator must be available so that the duplicate replicator can be created. The full path to the file should be used.
- The MySQL Sandbox tool must have been installed. For more information, see MySQL Sandbox.

Installing MySQL Sandbox requires the `ExtUtils::MakeMaker` and `Test::Simple` Perl modules. You may install these through CPAN or a package manager:

```
shell> yum install -y perl-ExtUtils-MakeMaker perl-Test-Simple
```

After those packages are available, you can proceed with building MySQL Sandbox and installing it. If you do not have sudo access, make sure that `/MySQL-Sandbox-3.0.44/bin` is added to `$PATH`

```
shell> cd ~
shell> wget https://launchpad.net/mysql-sandbox/mysql-sandbox-3/mysql-sandbox-3/+download/MySQL-Sandbox-3.0.44.tar.gz
shell> tar -xzf MySQL-Sandbox-3.0.44.tar.gz
shell> cd MySQL-Sandbox-3.0.44
shell> perl Makefile.PL
shell> make
shell> make test
shell> sudo make install
```

- A tarball of a MySQL release must be available to create the sandbox MySQL environment. The release should match the installed version of MySQL. The full path to the file should be used.
- The replicator deployment should already be installed. The master should be `OFFLINE`, but the command can place the replicator offline automatically as part of the provisioning process.

Once these prerequisites have been met, the basic method of executing the command is to specify the location of the Tungsten Replicator tarball, MySQL tarball and the databases that you want to provision:

```
shell> tungsten_provision_thl
```
The command reports the MySQL binary log point and host on which the THL has been provisioned. Put the Tungsten Replicator back online from the reported position:

```
shell> trepctl online -from-event 000025:493
```

The Tungsten Replicator will start extracting from that position and continue with any additional changes. Check all slaves to be sure they are online. The slaves services will process all extracted entries.

### 8.33.1. Provisioning from RDS

The `tungsten_provision_thl` script is designed to run from a replication master connected to a standard MySQL instance. The standard commands will not work if you are using RDS as a master.

The simplest method is to add the `--extract-from` [375] argument to your command. This will make the script compatible with RDS. The drawback is that we are not able to guarantee a consistent provisioning snapshot in RDS unless changes to the database are stopped. The script will monitor the binary log position during the provisioning process and alert you if there are changes. After the script completes, run `trepctl online` to resume extraction from the master at the current binary log position.

```
shell> tungsten_provision_thl \
   --extract-from=rds \
   --tungsten-replicator-package=/home/tungsten/tungsten-replicator-3.0.0-254.tar.gz \
   --mysql-package=/home/tungsten/mysql-5.6.20-linux-glibc2.5-x86_64.tar.gz \
   --schemas=test
```

If you aren’t able to stop access to the database, the script can provision from an RDS Read Replica. Before running `tungsten_provision_thl`, replication to the replica must be stopped. This may be done by running `CALL mysql.rds_stop_replication();` in an RDS shell. Call `tungsten_provision_thl` with the `--extract-from` [375] and `--extract-from-host` [375] arguments. The script will read the correct master position based on the slave replication position. After completion, resume extraction from the master using the standard procedure.

```
# Run 'CALL mysql.rds_stop_replication();' on the RDS Read Replica
shell> tungsten_provision_thl \
   --extract-from=rds-read-replica \
   --extract-from-host=rds-host2 \
   --tungsten-replicator-package=/home/tungsten/tungsten-replicator-3.0.0-254.tar.gz \
   --mysql-package=/home/tungsten/mysql-5.6.20-linux-glibc2.5-x86_64.tar.gz \
   --schemas=test
```

### 8.33.2. `tungsten_provision_thl` Reference

The format of the command is:

```
```

Where:

- `--cleanup-on-failure` [374]
  - Option: `--cleanup-on-failure` [374]
  - Description: Cleanup the sandbox installations when the provision process fails
  - Value Type: boolean
  - Default: false

- `--clear-logs` [374]
  - Option: `--clear-logs` [374]
  - Description: Delete all THL and relay logs for the service
  - Value Type: boolean
  - Default: false
### Command-line Tools

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>--directory</td>
<td>Use this installed Tungsten directory as the base for all operations</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>--extract-from</td>
<td>The type of server you are going to extract from</td>
<td>string</td>
<td>mysql-native-slave, rds, rds-read-replica, tungsten-slave</td>
</tr>
<tr>
<td></td>
<td>A MySQL native slave with binary logging enabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>An Amazon RDS instance</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>An Amazon RDS read replica instance</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>An instance with Tungsten Clustering already installed with generated THL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--extract-from-host</td>
<td>The hostname of a different MySQL server that will be used as the source for mysqldump</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The hostname of a different MySQL server that will be used as the source for mysqldump. When given, the script will use \texttt{SHOW SLAVE STATUS} to determine the binary log position on the master server. You must run \texttt{STOP SLAVE} prior to executing \texttt{tungsten_provision_thl}.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--extract-from-port</td>
<td>The listening port of a different MySQL server that will be used as the source for mysqldump</td>
<td>numeric</td>
<td></td>
</tr>
<tr>
<td>--help</td>
<td>Display the help message</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>--info</td>
<td>Provide information-level messages</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>--java-file-encoding</td>
<td>Java platform charset</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>--json</td>
<td>Provide return code and logging messages as a JSON object after the script finishes</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>
### Command-line Tools

<table>
<thead>
<tr>
<th>Value Type</th>
<th>string</th>
</tr>
</thead>
</table>

- **--mysql-package [376]**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mysql-package [376]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The location of a the MySQL tar.gz package</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

- **--net-ssh-option [376]**

<table>
<thead>
<tr>
<th>Option</th>
<th>--net-ssh-option [376]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets additional options for SSH usage by the system, such as port numbers and passwords.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Default</td>
<td>default</td>
</tr>
</tbody>
</table>

Sets options for the `Net::SSH` Ruby module. This allows you to set explicit SSH options, such as changing the default network communication port, password, or other information. For example, using `--net-ssh-option=port=80 [376]` will use port 80 for SSH communication in place of the default port 22.

For more information on the options, see [http://net-ssh.github.com/ssh/v2/api/classes/Net/SSH.html#M000002](http://net-ssh.github.com/ssh/v2/api/classes/Net/SSH.html#M000002).

- **--notice [376]**

<table>
<thead>
<tr>
<th>Option</th>
<th>--notice [376]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Provide notice-level messages</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

- **--offline [376]**

<table>
<thead>
<tr>
<th>Option</th>
<th>--offline [376]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Put required replication services offline before processing</td>
</tr>
<tr>
<td>Value Type</td>
<td>boolean</td>
</tr>
<tr>
<td>Default</td>
<td>false</td>
</tr>
</tbody>
</table>

- **--online [376]**

<table>
<thead>
<tr>
<th>Option</th>
<th>--online [376]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Put required replication services online after successful processing</td>
</tr>
<tr>
<td>Value Type</td>
<td>boolean</td>
</tr>
<tr>
<td>Default</td>
<td>false</td>
</tr>
</tbody>
</table>

- **--quiet [376]**

<table>
<thead>
<tr>
<th>Option</th>
<th>--quiet [376]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Execute with the minimum of output</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

- **--sandbox-directory [376]**

<table>
<thead>
<tr>
<th>Option</th>
<th>--sandbox-directory [376]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The location to use for storing the temporary replicator and MySQL server</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

- **--sandbox-mysql-port [376]**

<table>
<thead>
<tr>
<th>Option</th>
<th>--sandbox-mysql-port [376]</th>
</tr>
</thead>
</table>
### Command-line Tools

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--sandbox-password</code></td>
<td>The password for the MySQL sandbox user</td>
<td>string</td>
<td>secret</td>
</tr>
<tr>
<td><code>--sandbox-rmi-port</code></td>
<td>The listening port for the temporary Tungsten Replicator</td>
<td>string</td>
<td>10002</td>
</tr>
<tr>
<td><code>--sandbox-user</code></td>
<td>The MySQL user to create and use in the MySQL Sandbox</td>
<td>string</td>
<td>tungsten</td>
</tr>
<tr>
<td><code>--schemas</code></td>
<td>The provision process will be limited to these schemas</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td><code>--service</code></td>
<td>Replication service to read information from</td>
<td>string</td>
<td>alpha</td>
</tr>
<tr>
<td><code>--tungsten-replicator-package</code></td>
<td>The location of a fresh Tungsten Replicator tar.gz package</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td><code>--validate</code></td>
<td>Run the script validation for the provided options and files</td>
<td>boolean</td>
<td>false</td>
</tr>
<tr>
<td><code>--verbose</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Aliases:**
- `-v`
8.34. 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 ] [ -d ] [ --directory ] [ -f, --force ] [ --help, -h ] [ --info, -i ] [ --json ] [ --mysqldump ] [ --net-ssh-option ] [ --notice, -n ] [ --offline ] [ --offline-timeout ] [ --online ] [ --service ] [ --source ] [ --source-directory ] [ --validate ] [ --verbose, -v ] [ --xtrabackup ]
```

Where:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--clear-logs</code></td>
<td>Delete all THL and relay logs for the service</td>
</tr>
<tr>
<td><code>--directory</code></td>
<td>Use the MySQL data directory for staging and preparation</td>
</tr>
<tr>
<td><code>--force, -f</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>
<tr>
<td><code>--json</code></td>
<td>Output all messages and the return code as a JSON object</td>
</tr>
<tr>
<td><code>--mysqldump</code></td>
<td>Use mysqldump for generating the information</td>
</tr>
<tr>
<td><code>--net-ssh-option</code></td>
<td>Provide custom SSH options to use for SSH communication to other hosts.</td>
</tr>
<tr>
<td><code>--notice, -n</code></td>
<td>Display notice, warning, and error messages</td>
</tr>
<tr>
<td><code>--offline</code></td>
<td>Put required replication services offline before processing</td>
</tr>
<tr>
<td><code>--offline-timeout</code></td>
<td>Put required replication services offline before processing</td>
</tr>
<tr>
<td><code>--online</code></td>
<td>Put required replication services online after successful processing</td>
</tr>
<tr>
<td><code>--service</code></td>
<td>Replication service to read information from</td>
</tr>
<tr>
<td><code>--source</code></td>
<td>Server to use as a source for the backup</td>
</tr>
<tr>
<td><code>--source-directory</code></td>
<td>Directory on <code>--source</code> to find installed software</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>
<tr>
<td><code>--xtrabackup</code></td>
<td>Use xtrabackup for generating the information</td>
</tr>
</tbody>
</table>

The script will automatically put all replication services offline prior to beginning. If the services were online, the script will put them back online following a successful completion. All THL logs will be cleared prior to going online. The replicator will start replication from the position reflected on the source host.

Provisioning will fail from a slave that is stopped, or if the slave is not in either the **ONLINE** or **OFFLINE:NORMAL** (in [Tungsten Replicator 6.1 Manual]) states. This can be overridden by using the `-f` or `--force` options.

When provisioning masters, for example in fan-in or multimaster topologies, 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.

For more information on Fan-In topologies, see Deploying a Fan-In Topology (in [Tungsten Replicator 6.1 Manual]).

The `--service` argument is used to determine which database server should be provisioned. If there are multiple services defined in the replicator and one of those is a master, the master service must be specified.

If the installation directory on `--source` is different from the target, specify `--source-directory` to specify where it can be found. This option should point to an installation that is running the `--service` replication service. The `--source-directory` option is not required if the software is installed to the same directory on both servers.
Using xtrabackup

The script will use Xtrabackup by default. It will run validation prior to starting to make sure the needed scripts are available. The provision process will run Xtrabackup on the source server and stream the contents to the server you are provisioning. Passing the --direct option will empty the MySQL data directory prior to doing the backup and place the streaming backup there. After taking the backup, the script will prepare the directory and restart the MySQL server.

Using mysqldump

If you have a small dataset or don’t have Xtrabackup, you may pass the --mysqldump option to use it. It implements the Tungsten Script Interface as well as these additional options.

Compatibility

The script only works with MySQL at this time.

8.35. 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.

```
tungsten_read_master_events [ --directory ] [ --force ] [ --help, -h ] [ --info, -i ] [ --json ] [ --low ] [ --net-ssh-option ] [ --notice, -n ] [ --service ] [ --source ] [ --validate ] [ --verbose, -v ]
```

Where:

Table 8.60. `tungsten_read_master_events` Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>--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>--high</td>
<td>Display events ending with this sequence number</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>--low</td>
<td>Display events starting with this sequence number</td>
</tr>
<tr>
<td>--net-ssh-option</td>
<td>Provide custom SSH options to use for SSH communication to other hosts.</td>
</tr>
<tr>
<td>--notice, -n</td>
<td>Display notice, warning, and error messages</td>
</tr>
<tr>
<td>--service</td>
<td>Replication service to read information from</td>
</tr>
<tr>
<td>--source</td>
<td>Determine metadata for the --after, --low, --high statements from this host</td>
</tr>
<tr>
<td>--validate</td>
<td>Only run script validation</td>
</tr>
<tr>
<td>--verbose, -v</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
8.36. The `tungsten_reset_manager` Command

The `tungsten_reset_manager` command was added in versions 5.3.7 and 6.0.5.

- The `tungsten_reset_manager` command stops the manager and then deletes the manager's dynamic state files.
- For a proper service-wide reset to work, this command needs to be run on every node within the local cluster to ensure all managers have been stopped at the same time. This prevents any manager from storing an old state in memory.
- The dynamic state files are automatically rebuilt when the manager is started.
- The directory needs to be emptied, for example, given a service name of `alpha` and a TUNGSTEN_HOME of `/opt/continuent`, the full path would be `/opt/continuent/tungsten/cluster-home/conf/cluster/alpha/datasource/`.
- The `tungsten_reset_manager` command requires the cluster to be in MAINTENANCE mode first.

```shell
shell> tungsten_reset_manager
Policy is not MAINTENANCE.
Please select one node only and execute 'echo 'set policy maintenance'' | cctrl'
If you are sure the policy is maintenance you may proceed by using -f or --force
```

- By default, the command expects the manager to be running so the policy may be checked. If you are sure the policy is maintenance you may proceed by using either the `-f` or `--force` flags.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--debug</code>, <code>-d</code></td>
<td>Enable additional debug output.</td>
</tr>
<tr>
<td><code>--force</code>, <code>-f</code></td>
<td>Bypass manager and policy checks</td>
</tr>
<tr>
<td><code>--help</code>, <code>-h</code></td>
<td>Show help text</td>
</tr>
<tr>
<td><code>--verbose</code>, <code>-v</code></td>
<td>Show verbose output</td>
</tr>
<tr>
<td><code>--yes</code>, <code>-y</code></td>
<td>Bypass interactive confirmation prompt</td>
</tr>
</tbody>
</table>

Below is a sample session.

First, set policy to maintenance on a single node only:

```shell
shell> echo 'set policy maintenance' | cctrl
Tungsten Clustering 6.0.4 build 27
north: session established, encryption=false, authentication=false
[LOGICAL] /alpha > set policy maintenance
policy mode is now MAINTENANCE
[LOGICAL] /alpha > Exiting...
```

Warning
---
Enable maintenance mode on a single host only.

On all nodes in the local cluster, execute the `tungsten_reset_manager` command:

```shell
shell> tungsten_reset_manager
About to stop the manager and empty directory:
/./opt/continuent/tungsten/cluster-home/conf/cluster/north/datasource

Do you wish to continue? [y/N] y
Stopping Tungsten Manager Service...
Stopped Tungsten Manager Service.
Manager directory /opt/continuent/tungsten/cluster-home/conf/cluster/north/datasource cleared.
Done!

Make sure all nodes have been cleared first, then execute 'manager start' on each node one-by-one, starting with the master.
```
When all managers are running, pick one node and execute `echo 'set policy automatic' | cctrl`

Once the `tungsten_reset_manager` command has completed successfully on all nodes in the local cluster, execute the manager start command on all nodes:

```
shell> manager start
Starting Tungsten Manager Service...
waiting for Tungsten Manager Service..........
running: PID:3562
```

Once the manager has started successfully on all nodes in the local cluster, set policy to automatic on a single node only:

```
shell> echo "set policy automatic" | cctrl

Tungsten Clustering 6.8.4 build 27
north: session established, encryption=false, authentication=false
[LOGICAL] /alpha > set policy automatic
policy mode is now AUTOMATIC
[LOGICAL] /alpha >
Exiting...
```

Warning

Enable automatic mode on a single host only.

At this point, the manager state should be completely back to normal. Check it using the `cctrl ls` command.

**8.37. The `tungsten_send_diag` Script**

The script was added in Continuent Tungsten 5.2.0

The `tungsten_send_diag` command is a utility script which assists in the upload of files to Continuent support.

`tungsten_send_diag` may be used in place of the Section 9.5.3, "tpm diag Command" to generate a diagnostic package.

`tungsten_send_diag [ --case, -c ][ --contentType ][ --debug ][ --diag, -d ][ --email, -e ][ --file, -f ][ --help, -h ][ --tpm, -t ][ --verbose, -v ]`

Where:

**Table 8.62. `tungsten_send_diag` Command-line Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--case, -c</td>
<td>Specify the support case number</td>
</tr>
<tr>
<td>--contentType</td>
<td>Specify the Content-Type for a file you are uploading</td>
</tr>
<tr>
<td>--debug</td>
<td>Debug mode is VERY chatty, avoid it unless you really need it.</td>
</tr>
<tr>
<td>--diag, -d</td>
<td>Automatically generate a tpm diag zip file and upload it</td>
</tr>
<tr>
<td>--email, -e</td>
<td>Email address to embed into the uploaded file name</td>
</tr>
<tr>
<td>--file, -f</td>
<td>File name to upload</td>
</tr>
<tr>
<td>--help, -h</td>
<td>Show help text</td>
</tr>
<tr>
<td>--tpm, -t</td>
<td>Full path to the tpm command you wish to use to execute a tpm diag</td>
</tr>
<tr>
<td>--verbose, -v</td>
<td>Show verbose output</td>
</tr>
</tbody>
</table>

You must specify either `--diag`, `--tpm`, or `--file`, but not both. For example:

```
shell> tungsten_send_diag --diag -c 1234
```

You must specify either `--email` or `--case`, and you may provide both if you wish. For example:

```
shell> tungsten_send_diag --email you@yourdomain.com --case 1234
```

Using `--tpm` to specify one or more tpm commands implies the `--diag` option, you do not need to specify `--diag` if you use `--tpm` [or `-t`]. For example:

```
shell> tungsten_send_diag --case 1234 -t /opt/replicator/tungsten/tools/tpm
```

You may generate multiple diags by specifying multiple `tpm` binaries with multiple arguments, i.e.:

```
shell> tungsten_send_diag --case 1234 -t /opt/continuent/tungsten/tools/tpm -t /opt/replicator/tungsten/tools/tpm
```
8.38. 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.

```
tungsten_set_position [ --clear-logs ][ --epoch ][ --event-id ][ --high ][ --low ][ --offline ][ --offline-timeout ][ --online ][ --replicate-statements ][ --seqno ][ --service ][ --source ][ --source-directory ][ --source-id ][ --sql ]
```

Where:

Table 8.63. `tungsten_set_position` 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>--epoch</code></td>
<td>The epoch number to use for updating the <code>trep_commit_seqno</code> table</td>
</tr>
<tr>
<td><code>--event-id</code></td>
<td>The event id to use for updating the <code>trep_commit_seqno</code> table</td>
</tr>
<tr>
<td><code>--high</code></td>
<td>Display events ending with this sequence number</td>
</tr>
<tr>
<td><code>--low</code></td>
<td>Display events starting with this sequence number</td>
</tr>
<tr>
<td><code>--offline</code></td>
<td>Put required replication services offline before processing</td>
</tr>
<tr>
<td><code>--offline-timeout</code></td>
<td>Put required replication services offline before processing</td>
</tr>
<tr>
<td><code>--online</code></td>
<td>Put required replication services online after successful processing</td>
</tr>
<tr>
<td><code>--replicate-statements</code></td>
<td>Execute the events so they will be replicated if the service is a master</td>
</tr>
<tr>
<td><code>--seqno</code></td>
<td>The sequence number to use for updating the <code>trep_commit_seqno</code> table</td>
</tr>
<tr>
<td><code>--service</code></td>
<td>Replication service to read information from</td>
</tr>
<tr>
<td><code>--source</code></td>
<td>Determine metadata for the <code>--after</code>, <code>--low</code>, <code>--high</code> statements from this host</td>
</tr>
<tr>
<td><code>--source-directory</code></td>
<td>Directory on <code>--source</code> to find installed software</td>
</tr>
<tr>
<td><code>--source-id</code></td>
<td>The source id to use for updating the <code>trep_commit_seqno</code> table</td>
</tr>
<tr>
<td><code>--sql</code></td>
<td>Only output the SQL statements needed to update the schema</td>
</tr>
</tbody>
</table>

General Operation

In order to update the `trep_commit_seqno` table, the replication service must be offline. You may pass the `--offline` option to do that for you. The `--online` option will put the replication services back online at successful completion.

In most cases you will want to pass the `--clear-logs` argument so that all THL and relay logs are delete from the server following provisioning. This ensures that any corrupted or inconsistent THL records are removed prior to replication coming back online.

The `--service` argument is used to determine which database server should be provisioned.

If the installation directory on `--source` is different from the target, specify `--source-directory` to specify where it can be found. This option should point to an installation that is running the `--service` replication service. The `--source-directory` option is not required if the software is installed to the same directory on both servers.

This command will fail if there is more than one record in the `trep_commit_seqno` table. This may happen if parallel replication does not stop cleanly. You may bypass that error with the `--force` option.

Update trep_commit_seqno with information from a THL event

This will read the THL information from the host specified as `--source`.

```
shell> tungsten_set_position --seqno=5273 --source=db1
```

Update trep_commit_seqno with specific information

The script will also accept specific values to update the `trep_commit_seqno` table. This may be used when bringing a new master service online or when the THL event is no longer available.

```
shell> tungsten_set_position --seqno=5273 --epoch=5264
```
Command-line Tools

```
--source-id=db1
shell> tungsten_set_position --seqno=5273 --epoch=5264
--source-id=db1 --event-id=mysql-bin.000025:0000000000000421
```

**Compatibility**

The script only works with MySQL at this time.

### 8.39. The undeployall Command

The `undeployall` command removes startup the startup and reboot scripts created by `deployall`, disabling automatic startup and shutdown of available services.

To use, the tool should be executed with superuser privileges, either directly using `sudo`, or by logging in as the superuser and running the command directly:

```
shell> sudo deployall
Removing any system startup links for /etc/init.d/treplicator ...
/etc/rc0.d/K80treplicator
/etc/rc1.d/K80treplicator
/etc/rc2.d/S80treplicator
/etc/rc3.d/S80treplicator
/etc/rc4.d/S80treplicator
/etc/rc5.d/S80treplicator
/etc/rc6.d/K80treplicator
```

To enable the scripts on the system, use `deployall`.

### 8.40. The updateCDC.sh Command

The `updateCDC.sh` script updates and existing configuration for Oracle CDC, updating for new tables and user/password configuration.

The script accepts one argument, the filename of the configuration file that will define the CDC configuration. The file accepts the parameters as listed in Table 8.36, "setupCDC.conf Configuration Options".

To use, supply the name of the configuration file:

```
shell> ./updateCDC.sh sample.conf
```

### 8.41. The zabbix_tungsten_latency Command

### 8.42. The zabbix_tungsten_online Command

### 8.43. The zabbix_tungsten_progress Command

### 8.44. The zabbix_tungsten_services Command
Chapter 9. The tpm Deployment Command

tpm, or the Tungsten Package Manager, is a complete configuration, installation and deployment tool for Tungsten Clustering. It includes some utility commands to simplify those and other processes. In order to provide a stable system, all configuration changes must be completed using tpm. tpm makes use of ssh enabled communication and the sudo support as required by the Appendix B, Prerequisites.

tpm can operate in two different ways when performing a deployment:

• tpm staging configuration — a tpm configuration is created by defining the command-line arguments that define the deployment type, structure and any additional parameters. tpm then installs all the software on all the required hosts by using ssh to distribute Tungsten Clustering and the configuration, and optionally automatically starts the services on each host. tpm manages the entire deployment, configuration and upgrade procedure.

• tpm ini configuration — tpm uses an ini file to configure the service on the local host. The ini file must be create on each host that will run Tungsten Clustering. tpm only manages the services on the local host; in a multi-host deployment, upgrades, updates, and configuration must be handled separately on each host.

For a more detailed comparison of the two systems, see Section 9.1, “Comparing Staging and ini tpm Methods”.

During the staging-based configuration, installation and deployment, the tpm tool works as follows:

• tpm creates a local configuration file that contains the basic configuration information required by tpm. This configuration declares the basic parameters, such as the list of hosts, topology requirements, username and password information. These parameters describe top-level information, which tpm translates into more detailed configuration according to the topology and other settings.

• Within staging-based configuration, each host is accessed [using ssh], and various checks are performed, for example, checking database configuration, whether certain system parameters match required limits, and that the environment is suitable for running Tungsten Clustering.

• During an installation or upgrade, tpm copies the current distribution to each remote host.

• The core configuration file is then used to translate a number of template files within the configuration of each component of the system into the configuration properties files used by Tungsten Clustering. The configuration information is shared on every configured host within the service; this ensures that in the event of a host failure, the configuration can be recovered.

• The components of Tungsten Clustering are then started (installation) or restarted according to the configuration options.

Where possible, these steps are conducted in parallel to speed up the process and limit the interruption to services and operations.

This method of operation ensures:

• Active configurations and properties are not updated until validation is completed. This prevents a running Tungsten Clustering installation from being affected by an incompatible or potentially dangerous change to the configuration.

• Enables changes to be made to the staging configuration before the configuration is deployed.

• Services are not stopped/restarted unnecessarily.

• During an upgrade or update, the time required to reconfigure and restart is kept to a minimum.

Because of this safe approach to performing configuration, downtime is minimized, and the configuration is always based on files that are separate from, and independent of, the live configuration.

Important

tpm always creates the active configuration from the combination of the template files and parameters given to tpm. This means that changes to the underlying property files with the Tungsten Clustering configuration are overwritten by tpm when the service is configured or updated.

In addition to the commands that tpm supports for the installation and configuration, the command also supports a number of other utility and information modes, for example, the fetch command retrieves existing configuration information to your staging, while query returns information about an active configuration.

Using tpm is divided up between the commands that define the operation the command will perform, which are covered in Section 9.5, “tpm Commands”; configuration options, which determine the parameters that configure individual services, which are detailed in Section 9.8, “tpm Configuration Options”; and the options that alter the way tpm operates, covered in Section 9.3, “tpm Staging Configuration”. 
9.1. Comparing Staging and INI tpm Methods

tpm supports two different deployment methodologies. Both configure one or more Tungsten Clustering services, in a safe and secure manner, but differ in the steps and process used to complete the installation. The two methods are:

- **Staging Directory**

  When using the staging directory method, a single configuration that defines all services and hosts within the Tungsten Clustering deployment is created. tpm then communicates with all the hosts you are configuring to install and configure the different services required. This is best when you have a consistent configuration for all hosts and do not have any configuration management tools for your systems.

  Figure 9.1. tpm Staging Based Deployment

- **INI File**

  When using the INI file method, configuration for each service must be made individually using an INI configuration file on each host. This is ideal for deployments where you have a configuration management system (e.g. Puppet and Chef) to manage the INI file. It also works very well for deployments where the configuration for each system is different from the others.
The tpm Deployment Command

Figure 9.2. tpm INI Based Deployment

![Diagram showing Host A, Host B, and Host C with steps for Configure, Deploy, and Installed]

Table 9.1. TPM Deployment Methods

<table>
<thead>
<tr>
<th>Feature</th>
<th>Staging Directory</th>
<th>INI File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy Multiple Services</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Deploy to Multiple Hosts</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Individual Host-based Configuration</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Single-Step Upgrade</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Requires SSH Configuration</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>RPM/PKG Support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Note

Check the output of `tpm query staging` to determine which method your current installation uses. The output for an installation from a staging directory will start with # Installed from tungsten@staging-host:/opt/continuent/software/tungsten-clustering-6.1.3-43. An installation based on an INI file may include this line but the hostname will reference the current host and there will be an /etc/tungsten/tungsten.ini file present.

To install a three-node service using the staging method:

1. Extract Tungsten Clustering on your staging server.
2. On each host:
   a. Complete all the Appendix B, Prerequisites, including setting the ssh keys.
3. Execute the `tpm configure` and `tpm install` commands to configure and deploy the service from the staging server.

To install a three-node service using the INI method:

1. On each host:
   a. Extract Tungsten Clustering.
   b. Complete all the Appendix B, Prerequisites.
9.2. Processing Installs and Upgrades

The tpm command is designed to coordinate the deployment activity across all hosts in a dataservice. This is done by completing a stage on all hosts before moving on. These operations will happen on each host in parallel and tpm will wait for the results to come back before moving on.

- Copy Tungsten Clustering and deployment files to each server
  
  During this stage part of the Tungsten Clustering package is copied to each server. At this point only the tpm command is copied over so we can run validation checks locally on each machine.
  
  The configuration is also transferred to each server and checked for completeness. This will run some commands to make sure that we have all of the settings needed to run a full validation.

- Validate the configuration settings
  
  Each host will validate the configuration based on validation classes. This will do things like check file permissions and database credentials. If errors are found during this stage, they will be summarized and the script will exit.

```
# Validation failed
# Errors for host3
ERROR >> host3 >> Password specified for app@% does not match the running instance on tungsten@host3:13306 (WITH PASSWORD). This may indicate that the user has a password using the old format. (MySQLConnectorPermissionsCheck)
# Errors for host2
ERROR >> host2 >> Password specified for app@% does not match the running instance on tungsten@host2:13306 (WITH PASSWORD). This may indicate that the user has a password using the old format. (MySQLConnectorPermissionsCheck)
# Errors for host1
ERROR >> host1 >> Password specified for app@% does not match the running instance on tungsten@host1:13306 (WITH PASSWORD). This may indicate that the user has a password using the old format. (MySQLConnectorPermissionsCheck)
```

At this point you should verify the configuration settings and retry the tpm install command. Any errors found during this stage may be skipped by running tpm configure alpha --skip-validation-check=MySQLConnectorPermissionsCheck. When re-running the tpm install command this check will be bypassed.

- Deploy Tungsten Clustering and write configuration files
  
  If validation is successful, we will move on to deploying Tungsten Clustering and writing the actual configuration files. The tpm command uses a JSON file that summarizes the configuration. The Tungsten Clustering processes use many different files to store the configuration and tpm is responsible for writing them.

  The /opt/continuent/releases directory will start to collect multiple directories after you have run multiple upgrades. We keep the previous versions of Tungsten Clustering in case a downgrade is needed or for review at a later date. If your upgrade has been successful, you can remove old directories. Make sure you do not remove the directory that is linked to by the /opt/continuent/tungsten symlink.

  **Note**
  
  Do not change Tungsten Clustering configuration files by hand. This will cause future updates to fail. One of the validation checks compares the file that tpm written with the current file. If there are differences, validation will fail.

  This is done to make sure that any configuration changes made by hand are not wiped out without giving you a chance to save them. You can run tpm query modified-files to see what, if any, changes have been made.

- Start Tungsten Clustering services
After Tungsten Clustering is fully configured, the `tpm` command will start services on all of the hosts. This process is slightly different depending on if you are doing a clean install or an upgrade.

- **Install**
  1. Check if `--start` or `--start-and-report` were provided in the configuration
  2. Start the Tungsten Replicator and Tungsten Manager on all hosts
  3. Wait for the Tungsten Manager to become responsive
  4. Start the Tungsten Connector on all hosts

- **Upgrade**
  1. Put all dataservices into `MAINTENANCE` mode
  2. Stop the Tungsten Replicator 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`.

### 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.

When configuring defaults, the defaults affect all configured services, with individual services able to override or set their own parameters.

```shell
tpm configure [service_name|defaults] [tpm options] [service configuration options]
```

In addition to the Section 9.8, "tpm Configuration Options", the common options in Table 9.5, "tpm Common Options" may be given.

The `tpm` command will store the staging configuration in the staging directory that you run it from. This behavior is changed if you have `$CONTINUENT_PROFILES` or `$REPLICATOR_PROFILES` defined in the environment. If present, `tpm` will store the staging configuration in that directory. Doing this will allow you to upgrade to a new version of the software without having to run the `tpm fetch` command.

If you are running Tungsten Clustering, 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`.

#### 9.3.1. Configuring default options for all services

```shell
tools/tpm configure defaults 
   --replication-user=tungsten 
   --replication-password=secret 
   --replication-port=13306
```

These options will apply to all services in the configuration file. This is useful when working with a composite dataservice or multiple independent services. These options may be overridden by calls to `tpm configure service_name` or `tpm configure service_name --hosts`.

#### 9.3.2. Configuring a single service

```shell
tools/tpm configure alpha 
   --master=host1 
   --member=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`. 

388
Relationship of `--members` and `--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.

9.3.3. Configuring a single host

```bash
./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`.

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.

```bash
./tools/tpm reverse
# Defaults for all data services and hosts
tools/tpm configure defaults
  --application-password=secret
  --application-port=3306
  --application-user=app
  --replication-password=secret
  --replication-port=13306
  --replication-user=tungsten
  --start-and-report=true
  --user=tungsten
# Options for the alpha data service
tools/tpm configure alpha
  --connectors=host1,host2,host3
  --master=host1
  --members=host1,host2,host3
```

The output includes all of the `tpm configure` commands necessary to rebuild the configuration. It includes all default, dataservice and host specific configuration settings. Review this output and make changes as needed until you are satisfied.

9.3.5. Installation

After you have prepared the configuration file, it is time to install.

```bash
./tools/tpm install
```

This will install all services defined in configuration. The installation will be done as explained in Section 9.2, “Processing Installs and Upgrades”. This will include the full set of `--members`, `--slaves`, `--master`, and `--connectors`.

9.3.5.1. Installing a set of specific services

```bash
./tools/tpm install alpha,bravo
```

All hosts included in the alpha and bravo services will be installed. The installation will be done as explained in Section 9.2, “Processing Installs and Upgrades”.

9.3.5.2. Installing a set of specific hosts

```bash
./tools/tpm install --hosts=host1,host2
```

Only `host1` and `host2` will be installed. The installation will be done as explained in Section 9.2, “Processing Installs and Upgrades”.

9.3.6. Upgrades from a Staging Directory

This process must be run from the staging directory in order to run properly. Determine where the current software was installed from.

```bash
./tpm query staging
```
The tpm Deployment Command

The tpm Deployment Command

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.

```
tar zxf tungsten-clustering-6.1.3-43.tar.gz
```

```
cd tungsten-clustering-6.1.3-43
```

**Warning**

Before performing and upgrade, please ensure that you have checked the Appendix B, Prerequisites, as software and system requirements may have changed between versions and releases.

Before any update, the current configuration must be known. If the CONTINUENT_PROFILES or REPLICA_PROFILES environment variables were used in the original deployment, these can be set to the directory location where the configuration was stored.

Alternatively, the update can be performed by fetching the existing configuration from the deployed directory by using the tpm fetch command:

```
./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.

```
./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.

```
./tools/tpm configure service_name ...
```

```
./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:

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

The connectors must be updated separately on each server by running:

```
./tpm promote-connector
```

Version 4.0.0. As of software version 4.0.0, the tpm command will use connector graceful-stop 30 followed by connector start when upgrading versions. If that command fails then a regular connector stop is run. This behavior is also applied when using tools/tpm update --replace-release.

### 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.

```
./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.

```
./tools/tpm help update | grep thl-log-retention
```

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.
The tpm Deployment Command

This outputs the hostname and directory where the software was installed from. Make your way to that host and directory before proceeding.

```
shell> tpm query staging
tungsten@staging-host:/opt/continuent/software/continuent-tungsten-2.0.3-519
```

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
```

### 9.3.8. Converting from INI to Staging

If you currently use the INI installation method and wish to convert to using the Staging method, there is currently no easy way to do that. The procedure involves uninstalling fully on each node, then reinstalling from scratch.

If you still wish to convert from the INI installation method to using the Staging method, use the following procedure:

1. Place cluster(s) in Maintenance Mode:
   ```
   shell> cctrl
   cctrl> set policy maintenance
   ```
2. On the staging node, extract the software into `/opt/continuent/software/{extracted_dir}`
   ```
   shell> cd /opt/continuent/software
   shell> tar zxf tungsten-clustering-6.1.3-43.tar.gz
   ```
   **Important**
   If this is a MSMM topology, make sure you extract both the clustering and replication packages.
3. Create the text file `config.sh` based on the output from `tpm reverse`:
   ```
   shell> cd tungsten-clustering-6.1.3-43
   shell> tpm reverse > config.sh
   ```
   Review the new `config.sh` script to confirm everything is correct, making any needed edits. When ready, create the new configuration:
   ```
   shell> sh config.sh
   ```
   **Important**
   If you are using a MSMM topology, repeat these steps using the replicator staging directory. For example:
   ```
   shell> cd tungsten-replicator-6.1.3-43
   shell> /opt/replicator/tungsten/tools/tpm reverse > config.sh
   shell> sh config.sh
   ```
   Review the new configuration:
   ```
   shell> tools/tpm reverse
   ```
The tpm Deployment Command

See Section 9.3, "tpm Staging Configuration" for more information.

4. On all nodes, uninstall the Tungsten software:

   Warning
   Executing this step WILL cause an interruption of service.

   ```
   shell> tpm uninstall --i-am-sure
   ```

   Important
   If you are using a MSMM topology, repeat these steps using the replicator tpm command. For example:

   ```
   shell> /opt/replicator/tungsten/tools/tpm uninstall --i-am-sure
   ```

5. On all nodes, rename the `tungsten.ini` file:

   ```
   shell> mv /etc/tungsten/tungsten.ini /etc/tungsten/tungsten.ini.old
   ```

6. On the staging node only, change to the extracted directory and execute the `tpm install` command:

   ```
   shell> cd /opt/continuent/software/tungsten-clustering-6.1.3-43
   shell> ./tools/tpm install
   ```

   Important
   If you are using a MSMM topology, repeat the install using the replicator staging directory. For example:

   ```
   shell> cd /opt/continuent/software/tungsten-replicator-6.1.3-43
   shell> ./tools/tpm install
   ```

7. Once all steps have been completed and the cluster(s) are stable, take each cluster out of maintenance mode by setting the policy back to automatic:

   ```
   shell> cctrl
   cctrl> set policy automatic
   ```

9.4. tpm INI File Configuration

tpm can use an INI file to manage host configuration. This is a fundamental difference from the normal model for using tpm. When using an INI configuration, the tpm command will only work with the local server.

In order to configure Tungsten on your server using an INI file you must still complete all of the Appendix B, Prerequisites. Copying SSH keys between your servers is optional but setting them up makes sure that certain scripts packaged with Continuent Tungsten will still work.

9.4.1. Creating an INI file

When using an INI configuration, installation and updates will still be done using the tpm command. Instead of providing configuration information on the command line, the tpm command will look for an INI file in three files:

1. $HOME/tungsten.ini
2. /etc/tungsten/tungsten.ini
3. /etc/tungsten.ini

In Tungsten Replicator 3.0 and later, tpm will automatically search all `tungsten*.ini` files within the /etc/tungsten directory. An alternative directory can be searched using `--ini [407]` option to tpm.

The INI file(s) must be readable by the tungsten system user.

Here is an example of a `tungsten.ini` file that would setup a simple dataservice.

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

[alpha]
master=host1
members=host1,host2,host3
connectors=host1,host2,host3

[defaults]
```
The tpm Deployment Command

The property names in the INI file are the same as what is used on the command line. Simply remove the leading -- characters and add it to the proper section. Each section in the INI file replaces a single tpm configure call. The section name inside of the square brackets is used as the service name. In the case of the [defaults] section, this will act like the tpm configure defaults command.

Include any host-specific options in the appropriate section. This configuration will only apply to the local server, so there is no need to put host-specific settings in a different section.

9.4.2. Installation with INI File

Once you have created the tungsten.ini file, the tpm command will recognize it and use it for configuration. Unpack the software into /opt/continuent/software and run the tpm install command.

```
shell> cd /opt/continuent/software/tungsten-clustering-6.1.3-43
shell> ./tools/tpm install
```

The tpm command will read the tungsten.ini file and setup all dataservices on the current server.

9.4.3. Upgrades with an INI File

Use the tpm update command to upgrade to the latest version.

```
shell> cd /opt/continuent/software
shell> tar xzf tungsten-clustering-6.1.3-43.tar.gz
shell> cd tungsten-clustering-6.1.3-43
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:

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

The connectors must be updated separately on each server by running:

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

Version 4.0.0. As of software version 4.0.0, the tpm command will use connector graceful-stop 30 followed by connector start [304] when upgrading versions. If that command fails then a regular connector stop [304] is run. This behavior is also applied when using tools/tpm update --replace-release.

During the lifetime of the cluster, switches may happen and the current master may well be a different node than what is reflected in the static ini file in the master= line. Normally, this difference is ignored during and update or an upgrade.

However, if a customer has some kind of procedure (i.e. automation) which hand-edits the ini configuration file master= line at some point, and such hand-edits do not reflect the current reality at the time of the update/upgrade, an update/upgrade will fail and the cluster may be left in an indeterminate state.

### Warning

The best practice is to NOT change the master= line in the INI configuration file after installation.

There is still a window of opportunity for failure. The update will continue, passing the CurrentTopologyCheck test and potentially leaving the cluster in an indeterminate state if the master= option is set to a hostname that is not the current master or the current host.

9.4.4. Configuration Changes with an INI file

The tpm update also allows you to apply any configuration changes. Start by making any necessary changes to the tungsten.ini file. Then proceed to running tpm update.
The `tpm` Deployment Command

```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
```

### 9.4.5. Converting from Staging to INI

If you currently use the Staging installation method and wish to convert to using INI files, use the following procedure.

You can also try using the script in Section 9.4.6, "Using the `translatetoini.pl` Script".

1. Place cluster(s) in Maintenance Mode:

```bash
shell> cctrl
cctrl> set policy maintenance
```

2. Create the text file `/etc/tungsten/tungsten.ini` on each node. They will normally all be the same.

```bash
shell> sudo mkdir /etc/tungsten
shell> sudo chown -R tungsten: /etc/tungsten
shell> chmod 700 /etc/tungsten
shell> touch /etc/tungsten/tungsten.ini
shell> chmod 600 /etc/tungsten/tungsten.ini
```

Each section in the INI file replaces a single `tpm configure` call. The section name inside of [square brackets] is used as the service name. In the case of the [defaults] section, this will act like the `tpm configure defaults` command. The property names in the INI file are the same as what is used on the command line. Simply remove the leading -- characters and add it to the proper section.

For example, to seed the `tungsten.ini` file, use the output of `tpm reverse`:

```bash
shell> tpm reverse > /etc/tungsten/tungsten.ini
```

Edit the new ini file and clean it up as per the rules above. For example, using vim:

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

```bash
:s/\s+\$/g
:s/\s*\)$/g
:%s/tools/^[tpm configure]/[(]/g
:%s/^--//g
```

**Important**

In the above example, you MUST manually add the trailing square bracket ] to the end of the defaults tag and to the end of every service name section. Just search for the opening square bracket [ and make sure there is a matching closing square bracket for every one.

See Section 9.4.1, "Creating an INI file" for more information.

3. On every node, extract the software into `/opt/continuent/software/{extracted_dir}`

**Warning**

Make sure you have the same release that is currently installed.

```bash
shell> cd /opt/continuent/software
shell> tar zxf tungsten-clustering-6.1.3-43.tar.gz
```

**Important**

If this is a MSMM topology, make sure you extract both the clustering and replication packages.

4. On each node, change to the extracted directory and execute the `tpm` command:
The tpm Deployment Command

Warning

Execute this step on the slaves first, then switch the master - this procedure will restart the Tungsten services so switch your master to avoid interruption of service. See Section 5.15.3, “Performing Maintenance on an Entire Dataservice” for more information.

```
shell> cd /opt/continuent/software/tungsten-clustering-6.1.3-43
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:

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

The connectors must be updated separately on each server by running:

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

5. If you have a MSMM topology, you must also update the cross-site replicators:

On each node, change to the extracted replicator directory and execute the `tpm` command:

```
shell> cd /opt/continuent/software/tungsten-replicator-6.1.3-43
shell> ./tools/tpm update
```

6. Once all steps have been completed and the cluster(s) are stable, take each cluster out of maintenance mode by setting the policy back to automatic:

```
shell> cctrl
shell> set policy automatic
```

9.4.6. Using the `translatetoini.pl` Script

You can download a script from the documentation library, `translatetoini.pl`. You must have a copy of Perl installed to be able to execute the script.

To use the script, you can either run the script and paste in the staging output, or pipe the output from `tpm reverse` directly into the script. When supplying the staging output, you should supply the output from the within the configured staging directory. For example:

```
shell> ./tools/tpm reverse|../translatetoini.pl
```

The script will create the file `tungsten.ini` in the current directory containing the converted output.

To change the destination, use the `--filename` option:

```
shell> ./tools/tpm reverse|../translatetoini.pl --filename=t.ini
```

You can also combine multiple staging configurations into a single INI conversion by appending to an existing INI file by adding the `--append` option:

```
shell> ./tools/tpm reverse|../translatetoini.pl --append
```

You should always check the INI file before using it for a live installation to ensure that all of the options and parameters have been identified and configured properly.

A training video is available on how to perform the staging to INI file conversion using the `translatetoini.pl` script:

Click here for a video of the INI conversion procedure, showing the full process from start to finish...

9.5. tpm Commands

All calls to `tpm` will follow a similar structure, made up of the command, which defines the type of operation, and one or more options.

```
shell> tpm command [sub command] [tpm options] [command options]
```

The command options will vary for each command. The core `tpm` options are:
The **tpm** Deployment Command

**Table 9.2. tpm Core Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--force [396], -f [396]</td>
<td>Do not display confirmation prompts or stop the configure process for errors</td>
</tr>
<tr>
<td>--help [396], -h [396]</td>
<td>Displays help message</td>
</tr>
<tr>
<td>--info [396], -i [396]</td>
<td>Display info, notice, warning and error messages</td>
</tr>
<tr>
<td>--notice [396], -n [396]</td>
<td>Display notice, warning and error messages</td>
</tr>
<tr>
<td>--preview [396], -p [396]</td>
<td>Displays the help message and preview the effect of the command line options</td>
</tr>
<tr>
<td>--profile file [396]</td>
<td>Sets name of config file</td>
</tr>
<tr>
<td>--quiet [396], -q [396]</td>
<td>Only display warning and error messages</td>
</tr>
<tr>
<td>--verbose [396], -v [396]</td>
<td>Display debug, info, notice, warning and error messages</td>
</tr>
</tbody>
</table>

- **--force [396]**
  Forces the deployment process to complete even if there are warning or error messages that would normally cause the process to fail. Forcing the installation also ignores all confirmation prompts during installation and always attempts to complete the process.

- **--help [396]**
  Displays the help message for **tpm** showing the current options, commands and version information.

- **--info [396]**
  Changes the reporting level to include information, notice, warning and error messages. Information level messages include annotations of the current process and stage in the deployment, such as configuration or generating files and configurations. This shows slightly more information than the default, but less than the full debug level offered by **--verbose [396]**.

- **--notice [396]**
  Sets the output level to include notice, warning, and error messages. Notice level messages include information about further steps or actions that should be taken, or things that should be noted without indicating a failure or error with the configuration options select.

- **--preview [396]**

- **--profile file [396]**
  Specify the name of the configuration file to be used. This can be useful if you are performing multiple configurations or deployments from the same staging directory. The entire configuration and deployment information is stored in the file before installation is started. By specifying a different file you can have multiple deployments and configurations without requiring separate staging directories.

- **--quiet [396]**
  Changes the error reporting level so that only warning and error messages are displayed. This mode can be useful in automated deployments as it provides output only when a warning or error exists. All other messages, including informational ones, are suppressed.

- **--verbose [396]**
  Displays a much more detailed output of the status and progress of the deployment. In verbose mode, **tpm** annotates the entire process describing both what it is doing and all debug, warning and other messages in the output.

The **tpm** utility handles operations across all hosts in the dataservice. This is true for simple and composite dataservices as well as complex multi-master replication services. The coordination requires SSH connections between the hosts according to the Appendix B, Prerequisites. There are two exceptions for this:

1. When the **--hosts [455]** 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 [460]**, **--slaves [472]**, **--master [459]**, **--connectors [444]** arguments should be used when calling **--hosts [455]**.

2. When you are using an INI configuration file (see Section 9.4, “tpm INI File Configuration”) all calls to **tpm** will only affect the current host.

The installation process starts in a staging directory. This is different from the installation directory where Tungsten Clustering will ultimately be placed but may be a sub-directory. In most cases we will install to **/opt/continuent** but use **/opt/continuent/software** as a staging directory. The release package should be unpacked in the staging directory before proceeding. See the Section B.1, “Staging Host Configuration” for instructions on selecting a staging directory.

---

396
### Table 9.3. `tpm` Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>configure</td>
<td>Configure a data service within the global configuration</td>
</tr>
<tr>
<td>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</td>
<td>Make a previously configured and prepared directory and make it the active</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>ssh-copy-cert</td>
<td>Executes the commands required to generate authorized SSH certificate and keys required for tpm</td>
</tr>
<tr>
<td>uninstall</td>
<td>Uninstall software from host(s)</td>
</tr>
<tr>
<td>update</td>
<td>Update an existing configuration or software version</td>
</tr>
<tr>
<td>validate</td>
<td>Validate the current configuration</td>
</tr>
<tr>
<td>validate-update</td>
<td>Validate the current configuration and update</td>
</tr>
</tbody>
</table>

### 9.5.1. `tpm configure` Command

The `configure` command to `tpm` creates a configuration file within the current profiles directory.

### 9.5.2. `tpm connector` Command

This will open a MySQL CLI connection to the local Tungsten Connector using the current values for `--application-user` [437], `--application-password` [437] and `--application-port` [437].

```bash
shell> tpm connector
```

**Warning**

This command will fail if the `mysql` utility is not available or if the local server does not have a running Tungsten Connector.

**Important**

The MySQL 5.7 command-line client will now attempt to connect via SSL by default, which will fail on the Connector unless it is configured for SSL operations. You may add the `--skip-ssl` option to bypass this issue. See Section 2.7.13, “Configuring Connector SSL” for more information about using SSL with the Connector.

#### 9.5.2.1. `tpm connector --hosts` Command

Limits the connection to the list of specified hosts. For example:

```bash
shell> tpm connector --hosts host1,host2
```

Would limit the connection to the connector on one of the specified hosts. The hostname must be specified in the same form as it is in the configuration.

#### 9.5.2.2. `tpm connector --dataservice-name` Command

Limit the command to the hosts in the specified dataservice. Multiple dataservices can be specified by providing each dataservice separated by a comma.
9.5.2.3. **tpm connector --dataservice-name**

Provides sample configuration information for various common development environments:

```bash
shell> tpm connector --samples
```

- **Bash**
  ```
  mysql -hdemo-c11 -P3306 -uapp_user -ppassword
  ```

- **Perl::dbi**
  ```
  $dbh=DBI->connecti('DBI:mysql:host=demo-c11;port=3306', 'app_user', 'password')
  ```

- **PHP::mysqli**
  ```
  $dbh = new mysqli('demo-c11', 'app_user', 'password', 'schema', '3306');
  ```

- **PHP::pdo**
  ```
  $dbh = new PDO('mysql:host=demo-c11;port=3306', 'app_user', 'password');
  ```

- **Python::mysql.connector**
  ```
  dbh = mysql.connector.connect(user='app_user', password='password', host='demo-c11', port=3306, database='schema')
  ```

- **Java::DriverManager**
  ```
  dbh=DriverManager.getConnection("jdbc:mysql://demo-c11:3306/schema", "app_user", "password")
  ```

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
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`.

5.3.7 and 6.0.5 Behavior Change. As of versions 5.3.7 and 6.0.5, the `tpm diag` command behavior has been modified.

- With Staging-method deployments, the `tpm diag` command continues to behave as before:
  - The `tpm diag` command alone will obtain diagnostics from all hosts in the cluster.
  - The `tpm diag --hosts host1,host2,hostN` command will obtain diagnostics from the specified host(s) only.

- With INI-method deployments, the new behavior is as follows:
  - The `tpm diag` command alone will obtain diagnostics from the local host only.
  - The `tpm diag --hosts host1,host2,hostN` command will obtain diagnostics from the specified host(s) only.

**Warning**

Limitation: the host list MUST include the local hostname or the command will fail.

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:

```
Archive: tungsten-diag-2014-07-08-11-24-01.zip
Length Date Time Name
------- --- ----- ----
 0 2014-07-08 11:24 tungsten-diag-2014-07-08-11-24-01/
  0 2014-07-08 11:24 tungsten-diag-2014-07-08-11-24-01/ct_ssl1/
 1411 2014-07-08 11:24 tungsten-diag-2014-07-08-11-24-01/ctl_ssl1/trepctl.txt
  0 2014-07-08 11:24 tungsten-diag-2014-07-08-11-24-01/ctl_ssl2/
  0 2014-07-08 11:24 tungsten-diag-2014-07-08-11-24-01/ctl_ssl3/
```
9.5.4. tpm fetch Command

There are some cases where you would like to review the configuration or make changes prior to the upgrade. In these cases it is possible to fetch the configuration and process the upgrade as different steps.

```shell
dir=tools/tpm fetch \
--directory=/opt/continuent \
--hosts=host1,autodetect
```

This will load the configuration into the local staging directory. You can then make changes using `tpm configure` before pushing out the upgrade.

The `tpm fetch` command supports the following arguments:

- **--hosts [455]**
  
  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 [477]**
  
  The username to be used when logging in to other hosts.

- **--directory [451]**
  
  The installation directory of the current Tungsten Clustering installation. If `autodetect` is specified, then `tpm` will look for the installation directory by checking any running Tungsten Clustering processes.

9.5.5. tpm firewall Command

The `tpm firewall` command displays port information required to configured a firewall. When used, the information shown is for the current host:

```shell
tpm firewall
```

To host1
---------------------------------------------------------------------------------
| From application servers | 9999 |
| From connector servers   | 11999, 12000, 13306 |
| From database servers    | 2112, 7800, 8896, 9997, 10999, 11999, 12000, 13306 |
---------------------------------------------------------------------------------

The information shows which ports, on which hosts, should be opened to enable communication.

9.5.6. tpm help Command

The `tpm help` command outputs the help information for `tpm` showing the list of supported commands and options.

```shell
tpm help
```

Usage: tpm help [commands,config-file,template-file] [general-options] [command-options]

General options:

- `-f`, `--force`  Do not display confirmation prompts or stop the configure process for errors
- `-h`, `--help`  Displays help message
- `-p`, `--profile file`  Sets name of config file (default: tungsten.cfg)
- `-s`, `--preview`  Displays the help message and preview the effect of the command line options
- `-q`, `--quiet`  Only display warning and error messages
- `-n`, `--notice`  Display info, notice, warning and error messages
- `-v`, `--verbose`  Display debug, info, notice, warning and error messages

To get a list of available configuration options, use the `config-file` subcommand:

```shell
tpm help config-file
```

...
The tpm Deployment Command

# Config File Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--automatic, --auto, -a</td>
<td>Use the --automatic option to set the cluster policy to AUTOMATIC mode. You may also use -a and --auto.</td>
</tr>
<tr>
<td>--display, -d</td>
<td>Use the --display option to display the current cluster policy. Please note that tpm policy with no arguments will also display the current policy.</td>
</tr>
<tr>
<td>--get, -g</td>
<td>Use the --get option to display the current cluster policy. Please note that tpm policy with no arguments will also display the current policy.</td>
</tr>
<tr>
<td>--list, -l</td>
<td>Use the --list option to display the current cluster policy. Please note that tpm policy with no arguments will also display the current policy.</td>
</tr>
</tbody>
</table>

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
```

Configures the service first, then performs the installation steps.

During installation, tpm checks for any host configuration problems and issues, copies the Tungsten Clustering software to each machine, creates the necessary configuration files, and if requests, starts and reports the status of the service.

If any of these steps fail, changes are backed out and installation is stopped.

9.5.8. tpm mysql Command

This will open a MySQL CLI connection to the local MySQL server using the current values for --replication-user [470], --replication-password [469] and --replication-port [469].

```
shell> ./tools/tpm mysql
```

This command will fail if the mysql utility is not available or if the local server does not have a running database server.

9.5.9. tpm policy Command

The tpm policy command displays and optionally sets the cluster policy.

Table 9.4. tpm policy Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--automatic, --auto, -a</td>
<td>Use the --automatic option to set the cluster policy to AUTOMATIC mode. You may also use -a and --auto.</td>
</tr>
<tr>
<td>--display, -d</td>
<td>Use the --display option to display the current cluster policy. Please note that tpm policy with no arguments will also display the current policy.</td>
</tr>
<tr>
<td>--get, -g</td>
<td>Use the --get option to display the current cluster policy. Please note that tpm policy with no arguments will also display the current policy.</td>
</tr>
<tr>
<td>--list, -l</td>
<td>Use the --list option to display the current cluster policy. Please note that tpm policy with no arguments will also display the current policy.</td>
</tr>
</tbody>
</table>
The **tpm** Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--maintenance, --maint, -m</td>
<td>Use the --maintenance option to set the cluster policy to MAINTENANCE mode. You may also use -m and --maint.</td>
</tr>
<tr>
<td>--print, -p [396]</td>
<td>Use the --print option to display the current cluster policy. Please note that tpm policy with no arguments will also display the current policy.</td>
</tr>
<tr>
<td>--show, -s</td>
<td>Use the --show option to display the current cluster policy. Please note that tpm policy with no arguments will also display the current policy.</td>
</tr>
</tbody>
</table>

The **tpm policy** command was designed as a replacement for the cctrl set policy command to make it easier to set the cluster policy on the command line.

### 9.5.10. 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
```

Version 4.0.0. As of software version 4.0.0, the tpm command will use connector graceful-stop 30 followed by connector start [304] when upgrading versions. If that command fails then a regular connector stop [304] is run. This behavior is also applied when using tools/tpm update --replace-release.

### 9.5.11. tpm query Command

The **query** command provides information about the current tpm installation. There are a number of subcommands to query specific information:

- tpm query config — return the full configuration values
- tpm query dataservices — return the list of dataservices
- tpm query default — return the list of configured default values
- tpm query deployments — return the configuration of all deployed hosts
- tpm query manifest — get the manifest information
- tpm query modified-files — return the list of files modified since installation by tpm
- tpm query staging — return the staging directory from where Tungsten Clustering was installed
- tpm query topology — return the current topology
- tpm query usermap — return the list of users organized 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

#### 9.5.11.1. tpm query config

Returns a list of all of the configuration values, both user-specified and implied within the current configuration. The information is returned in the form a JSON value:

```
shell> tpm query config
```
The tpm Deployment Command

```json
{"__system_defaults_will_be_overwritten__": {
  ...
  "staging_directory": "/home/tungsten/tungsten-clustering-6.1.3-43",
  "staging_host": "tr-ms1",
  "staging_user": "tungsten"
}
```

9.5.11.2. tpm query dataservices

Returns the list of configured dataservices that have, or will be, installed:

```shell
shell> tpm query dataservices
alpha : PHYSICAL
```

9.5.11.3. 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
shell> tpm query deployments
{
  "config_target_basename": "tungsten-clustering-6.1.3-43_pid22729",
  "dataservice_host_options": {
    "alpha": {
      "start": "true"
    }
    ...
    "staging_directory": "/home/tungsten/tungsten-clustering-6.1.3-43",
    "staging_host": "tr-ms1",
    "staging_user": "tungsten"
  }
```

9.5.11.4. tpm query manifest

Returns the manifest information for the identified release of Tungsten Clustering, including the build, source and component versions, returned in the form of a JSON value:

```shell
shell> tpm query manifest
{
  "SVN": {
    "bristlecone": {
      "URL": "https://bristlecone.googlecode.com/svn/trunk/bristlecone",
      "revision": 179
    },
    "connector": {
      "URL": "svn+ssh://svn.continuent.com/svnroot/tungsten/trunk/connector",
      "revision": 9150
    },
    "fsn": {
      "URL": "svn+ssh://svn.continuent.com/svnroot/tungsten/trunk/fsn",
      "revision": 9150
    },
    "manager": {
      "URL": "svn+ssh://svn.continuent.com/svnroot/tungsten/trunk/manager",
      "revision": 9150
    },
    "replicator": {
      "URL": "https://tungsten-replicator.googlecode.com/svn/trunk",
      "revision": 2020
    }
  },
  "date": "Fri Jan 24 22:25:07 UTC 2014",
  "host": "ip-10-250-35-16",
  "hudson": {
    "SVNRevision": null,
    "URL": "http://cc.aws.continuent.com/",
    "buildID": 227,
    "buildNumber": 227,
    "buildFlag": "jenkins-Base_Continuent_Tungsten_2.8.2_XUnit-227",
    "jobName": "Base_Continuent_Tungsten_2.8.2_XUnit"
  },
  "product": "Continuent Tungsten",
  "userAccount": "jenkins",
  "version": {
    "major": 2,
    "minor": 0,
    "revision": 2
  }
}
```
9.5.11.5. 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 to identify which files contain changes so that these modifications can be manually migrated to the new installation. To restore or replace files with their original installation, copy the `.filename.orig` file.

9.5.11.6. tpm query staging

Returns the host and directory from which the current installation was created:

```
shell> tpm query staging
tungsten@host1:/home/tungsten/tungsten-clustering-6.1.3-43
```

This can be useful when the installation host and directory from which the original configuration was made need to be updated or modified.

9.5.11.7. 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"
}
```

9.5.11.8. 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
```

9.5.11.9. tpm query version

Returns the version for the identified version of Tungsten Clustering:

```
shell> tpm query version
6.1.3-43
```

9.5.12. tpm reset Command

This command will clear the current state for all Tungsten services:

- Management metadata
- Replication metadata
- THL files
- Relay log files
- Replication position

If you run the command from an installed directory, it will only apply to the current server. If you run it from a staging directory, it will apply to all servers unless you specify the `--hosts [455]` option.

```
shell> ./tools/tpm reset
```

9.5.13. tpm reset-thl Command

This command will clear the current replication state for the Tungsten Replicator:

- THL files
The \textbf{tpm Deployment Command}

- 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{lstlisting}[language=bash]
shells> ./tools/tpm reset-thl
\end{lstlisting}

### 9.5.14. tpm reverse Command

The \textbf{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{lstlisting}[language=bash]
shells> ./tools/tpm reverse
# Defaults for all data services and hosts
tools/tpm configure defaults \\
\hspace{1em}--application-password=secret \\
\hspace{1em}--application-port=3306 \\
\hspace{1em}--application-user=app \\
\hspace{1em}--replication-password=secret \\
\hspace{1em}--replication-port=13306 \\
\hspace{1em}--replication-user=tungsten \\
\hspace{1em}--start-and-report=true \\
\hspace{1em}--user=tungsten \\
# Options for the alpha data service
tools/tpm configure alpha \\
\hspace{1em}--connectors=host1,host2,host3 \\
\hspace{1em}--master=host1 \\
\hspace{1em}--members=host1,host2,host3
\end{lstlisting}

The \textbf{tpm reverse} command supports the following arguments:

- \texttt{--public}
  - Hide passwords in the command output
- \texttt{--ini-format}
  - Display output in ini format for use in \texttt{/etc/tungsten/tungsten.ini} and similar configuration files

### 9.5.15. tpm ssh-copy-cert Command

The \textbf{tpm ssh-copy-cert} command executes all the required commands to generate the required \texttt{ssh} certificates required for SSH operation by \texttt{tpm}. Executing the command should generate the required directory, certificate and add that information to the required \texttt{SSH} files, then ensure that the directory permissions and ownership on \texttt{-/ssh} are set correctly.

For example, executing the command outputs the stages and progress:

\begin{lstlisting}[language=bash]
shells> ./tools/tpm ssh-copy-cert
mkdir -p ~/.ssh
echo "-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEmZ9xPn/FfYPFR12k5KvT2/YZcKqQ9FiYn10bRRc6m7YBVDc5gFtsN
psZkz6t1sZVqHHHJh3tJ0v3G2NN2oYB2oN5YJPGX8ZbWm3b8m2s0ycSjKvSDva7v8nN
AlcEqDDY9KsbkW8M0kFyX6QoGZy9j3pLj4uAww6NHflF1kKt7fsiLjAqFzeh7QV
0RZJ2z117FBoz2N2xI4qFyBbkrBb7BvZBZfZAgwCZB3yfjyfjwOwACPz3fl7sFpkf
1ZcQ9Q9m366rLT3T7YV91JvMCvCJv6XQ5vFyFf6bW7K6Qg6V0A6GwJ7kYzUV681t
weFOErjN3EgZ7gbIA1xH2s2JFrcnozBqUNBNQ3D4Q38xO1BhK+rJbQHmpn+6R
EqJOv9Ba3cSAD2J1uX4mKg8Pcri2j[vhsrVU7CY25X130pL0bBxqIQ]yuLaqV7p
4Mjg3jn[8b3cxa2s20t1f366lM61cv1V2r3f4Mwo3ID13TrDyDq6cLJyt7lZb9
ALqVhMtL3j8e3MHI01v1r+jCv1n1Nj43j2CycQYfB1g2zVoVN107e?+Yzk8hR3
414e4tqJgY096M/kB4s1U6jPf4G0X9299u4v4LrLbJPtY009jGDI5F6wdMrNMCrKz
ygg9Qe6kY4k2d3l9995l/JoLu1337k7FGDg9yvQnhfA121wg0QpPtjC1Oz3Q9wckG
v9HsFG90v+QVlQcVf3h38hJ3O1XAmz22cC3Gf2Gh10kKmGn4N4JcaWbB4/BdRd
C5Gv2xXb2vKsMosL3ud7wvXfizgCy8mTwgR6S5T1CkCHONhVvNM03O82F7sCfGwZ5
Fm72bXi8P4kZGKzCrotdxe3RoU3vJ6JW2K0SpDg1Fq510NeQOcYcAYkX3hw
1Kz2A6wD3jG2yXVwG9h58K9RO65f5f56g65g61b7G0LufwP3t6N66n4fxt4N44z
66kLzXv3S5Kt71tXei4q8q3132gCc4cppx50a92H90sOlsqI1Irn1tgsS5O
X92yZg64k665/cKYBcr32xqXMKRqYr37BC1V1c5CyEA8btym91v1GqVQ80vU0I
11L3Xh1F4KF38v1zR563yj2aB0K219CJlT2Dcy10ct0y0q3Zlp37A600r7rHden
UPY1wA2XoYlT0g120kYpFqWNX8b7408C51D7pZpOQ21yv5eJsl3I6/g29nab
sqPqGP3916EvopypjpsJa3KcBqyXwx8s9Frcs1jkryJmKJ1hK47c1s8XF3y8xsp6Qou
wEc5S6wom27x1kA9mO3D131v1B/4F3Y2PS6c5eGhGK6hoyAmsImPQ/fakliLyT
2y/6Z+hDNpWVXxhZ2A31jA3h9G772JveY8FmPOPELChfR1f4UX84h0guy
7ah2QfOCrGcpG/PlMo3F7fawdV66y11b+1q/6X4adnL3bNc7Q3jg3rPMf1vF
2Z/cjP97Wd8MLX2eEOYgXgypM1zF+FPNPS38rdbhc3J2314tCc1fS1CAjlk3pwwm
Zn0JpAcdC/SCfHf62v5B68fs173eBskFkAp490AkA3dU00CRKd
-----END RSA PRIVATE KEY-----" > ~/.ssh/id_rsa
\end{lstlisting}
### 9.5.16. tpm uninstall Command

The **tpm uninstall** command is used to remove the installation.

**Warning**

The uninstall command must be used with care. This is a destructive command and irreversible.

To uninstall the software, you need to issue the following command from the installed software staging directory on every host for INI installs, or from the staging host only for Staging Installs. Running the command on the staging hosts installed via the staging method, will cascade through all nodes in the topology.

```
shell> ./tools/tpm uninstall --i-am-sure
```

### 9.5.17. 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 9.2, “Processing Installs and Upgrades”. The behavior of **tpm update** is dependent on two factors.

1. **Are you upgrading to a new version or applying configuration changes to the current version?**
2. **The installation method used during deployment.**

**Note**

Check the output of **tpm query staging** to determine which method your current installation uses. The output for an installation from a staging directory will start with a `# Installed from tungsten@staging-host:/opt/continuent/software/tungsten-clustering-6.1.3-43`. An installation based on an INI file may include this line but there will be an `/etc/tungsten/tungsten.ini` file on each node.

**Upgrading to a new version**

If a staging directory was used; see Section 9.3.6, “Upgrades from a Staging Directory”.

If an INI file was used; see Section 9.4.3, “Upgrades with an INI File”

**Applying configuration changes to the current version**

If a staging directory was used; see Section 9.3.7, “Configuration Changes from a Staging Directory”.

If an INI file was used; see Section 9.4.4, “Configuration Changes with an INI file”.

**Special Considerations for the Connector**

As of software version 4.0.0, the **tpm** command will use `connector graceful-stop 30` followed by `connector start` when upgrading versions. If that command fails then a regular `connector stop` is run.

This behavior is also applied when using tools/tpm update --replace-release.

The **tpm** command will use `connector reconfigure` when changing connector settings without a version upgrade.

The use of `connector reconfigure` is disabled for the following:

- `--application-port`
- `--application-readonly-port`
- `--router-gateway-port`
- `--router-jmx-port`
- `--conn-java-mem-size`

If `connector reconfigure` can’t be used, `connector graceful-stop 30` and `connector start` are used.

### 9.5.18. 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.
The tpm Deployment Command

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 > 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 > 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 > MyISAM tables exist within this instance - These tables are not crash safe and may lead to data loss in a failover. (MySQLMyISAMCheck)
```

Any problems noted should be addressed before you perform the update using `tpm update`.

9.6. tpm Common Options

`tpm` accepts these options along with those in Section 9.8, "tpm Configuration Options".

- On the command-line, using a double-dash prefix, i.e. `--skip-validation-check=MySQLConnectorPermissionsCheck`
- In an INI file, without the double-dash prefix, i.e. `skip-validation-check=MySQLConnectorPermissionsCheck`

Table 9.5. `tpm` Common Options

<table>
<thead>
<tr>
<th>CommandLine Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--enable-validation-check [406]</td>
<td>enable-validation-check [406]</td>
<td>Enable a specific validation check, overriding any configured skipped checks</td>
</tr>
<tr>
<td>--enable-validation-warnings [407]</td>
<td>enable-validation-warnings [407]</td>
<td>Enable a specific validation warning, overriding any configured skipped warning</td>
</tr>
<tr>
<td>--ini [407]</td>
<td>ini [407]</td>
<td>Specify the location of the directory where INI files will be located</td>
</tr>
<tr>
<td>--net-ssh-option [407]</td>
<td>net-ssh-option [407]</td>
<td>Set the Net:SSH option for remote system calls</td>
</tr>
<tr>
<td>--remove-property [408]</td>
<td>remove-property [408]</td>
<td>Remove the setting for a previously configured property</td>
</tr>
</tbody>
</table>

---
The `tpm` Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--enable-validation-check</code></td>
<td>Enable a specific validation check, overriding any configured skipped checks</td>
<td>string</td>
</tr>
<tr>
<td><code>--enable-validation-warnings</code></td>
<td>Enable a specific validation warning, overriding any configured skipped warning</td>
<td>string</td>
</tr>
<tr>
<td><code>--ini</code></td>
<td>Specify the location of the directory where INI files will be located</td>
<td>string</td>
</tr>
<tr>
<td><code>--net-ssh-option</code></td>
<td>Set the Net::SSH option for remote system calls</td>
<td>string</td>
</tr>
<tr>
<td><code>--property</code></td>
<td>Modify specific property values for the key in any file that the configure script touches.</td>
<td>string</td>
</tr>
</tbody>
</table>

The `--enable-validation-check` will specifically enable a given validation check if the check had previously been set it be ignored in a previous invocation of the configuration through `tpm`. If a check fails, installation is canceled.

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

The `--enable-validation-warnings` will specifically enable a given validation warning check if the check had previously been set it be ignored in a previous invocation of the configuration through `tpm`.

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

The `--ini` specifies an alternative location for the INI files from the default.

The `--net-ssh-option` enables you to set a specific Net::SSH option. For example:

```
shell> tpm update ... --net-ssh-option=compression=zlib
```

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

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

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

- `key=/match/replace/`
The tpm Deployment Command

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

\texttt{--remove-property [408]}

<table>
<thead>
<tr>
<th>Option</th>
<th>--remove-property [408]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>remove-property [408]</td>
</tr>
<tr>
<td>Description</td>
<td>Remove the setting for a previously configured property</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

Remove a previous explicit property setting. For example:

\texttt{shell> tpm configure --remove-property=replicator.filter.pkey.addPkeyToInserts}

\texttt{--skip-validation-check [408]}

<table>
<thead>
<tr>
<th>Option</th>
<th>--skip-validation-check [408]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>skip-validation-check [408]</td>
</tr>
<tr>
<td>Description</td>
<td>Do not run the specified validation check.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

The \texttt{--skip-validation-check [408]} disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

Warning

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

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

\texttt{...}
\texttt{ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)}
\texttt{...}

The check in this case is \texttt{MySQLDefaultTableTypeCheck [418]}, and could be ignored using \texttt{--skip-validation-check=MySQLDefaultTableTypeCheck [408]}. Setting both \texttt{--skip-validation-check [408]} and \texttt{--enable-validation-check [406]} is equivalent to explicitly disabling the specified check.

\texttt{--skip-validation-warnings [408]}

<table>
<thead>
<tr>
<th>Option</th>
<th>--skip-validation-warnings [408]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>skip-validation-warnings [408]</td>
</tr>
<tr>
<td>Description</td>
<td>Do not display warnings for the specified validation check.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

The \texttt{--skip-validation-warnings [408]} disables a given validation check.

You can identify a given check by examining the warnings generated during configuration. For example, the Linux swappiness warning:

\texttt{...}
\texttt{WARN >> centos >> Linux swappiness is currently set to 60, on restart it will be 60, consider setting this to 10 or under to avoid swapping. (SwappinessCheck)}
\texttt{...}

The check in this case is \texttt{MySQLDefaultTableTypeCheck [418]}, and could be ignored using \texttt{--skip-validation-warnings=SwappinessCheck [408]}. Setting both \texttt{--skip-validation-warnings [408]} and \texttt{--enable-validation-warnings [407]} is equivalent to explicitly disabling the specified warning.

9.7. tpm Validation Checks

During configuration and installation, tpm runs a number of configuration, operating system, datasource, and other validation checks to ensure that the correct environment, prerequisites and other settings will produce a valid, working, configuration.

All relevant checks are executed automatically unless specifically ignored (warnings) or disabled (checks) using the corresponding \texttt{--skip-validation-warnings [408]} or \texttt{--skip-validation-check [408]} options.
### Table 9.6. tpm Validation Checks

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackupDirectoryWriteableCheck [412]</td>
<td>Checks that the configured backup directory is writeable</td>
</tr>
<tr>
<td>BackupDumpDirectoryWriteableCheck [412]</td>
<td>Checks the backup temp directory is writeable</td>
</tr>
<tr>
<td>BackupScriptAvailableCheck [412]</td>
<td>Checks that the configured backup script exists and can be executed</td>
</tr>
<tr>
<td>ClusterDiagnosticCheck [412]</td>
<td></td>
</tr>
<tr>
<td>ClusterStatusCheck [412]</td>
<td></td>
</tr>
<tr>
<td>CommitDirectoryCheck [412]</td>
<td></td>
</tr>
<tr>
<td>ConfigurationStorageDirectoryCheck [413]</td>
<td></td>
</tr>
<tr>
<td>ConfigureValidationCheck [413]</td>
<td></td>
</tr>
<tr>
<td>ConfiguredDirectoryCheck [413]</td>
<td></td>
</tr>
<tr>
<td>ConflictingReplicationServiceTHLPortsCheck [413]</td>
<td>Ensures that the configured connector selection is valid</td>
</tr>
<tr>
<td>ConnectorChecks [413]</td>
<td></td>
</tr>
<tr>
<td>Connector@VersionCheck [413]</td>
<td></td>
</tr>
<tr>
<td>ConnectorListenerAddressCheck [413]</td>
<td></td>
</tr>
<tr>
<td>ConnectorRWROAddressesCheck [413]</td>
<td>Ensure the RW and RO addresses are different</td>
</tr>
<tr>
<td>ConnectorSmartScaleAllowedCheck [413]</td>
<td>Confirms whether SmartScale is valid within the current configured parameters</td>
</tr>
<tr>
<td>ConnectorUserCheck [413]</td>
<td></td>
</tr>
<tr>
<td>ConsistentReplicationCredentialsCheck [413]</td>
<td></td>
</tr>
<tr>
<td>CurrentCommandCoordinatorCheck [414]</td>
<td></td>
</tr>
<tr>
<td>CurrentConnectorCheck [414]</td>
<td></td>
</tr>
<tr>
<td>CurrentReleaseDirectoryIsSymlink [414]</td>
<td></td>
</tr>
<tr>
<td>CurrentTopologyCheck [414]</td>
<td></td>
</tr>
<tr>
<td>CurrentVersionCheck [414]</td>
<td></td>
</tr>
<tr>
<td>DatasourceBootScriptCheck [414]</td>
<td></td>
</tr>
<tr>
<td>DifferentMasterSlaveCheck [414]</td>
<td></td>
</tr>
<tr>
<td>DirectOracleServiceSIDCheck [414]</td>
<td></td>
</tr>
<tr>
<td>ElasticsearchValidationCheck [414]</td>
<td></td>
</tr>
<tr>
<td>EncryptionCheck [414]</td>
<td></td>
</tr>
<tr>
<td>EncryptionKeystoreCheck [414]</td>
<td></td>
</tr>
<tr>
<td>FileValidationCheck [414]</td>
<td></td>
</tr>
<tr>
<td>FirewallCheck [415]</td>
<td></td>
</tr>
<tr>
<td>GlobalHostAddressesCheck [415]</td>
<td></td>
</tr>
<tr>
<td>GlobalHostOracleLibrariesFoundCheck [415]</td>
<td></td>
</tr>
<tr>
<td>GlobalMatchingPingMethodCheck [415]</td>
<td></td>
</tr>
<tr>
<td>GlobalRestartComponentsCheck [415]</td>
<td></td>
</tr>
<tr>
<td>GroupValidationCheck [415]</td>
<td></td>
</tr>
<tr>
<td>HdfsValidationCheck [415]</td>
<td></td>
</tr>
<tr>
<td>HostLicensesCheck [415]</td>
<td></td>
</tr>
<tr>
<td>HostOracleLibrariesFoundCheck [415]</td>
<td></td>
</tr>
<tr>
<td>HostReplicatorServiceRunningCheck [415]</td>
<td></td>
</tr>
<tr>
<td>HostSkippedChecks [415]</td>
<td></td>
</tr>
<tr>
<td>HostnameCheck [415]</td>
<td></td>
</tr>
<tr>
<td>HostsFileCheck [416]</td>
<td></td>
</tr>
<tr>
<td>InstallServicesCheck [416]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InstallationScriptCheck [416]</td>
<td>Checks whether a master host has been defined for the configured service.</td>
</tr>
<tr>
<td>InstallerMasterSlaveCheck [416]</td>
<td></td>
</tr>
<tr>
<td>InstallingOverExistingInstallation [416]</td>
<td></td>
</tr>
<tr>
<td>JavaUserTimeZoneCheck [416]</td>
<td></td>
</tr>
<tr>
<td>JavaVersionCheck [416]</td>
<td></td>
</tr>
<tr>
<td>KeystoresCheck [416]</td>
<td></td>
</tr>
<tr>
<td>KeystoresToCommitCheck [416]</td>
<td></td>
</tr>
<tr>
<td>ManagerActiveWitnessConversionCheck [416]</td>
<td></td>
</tr>
<tr>
<td>ManagerChecks [416]</td>
<td></td>
</tr>
<tr>
<td>ManagerHeapThresholdCheck [416]</td>
<td></td>
</tr>
<tr>
<td>ManagerListenerAddressCheck [417]</td>
<td></td>
</tr>
<tr>
<td>ManagerPingMethodCheck [417]</td>
<td></td>
</tr>
<tr>
<td>ManagerWitnessAvailableCheck [417]</td>
<td></td>
</tr>
<tr>
<td>ManagerWitnessNeededCheck [417]</td>
<td></td>
</tr>
<tr>
<td>MatchingHomeDirectoryCheck [417]</td>
<td></td>
</tr>
<tr>
<td>MissingReplicationServiceConfigurationCheck [417]</td>
<td></td>
</tr>
<tr>
<td>ModifiedConfigurationFilesCheck [417]</td>
<td></td>
</tr>
<tr>
<td>MySQLAllowIntensiveChecks [417]</td>
<td>Enables searching MySQL INFORMATION_SCHEMA for validation checks</td>
</tr>
<tr>
<td>MySQLApplierLogsCheck [417]</td>
<td></td>
</tr>
<tr>
<td>MySQLApplierPortCheck [417]</td>
<td></td>
</tr>
<tr>
<td>MySQLApplierServerIDCheck [417]</td>
<td></td>
</tr>
<tr>
<td>MySQLAvailableCheck [418]</td>
<td>Checks if MySQL is installed</td>
</tr>
<tr>
<td>MySQLBinaryLogsEnabledCheck [418]</td>
<td>Checks that binary logging has been enabled on MySQL</td>
</tr>
<tr>
<td>MySQLBinLogDoDBCheck [418]</td>
<td></td>
</tr>
<tr>
<td>MySQLClientCheck [418]</td>
<td>Checks whether the MySQL client command tool is available</td>
</tr>
<tr>
<td>MySQLConfigFileCheck [418]</td>
<td>Checks the existence of a MySQL configuration file</td>
</tr>
<tr>
<td>MySQLConnectorBridgeModePermissionsCheck [418]</td>
<td></td>
</tr>
<tr>
<td>MySQLConnectorPermissionsCheck [418]</td>
<td></td>
</tr>
<tr>
<td>MySQLDefaulTableTypeCheck [418]</td>
<td>Checks the default table type for MySQL</td>
</tr>
<tr>
<td>MySQLDumpCheck [418]</td>
<td>Checks that the mysqldump command version matches the installed MySQL</td>
</tr>
<tr>
<td>MySQLgeneratedColumnCheck [418]</td>
<td>Checks whether MySQL virtual/generated columns are defined</td>
</tr>
<tr>
<td>MySQLInnoDBEnabledCheck [419]</td>
<td></td>
</tr>
<tr>
<td>MySQLJsonDataTypeCheck [419]</td>
<td></td>
</tr>
<tr>
<td>MySQLLoadDataInfilePermissionsCheck [419]</td>
<td></td>
</tr>
<tr>
<td>MySQLLoginCheck [419]</td>
<td>Checks whether Tungsten Clustering can connect to MySQL using the configured credentials</td>
</tr>
<tr>
<td>MySQLMyISAMCheck [419]</td>
<td>Checks for the existence of MyISAM tables</td>
</tr>
<tr>
<td>MySQLMySQLReplicationCheck [419]</td>
<td></td>
</tr>
<tr>
<td>MySQLPasswordSettingCheck [419]</td>
<td></td>
</tr>
<tr>
<td>MySQLPermissionsCheck [419]</td>
<td></td>
</tr>
<tr>
<td>MySQLReadableLogsCheck [419]</td>
<td></td>
</tr>
<tr>
<td>MySQLSettingsCheck [420]</td>
<td></td>
</tr>
<tr>
<td>MySQLSuperReadOnlyCheck [420]</td>
<td>Checks whether super_read_only has been enabled on MySQL</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MySQLTriggerCheck</td>
<td>[420]</td>
</tr>
<tr>
<td>MySQLUnsupportedDataTypesCheck</td>
<td>[420]</td>
</tr>
<tr>
<td>MysqlConnectorCheck</td>
<td>[420]</td>
</tr>
<tr>
<td>MysqldumpAvailableCheck</td>
<td>[420]</td>
</tr>
<tr>
<td>MysqldumpSettingsCheck</td>
<td>[420]</td>
</tr>
<tr>
<td>NewDirectoryRequiredCheck</td>
<td>[420]</td>
</tr>
<tr>
<td>NtpdRunningCheck</td>
<td>[420]</td>
</tr>
<tr>
<td>OSCheck</td>
<td>[420]</td>
</tr>
<tr>
<td>OldServicesRunningCheck</td>
<td>[420]</td>
</tr>
<tr>
<td>OpensslInitCheck</td>
<td>[421]</td>
</tr>
<tr>
<td>OpensslLibraryCheck</td>
<td>[421]</td>
</tr>
<tr>
<td>OracleLoginCheck</td>
<td>[421]</td>
</tr>
<tr>
<td>OraclePermissionsCheck</td>
<td>[421]</td>
</tr>
<tr>
<td>OracleRedoHeaderMinerDirectoryCheck</td>
<td>[421]</td>
</tr>
<tr>
<td>OracleServiceSIDCheck</td>
<td>[421]</td>
</tr>
<tr>
<td>OracleVersionCheck</td>
<td>[421]</td>
</tr>
<tr>
<td>PGAvailableCheck</td>
<td>[421]</td>
</tr>
<tr>
<td>ParallelReplicationCheck</td>
<td>[421]</td>
</tr>
<tr>
<td>ParallelReplicationCountCheck</td>
<td>[421]</td>
</tr>
<tr>
<td>PgControlAvailableCheck</td>
<td>[421]</td>
</tr>
<tr>
<td>PgStandbyAvailableCheck</td>
<td>[421]</td>
</tr>
<tr>
<td>PgdumpAvailableCheck</td>
<td>[422]</td>
</tr>
<tr>
<td>PgdumpallAvailableCheck</td>
<td>[422]</td>
</tr>
<tr>
<td>PingSyntaxCheck</td>
<td>[422]</td>
</tr>
<tr>
<td>PortAvailabilityCheck</td>
<td>[422]</td>
</tr>
<tr>
<td>ProfileScriptCheck</td>
<td>[422]</td>
</tr>
<tr>
<td>RMIListenerAddressCheck</td>
<td>[422]</td>
</tr>
<tr>
<td>RelayDirectoryWriteableCheck</td>
<td>[422] Checks that the relay log directory can be written to</td>
</tr>
<tr>
<td>ReplicatorChecks</td>
<td>[422]</td>
</tr>
<tr>
<td>RestartComponentsCheck</td>
<td>[422]</td>
</tr>
<tr>
<td>RouterAffinityCheck</td>
<td>[422]</td>
</tr>
<tr>
<td>RouterBridgeModeDefaultCheck</td>
<td>[422]</td>
</tr>
<tr>
<td>RouterDelayBeforeOfflineCheck</td>
<td>[422]</td>
</tr>
<tr>
<td>RouterKeepAliveTimeoutCheck</td>
<td>[423]</td>
</tr>
<tr>
<td>RowBasedBinaryLoggingCheck</td>
<td>[423] Checks that Row-based binary logging has been enabled for heterogeneous deployments</td>
</tr>
<tr>
<td>RsyncAvailableCheck</td>
<td>[423]</td>
</tr>
<tr>
<td>RubyVersionCheck</td>
<td>[423]</td>
</tr>
<tr>
<td>SSHLoginCheck</td>
<td>[423] Checks connectivity to other hosts over SSH</td>
</tr>
<tr>
<td>ServiceTransferredLogStorageCheck</td>
<td>[423]</td>
</tr>
<tr>
<td>StartingStoppedServices</td>
<td>[423]</td>
</tr>
<tr>
<td>SudoCheck</td>
<td>[423]</td>
</tr>
<tr>
<td>SwappinessCheck</td>
<td>[424] Checks the swappiness OS configuration is within a recommended range</td>
</tr>
<tr>
<td>THLDirectoryWriteableCheck</td>
<td>[424]</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>THLListenerAddressCheck [424]</td>
<td></td>
</tr>
<tr>
<td>THLSchemaChangeCheck [424]</td>
<td>Ensures that the existing THL format is compatible with the new release</td>
</tr>
<tr>
<td>THLStorageCheck [424]</td>
<td>Confirms the THL storage directory exists, is empty and writable</td>
</tr>
<tr>
<td>THLStorageChecksum [424]</td>
<td></td>
</tr>
<tr>
<td>TargetDirectoryDoesNotExist [424]</td>
<td></td>
</tr>
<tr>
<td>TransferredLogStorageCheck [424]</td>
<td></td>
</tr>
<tr>
<td>UpgradeSameProductCheck [424]</td>
<td>Ensures that the same product is being updated</td>
</tr>
<tr>
<td>VIPEnabledHostAllowsRootCommands [424]</td>
<td></td>
</tr>
<tr>
<td>VIPEnabledHostArpPath [425]</td>
<td></td>
</tr>
<tr>
<td>VIPEnabledHostIfconfigPath [425]</td>
<td></td>
</tr>
<tr>
<td>VerticaUserGroupsCheck [425]</td>
<td>Checks that the Vertica user has the correct OS group membership</td>
</tr>
<tr>
<td>WhichAvailableCheck [425]</td>
<td>Checks the existence of a working which command</td>
</tr>
<tr>
<td>WriteableHomeDirectoryCheck [425]</td>
<td>Ensures the home directory can be written to</td>
</tr>
<tr>
<td>WriteableTempDirectoryCheck [425]</td>
<td>Ensures the temporary directory can be written to</td>
</tr>
<tr>
<td>XtrabackupAvailableCheck [425]</td>
<td></td>
</tr>
<tr>
<td>XtrabackupDirectoryWriteableCheck [425]</td>
<td></td>
</tr>
<tr>
<td>XtrabackupSettingsCheck [425]</td>
<td></td>
</tr>
<tr>
<td>BackupDirectoryWriteableCheck [412]</td>
<td>Confirms that the directory defined in --backup-dir directory exists and can be written to.</td>
</tr>
<tr>
<td>BackupDumpDirectoryWriteableCheck [412]</td>
<td>Confirms that the directory defined in --backup-dump-dir directory exists and can be written to.</td>
</tr>
<tr>
<td>BackupScriptAvailableCheck [412]</td>
<td>Confirms that the script defined in --backup-script [439] exists and is executable.</td>
</tr>
<tr>
<td>ClusterDiagnosticCheck [412]</td>
<td></td>
</tr>
<tr>
<td>ClusterStatusCheck [412]</td>
<td></td>
</tr>
<tr>
<td>CommitDirectoryCheck [412]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>ConfigurationStorageDirectoryCheck</td>
<td>ConfigureValidationCheck</td>
</tr>
<tr>
<td>ConfiguredDirectoryCheck</td>
<td>ConnectorChecks</td>
</tr>
<tr>
<td>ConflictingReplicationServiceTHLPortsCheck</td>
<td>ConnectorDBVersionCheck</td>
</tr>
<tr>
<td>ConnectorListenerAddressCheck</td>
<td>ConnectorRWROAddressesCheck</td>
</tr>
<tr>
<td>ConnectorSmartScaleAllowedCheck</td>
<td>ConnectorUserCheck</td>
</tr>
<tr>
<td>ConsistentReplicationCredentialsCheck</td>
<td></td>
</tr>
</tbody>
</table>

Options:
- **ConfigurationStorageDirectoryCheck**: ConfigureValidationCheck
- **ConfiguredDirectoryCheck**: ConnectorChecks
- **ConflictingReplicationServiceTHLPortsCheck**: ConnectorDBVersionCheck
- **ConnectorListenerAddressCheck**: ConnectorRWROAddressesCheck
- **ConnectorSmartScaleAllowedCheck**: ConnectorUserCheck
- **ConsistentReplicationCredentialsCheck**
### The tpm Deployment Command

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>- CurrentCommandCoordinatorCheck [414]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>CurrentCommandCoordinatorCheck [414]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- CurrentConnectorCheck [414]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>CurrentConnectorCheck [414]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- CurrentReleaseDirectoryIsSymlink [414]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>CurrentReleaseDirectoryIsSymlink [414]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- CurrentTopologyCheck [414]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>CurrentTopologyCheck [414]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- CurrentVersionCheck [414]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>CurrentVersionCheck [414]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- DatasourceBootScriptCheck [414]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>DatasourceBootScriptCheck [414]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- DifferentMasterSlaveCheck [414]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>DifferentMasterSlaveCheck [414]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- DirectOracleServiceSIDCheck [414]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>DirectOracleServiceSIDCheck [414]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- ElasticsearchValidationCheck [414]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>ElasticsearchValidationCheck [414]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- EncryptionCheck [414]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>EncryptionCheck [414]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- EncryptionKeystoreCheck [414]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>EncryptionKeystoreCheck [414]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- FileValidationCheck [414]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>FileValidationCheck [414]</td>
</tr>
<tr>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>- FirewallCheck [415]</td>
<td>FirewallCheck [415]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>- GlobalHostAddressesCheck [415]</td>
<td>GlobalHostAddressesCheck [415]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>- GlobalMatchingPingMethodCheck [415]</td>
<td>GlobalMatchingPingMethodCheck [415]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>- GlobalRestartComponentsCheck [415]</td>
<td>GlobalRestartComponentsCheck [415]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>- GroupValidationCheck [415]</td>
<td>GroupValidationCheck [415]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>- HdfsValidationCheck [415]</td>
<td>HdfsValidationCheck [415]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>- HostLicensesCheck [415]</td>
<td>HostLicensesCheck [415]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>- HostOracleLibrariesFoundCheck [415]</td>
<td>HostOracleLibrariesFoundCheck [415]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>- HostReplicatorServiceRunningCheck [415]</td>
<td>HostReplicatorServiceRunningCheck [415]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>- HostSkippedChecks [415]</td>
<td>HostSkippedChecks [415]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>- HostnameCheck [415]</td>
<td>HostnameCheck [415]</td>
</tr>
</tbody>
</table>
The tpm Deployment Command

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HostsFileCheck [416]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>InstallServicesCheck [416]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>InstallationScriptCheck [416]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>InstallerMasterSlaveCheck [416]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Checks whether a master host has been defined for the configured service.</td>
</tr>
<tr>
<td>InstallingOverExistingInstallation [416]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>JavaUserTimezoneCheck [416]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>JavaVersionCheck [416]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>KeystoresCheck [416]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>KeystoresToCommitCheck [416]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>ManagerActiveWitnessConversionCheck [416]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>ManagerChecks [416]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>ManagerHeapThresholdCheck [416]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>- ManagerListenerAddressCheck [417]</td>
<td></td>
</tr>
<tr>
<td>Option                       ManagerListenerAddressCheck [417]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- ManagerPingMethodCheck [417]</td>
<td></td>
</tr>
<tr>
<td>Option                       ManagerPingMethodCheck [417]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- ManagerWitnessAvailableCheck [417]</td>
<td></td>
</tr>
<tr>
<td>Option                       ManagerWitnessAvailableCheck [417]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- ManagerWitnessNeededCheck [417]</td>
<td></td>
</tr>
<tr>
<td>Option                       ManagerWitnessNeededCheck [417]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- MatchingHomeDirectoryCheck [417]</td>
<td></td>
</tr>
<tr>
<td>Option                       MatchingHomeDirectoryCheck [417]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- MissingReplicationServiceConfigurationCheck [417]</td>
<td></td>
</tr>
<tr>
<td>Option                       MissingReplicationServiceConfigurationCheck [417]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- ModifiedConfigurationFilesCheck [417]</td>
<td></td>
</tr>
<tr>
<td>Option                       ModifiedConfigurationFilesCheck [417]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- MySQLAllowIntensiveChecks [417]</td>
<td></td>
</tr>
<tr>
<td>Option                       MySQLAllowIntensiveChecks [417]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Enables searching MySQL INFORMATION_SCHEMA for validation checks</td>
</tr>
</tbody>
</table>

Enables `tpm` to make use of the MySQL INFORMATION_SCHEMA to perform various validation checks. These include, but are not limited to:

- Tables not configured to use transactional tables
- Unsupported datatypes in MySQL tables
- MySQLApplierLogsCheck [417]
- MySQLApplierPortCheck [417]
- MySQLApplierServerIDCheck [417]
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQLAvailableCheck [418]</td>
<td>Checks if MySQL is installed</td>
</tr>
<tr>
<td>MySQLBinaryLogsEnabledCheck [418]</td>
<td>Checks that binary logging has been enabled on MySQL. Examines the <code>log_bin</code> variable has been defined within the running MySQL server. Binary logging must be enabled for replication to work.</td>
</tr>
<tr>
<td>MySQLBinlogDoDbCheck [418]</td>
<td></td>
</tr>
<tr>
<td>MySQLClientCheck [418]</td>
<td>Checks whether the MySQL client command tool is available</td>
</tr>
<tr>
<td>MySQLConfigFileCheck [418]</td>
<td>Checks the existence of a MySQL configuration file</td>
</tr>
<tr>
<td>MySQLConnectorBridgeModePermissionsCheck [418]</td>
<td></td>
</tr>
<tr>
<td>MySQLConnectorPermissionsCheck [418]</td>
<td></td>
</tr>
<tr>
<td>MySQLDefaultTableTypeCheck [418]</td>
<td>Checks that the default table type configured for MySQL is a compatible transactional storage engine such as InnoDB</td>
</tr>
<tr>
<td>MySQLDumpCheck [418]</td>
<td>Checks whether the <code>mysqldump</code> command within the configured PATH matches the version of MySQL being configured as a source or target. A mismatch could indicate that multiple MySQL versions are installed. A mismatch could create invalid or corrupt backups. Either correct your PATH or use --preferred-path [466] to point to the correct MySQL installation.</td>
</tr>
<tr>
<td>MySQLGeneratedColumnCheck [418]</td>
<td>Checks whether MySQL virtual/generated columns are defined</td>
</tr>
</tbody>
</table>
The tpm Deployment Command

Checks, whether any tables contain generated or virtual columns. The test is only executed on MySQL 5.7 and only if `--mysql-allow-intensive-checks` has been enabled.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQLInnoDBEnabledCheck</td>
<td>Checks, whether any tables contain JSON columns. The test is only executed on MySQL 5.7 and only if <code>--mysql-allow-intensive-checks</code> has been enabled.</td>
</tr>
</tbody>
</table>
| MySQLJsonDataTypeCheck | Checks for the existence of MyISAM tables. Use of MyISAM tables is not supported since MyISAM is not transactionally consistent. This can cause problems for both extraction and applying data. In order to check for the existence of MyISAM tables, tpm uses two techniques:  
  • Looking for .MYD files within the MySQL directory, which are the files which contains MyISAM data. tpm must be able to read and see the contents of the MySQL data directory. If the configured user does not already have access, you can use the `--root-command-prefix=true` option to grant root access to access the filesystem.  
  • Using the MySQL INFORMATION_SCHEMA to look for tables defined with the MyISAM engine. For this option to work, intensive checks must have been enabled using `--mysql-allow-intensive-checks`. If neither of these methods is available, the check will fail and installation will stop. |
| MySQLLoadDataInfilePermissionsCheck | Checks whether Tungsten Clustering can connect to MySQL using the configured credentials |
| MySQLMyISAMCheck | Checks for the existence of MyISAM tables within the database. Use of MyISAM tables is not supported since MyISAM is not transactionally consistent. This can cause problems for both extraction and applying data. In order to check for the existence of MyISAM tables, tpm uses two techniques:  
  • Looking for .MYD files within the MySQL directory, which are the files which contains MyISAM data. tpm must be able to read and see the contents of the MySQL data directory. If the configured user does not already have access, you can use the `--root-command-prefix=true` option to grant root access to access the filesystem.  
  • Using the MySQL INFORMATION_SCHEMA to look for tables defined with the MyISAM engine. For this option to work, intensive checks must have been enabled using `--mysql-allow-intensive-checks`. If neither of these methods is available, the check will fail and installation will stop. |
| MySQLNoMySQLReplicationCheck | Checks whether Tungsten Clustering can connect to MySQL using the configured credentials |
| MySQLPasswordSettingCheck | Checks for the existence of MyISAM tables. Use of MyISAM tables is not supported since MyISAM is not transactionally consistent. This can cause problems for both extraction and applying data. In order to check for the existence of MyISAM tables, tpm uses two techniques:  
  • Looking for .MYD files within the MySQL directory, which are the files which contains MyISAM data. tpm must be able to read and see the contents of the MySQL data directory. If the configured user does not already have access, you can use the `--root-command-prefix=true` option to grant root access to access the filesystem.  
  • Using the MySQL INFORMATION_SCHEMA to look for tables defined with the MyISAM engine. For this option to work, intensive checks must have been enabled using `--mysql-allow-intensive-checks`. If neither of these methods is available, the check will fail and installation will stop. |
| MySQLPermissionsCheck | Checks whether Tungsten Clustering can connect to MySQL using the configured credentials |
| MySQLReadableLogsCheck | Checks for the existence of MyISAM tables within the database. Use of MyISAM tables is not supported since MyISAM is not transactionally consistent. This can cause problems for both extraction and applying data. In order to check for the existence of MyISAM tables, tpm uses two techniques:  
  • Looking for .MYD files within the MySQL directory, which are the files which contains MyISAM data. tpm must be able to read and see the contents of the MySQL data directory. If the configured user does not already have access, you can use the `--root-command-prefix=true` option to grant root access to access the filesystem.  
  • Using the MySQL INFORMATION_SCHEMA to look for tables defined with the MyISAM engine. For this option to work, intensive checks must have been enabled using `--mysql-allow-intensive-checks`. If neither of these methods is available, the check will fail and installation will stop. |
The tpm Deployment Command

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQLSettingsCheck [420]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

Checks whether the `super_read_only` variable within MySQL has been enabled. If enabled, replication will not work. The check will test both the running server and the configuration file to determine whether the value has been enabled.

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQLSuperReadOnlyCheck [420]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQLTriggerCheck [420]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQLUnsupportedDataTypesCheck [420]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MysqlConnectorCheck [420]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MysqldumpAvailableCheck [420]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MysqldumpSettingsCheck [420]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewDirectoryRequiredCheck [420]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NtpdRunningCheck [420]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSCheck [420]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OldServicesRunningCheck [420]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>
### The tpm Deployment Command

- **OpenFilesInitCheck** [421]
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenFilesInitCheck [421]</td>
<td></td>
</tr>
</tbody>
</table>

- **OpensslLibraryCheck** [421]
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpensslLibraryCheck [421]</td>
<td></td>
</tr>
</tbody>
</table>

- **OracleLoginCheck** [421]
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleLoginCheck [421]</td>
<td></td>
</tr>
</tbody>
</table>

- **OraclePermissionsCheck** [421]
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OraclePermissionsCheck [421]</td>
<td></td>
</tr>
</tbody>
</table>

- **OracleRedoReaderMinerDirectoryCheck** [421]
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleRedoReaderMinerDirectoryCheck [421]</td>
<td></td>
</tr>
</tbody>
</table>

- **OracleServiceSIDCheck** [421]
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleServiceSIDCheck [421]</td>
<td></td>
</tr>
</tbody>
</table>

- **OracleVersionCheck** [421]
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OracleVersionCheck [421]</td>
<td></td>
</tr>
</tbody>
</table>

- **PGAvailableCheck** [421]
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGAvailableCheck [421]</td>
<td></td>
</tr>
</tbody>
</table>

- **ParallelReplicationCheck** [421]
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParallelReplicationCheck [421]</td>
<td></td>
</tr>
</tbody>
</table>

- **ParallelReplicationCountCheck** [421]
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParallelReplicationCountCheck [421]</td>
<td></td>
</tr>
</tbody>
</table>

- **PgControlAvailableCheck** [421]
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PgControlAvailableCheck [421]</td>
<td></td>
</tr>
</tbody>
</table>

- **PgStandbyAvailableCheck** [421]
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PgStandbyAvailableCheck [421]</td>
<td></td>
</tr>
</tbody>
</table>
## The `tpm` Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PgdumpAvailableCheck</code></td>
<td></td>
</tr>
<tr>
<td><code>PgdumpallAvailableCheck</code></td>
<td></td>
</tr>
<tr>
<td><code>PingSyntaxCheck</code></td>
<td></td>
</tr>
<tr>
<td><code>PortAvailabilityCheck</code></td>
<td></td>
</tr>
<tr>
<td><code>ProfileScriptCheck</code></td>
<td></td>
</tr>
<tr>
<td><code>RMIListenerAddressCheck</code></td>
<td></td>
</tr>
<tr>
<td><code>RelayDirectoryWriteableCheck</code></td>
<td>Checks that the relay log directory can be written to. Confirms that the directory defined in <code>--relay-log-dir</code> directory exists and can be written to.</td>
</tr>
<tr>
<td><code>ReplicatorChecks</code></td>
<td></td>
</tr>
<tr>
<td><code>RestartComponentsCheck</code></td>
<td></td>
</tr>
<tr>
<td><code>RouterAffinityCheck</code></td>
<td></td>
</tr>
<tr>
<td><code>RouterBridgeModeDefaultCheck</code></td>
<td></td>
</tr>
<tr>
<td><code>RouterDelayBeforeOfflineCheck</code></td>
<td></td>
</tr>
</tbody>
</table>
The tpm Deployment Command

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RouterKeepAliveTimeoutCheck [423]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>RouterKeepAliveTimeoutCheck [423]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

RowBasedBinaryLoggingCheck [423]

| Description | Checks that Row-based binary logging has been enabled for heterogeneous deployments |
| Option | RowBasedBinaryLoggingCheck [423] |

For heterogeneous deployments, row-based binary logging must have been enabled. For all services where heterogeneous support has been enabled, for example due to `--enable-heterogeneous-service [453]` or `--enable-batch-service` (in [Tungsten Replicator 6.1 Manual]), row-based logging within MySQL must have been switched on. The test looks for the value of `binlog_format=ROW`.

| RsyncAvailableCheck [423] |  |
| Option | RsyncAvailableCheck [423] |
| Description |  |

RubyVersionCheck [423]

| Description |  |
| Option | RubyVersionCheck [423] |

SSHLoginCheck [423]

| Description | Checks connectivity to other hosts over SSH |
| Option | SSHLoginCheck [423] |

Checks to confirm the SSH logins to other hosts in the cluster work, without requiring a password, and without returning additional rows of information when directly, remotely, running a command. In the event of the check failing, the following items should be checked:

- Confirm that it is possible to SSH to the remote site using the username provided, and without requiring a password. For example:
  ```sh
guest:shell> ssh tungsten@host2
guest:shell>
```

- Remove any remote messages returned when the user logs in. This includes the output from the `Banner` argument within `/etc/ssh/sshd_config`, or text or files output by the users shell login script or profile.

- Ensure that your remote shell has not been configured to output text or a message when a logout is attempted, for example by using:
  ```sh
cshell> trap "echo logout" 0
```

ServiceTransferredLogStorageCheck [423]

| Option | ServiceTransferredLogStorageCheck [423] |
| Description |  |

StartingStoppedServices [423]

| Option | StartingStoppedServices [423] |
| Description |  |

SudoCheck [423]

| Option | SudoCheck [423] |
| Description |  |
### SwappinessCheck [424]

<table>
<thead>
<tr>
<th>Option</th>
<th>SwappinessCheck [424]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Checks the swappiness OS configuration is within a recommended range</td>
</tr>
</tbody>
</table>

Checks whether the Linux swappiness parameter has been set to a value of 10 or less, both in the current setting and when the system reboots. A value greater than 10 may allow for running programs to be swapped out, which will affect the performance of the Tungsten Clustering when running. Change the value in `sysctl.conf`.

### THLDirectoryWriteableCheck [424]

<table>
<thead>
<tr>
<th>Option</th>
<th>THLDirectoryWriteableCheck [424]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

### THListenerAddressCheck [424]

<table>
<thead>
<tr>
<th>Option</th>
<th>THListenerAddressCheck [424]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

### THLSchemaChangeCheck [424]

<table>
<thead>
<tr>
<th>Option</th>
<th>THLSchemaChangeCheck [424]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Ensures that the existing THL format is compatible with the new release</td>
</tr>
</tbody>
</table>

Checks that the format of the current THL is compatible with the schema and format of the new software. A difference may mean that the THL needs to be reset before installation can continue.

### THLStorageCheck [424]

<table>
<thead>
<tr>
<th>Option</th>
<th>THLStorageCheck [424]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Confirms the THL storage directory exists, is empty and writeable</td>
</tr>
</tbody>
</table>

Confirms that the directory configured for THL storage using `--log-dir` directory exists, is writeable, and is empty.

### THLStorageChecksum [424]

<table>
<thead>
<tr>
<th>Option</th>
<th>THLStorageChecksum [424]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

### TargetDirectoryDoesNotExist [424]

<table>
<thead>
<tr>
<th>Option</th>
<th>TargetDirectoryDoesNotExist [424]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

### TransferredLogStorageCheck [424]

<table>
<thead>
<tr>
<th>Option</th>
<th>TransferredLogStorageCheck [424]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

### UpgradeSameProductCheck [424]

<table>
<thead>
<tr>
<th>Option</th>
<th>UpgradeSameProductCheck [424]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Ensures that the same product is being updated</td>
</tr>
</tbody>
</table>

Updates must occur with the same product, for example, Tungsten Replicator to Tungsten Replicator. It is not possible to update replicator to cluster, or cluster to replicator.

### VIPEnabledHostAllowsRootCommands [424]

<table>
<thead>
<tr>
<th>Option</th>
<th>VIPEnabledHostAllowsRootCommands [424]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>
The tpm Deployment Command

- **VIPEnabledHostArpPath** [425]

<table>
<thead>
<tr>
<th>Option</th>
<th>VIPEnabledHostArpPath [425]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

- **VIPEnabledHostIfconfigPath** [425]

<table>
<thead>
<tr>
<th>Option</th>
<th>VIPEnabledHostIfconfigPath [425]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

- **VerticaUserGroupsCheck** [425]

<table>
<thead>
<tr>
<th>Option</th>
<th>VerticaUserGroupsCheck [425]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Checks that the Vertica user has the correct OS group membership</td>
</tr>
</tbody>
</table>

Checks whether the user running Vertica is a member of the tungsten user's primary group. Without this setting, the CSV files generated by the replicator would not be readable by Vertica when importing them into the database during batchloading.

- **WhichAvailableCheck** [425]

<table>
<thead>
<tr>
<th>Option</th>
<th>WhichAvailableCheck [425]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Checks the existence of a working which command</td>
</tr>
</tbody>
</table>

Checks the existence of a working `which` command.

- **WriteableHomeDirectoryCheck** [425]

<table>
<thead>
<tr>
<th>Option</th>
<th>WriteableHomeDirectoryCheck [425]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Ensures the home directory can be written to</td>
</tr>
</tbody>
</table>

Checks that the home directory for the configured user can be written to.

- **WriteableTempDirectoryCheck** [425]

<table>
<thead>
<tr>
<th>Option</th>
<th>WriteableTempDirectoryCheck [425]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Ensures the temporary directory can be written to</td>
</tr>
</tbody>
</table>

The temporary directory is used during installation to store a variety of information. This check ensures that the directory is writeable, and that files can be created and deleted correctly.

- **XtrabackupAvailableCheck** [425]

<table>
<thead>
<tr>
<th>Option</th>
<th>XtrabackupAvailableCheck [425]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

- **XtrabackupDirectoryWriteableCheck** [425]

<table>
<thead>
<tr>
<th>Option</th>
<th>XtrabackupDirectoryWriteableCheck [425]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

- **XtrabackupSettingsCheck** [425]

<table>
<thead>
<tr>
<th>Option</th>
<th>XtrabackupSettingsCheck [425]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

### 9.8. tpm Configuration Options

tpm supports a large range of configuration options, which can be specified either:

- On the command-line, using a double-dash prefix, i.e. `--repl-thl-log-retention=3d` [476]
- In an INI file, without the double-dash prefix, i.e. `repl-thl-log-retention=3d` [476]
A full list of all the available options supported is provided in Table 9.7, "tpm Configuration Options".

### Table 9.7. tpm Configuration Options

<table>
<thead>
<tr>
<th>CmdLine Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--connector-reset-when-affinity-back [435]</td>
<td>connector-reset-when-affinity-back [435]</td>
<td>Forces reconnection of clients to datasources with the configured affinity when they become available</td>
</tr>
<tr>
<td>--allow-bidi-unsafe [436], --repl-allow-bidi-unsafe [436]</td>
<td>allow-bidi-unsafe [436], repl-allow-bidi-unsafe [436]</td>
<td>Allow unsafe SQL from remote service</td>
</tr>
<tr>
<td>--api [436], --repl-api [436]</td>
<td>api [436], repl-api [436]</td>
<td>Enable the replication API</td>
</tr>
<tr>
<td>--api-host [436], --repl-api-host [436]</td>
<td>api-host [436], repl-api-host [436]</td>
<td>Hostname that the replication API should listen on</td>
</tr>
<tr>
<td>--api-password [436], --repl-api-password [436]</td>
<td>api-password [436], repl-api-password [436]</td>
<td>HTTP basic auth password for the replication API</td>
</tr>
<tr>
<td>--api-port [436], --repl-api-port [436]</td>
<td>api-port [436], repl-api-port [436]</td>
<td>Port that the replication API should bind to</td>
</tr>
<tr>
<td>--api-user [437], --repl-api-user [437]</td>
<td>api-user [437], repl-api-user [437]</td>
<td>HTTP basic auth username for the replication API</td>
</tr>
<tr>
<td>--application-password [437], --connector-password [437]</td>
<td>application-password [437], connector-password [437]</td>
<td>Database password for the connector</td>
</tr>
<tr>
<td>--application-port [437], --connector-listen-port [437]</td>
<td>application-port [437], connector-listen-port [437]</td>
<td>Port for the connector to listen on</td>
</tr>
<tr>
<td>--application-readonly-port [437], --connector-readonly-listen-port [437]</td>
<td>application-readonly-port [437], connector-readonly-listen-port [437]</td>
<td>Port for the connector to listen for read-only connections on</td>
</tr>
<tr>
<td>--application-user [437], --connector-user [437]</td>
<td>application-user [437], connector-user [437]</td>
<td>Database username for the connector</td>
</tr>
<tr>
<td>--auto-enable [437], --repl-auto-enable [437]</td>
<td>auto-enable [437], repl-auto-enable [437]</td>
<td>Auto-enable services after start-up</td>
</tr>
<tr>
<td>--auto-recovery-delay-interval [438], --repl-auto-recovery-delay-interval [438]</td>
<td>auto-recovery-delay-interval [438], repl-auto-recovery-delay-interval [438]</td>
<td>Delay between going OFFLINE and attempting to go ONLINE</td>
</tr>
<tr>
<td>--auto-recovery-max-attempts [438], --repl-auto-recovery-max-attempts [438]</td>
<td>auto-recovery-max-attempts [438], repl-auto-recovery-max-attempts [438]</td>
<td>Maximum number of attempts at automatic recovery</td>
</tr>
<tr>
<td>--auto-recovery-reset-interval [438], --repl-auto-recovery-reset-interval [438]</td>
<td>auto-recovery-reset-interval [438], repl-auto-recovery-reset-interval [438]</td>
<td>Delay before autorecovery is deemed to have succeeded</td>
</tr>
<tr>
<td>--backup-directory [438], --repl-backup-directory [438]</td>
<td>backup-directory [438], repl-backup-directory [438]</td>
<td>Permanent backup storage directory</td>
</tr>
<tr>
<td>--backup-online [439], --repl-backup-online [439]</td>
<td>backup-online [439], repl-backup-online [439]</td>
<td>Does the backup script support backing up a datasource while it is ONLINE</td>
</tr>
<tr>
<td>--backup-retention [439], --repl-backup-retention [439]</td>
<td>backup-retention [439], repl-backup-retention [439]</td>
<td>Number of backups to retain</td>
</tr>
<tr>
<td>--backup-script [439], --repl-backup-script [439]</td>
<td>backup-script [439], repl-backup-script [439]</td>
<td>What is the path to the backup script</td>
</tr>
<tr>
<td>--batch-enabled [439]</td>
<td>batch-enabled [439]</td>
<td>Should the replicator service use a batch applier</td>
</tr>
<tr>
<td>--batch-load-template [440]</td>
<td>batch-load-template [440]</td>
<td>Value for the loadBatchTemplate property</td>
</tr>
<tr>
<td>CmdLine Option</td>
<td>INI File Option</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>--cctrl-column-width [440]</td>
<td>cctrl-column-width [440]</td>
<td>Sets the minimum column width for service names within cctrl</td>
</tr>
<tr>
<td>--channels [440], repl-channels [440]</td>
<td>channels [440], repl-channels [440]</td>
<td>Number of replication channels to use for services</td>
</tr>
<tr>
<td>--composite-datasources [441], dataservice-composite-datasources [441]</td>
<td>composite-datasources [441], dataservice-composite-datasources [441]</td>
<td>Data services that should be added to this composite data service</td>
</tr>
<tr>
<td>--config-file-help [441]</td>
<td>config-file-help [441]</td>
<td>Display help information for content of the config file</td>
</tr>
<tr>
<td>--conn-java-mem-size [441]</td>
<td>conn-java-mem-size [441]</td>
<td>Connector Java heap memory size used to buffer data between clients and databases</td>
</tr>
<tr>
<td>--conn-round-robin-include-master [441]</td>
<td>conn-round-robin-include-master [441]</td>
<td>Should the Connector include the master in round-robin load balancing</td>
</tr>
<tr>
<td>--connector-affinity [441]</td>
<td>connector-affinity [441]</td>
<td>The default affinity for all connections</td>
</tr>
<tr>
<td>--connector-autoreconnect [442]</td>
<td>connector-autoreconnect [442]</td>
<td>Enable auto-reconnect in the connector</td>
</tr>
<tr>
<td>--connector-autoreconnect-killed-connections [442]</td>
<td>connector-autoreconnect-killed-connections [442]</td>
<td>Enable autoreconnect for connections killed within the connector</td>
</tr>
<tr>
<td>--connector-bridge-mode [442], --enable-connector-bridge-mode [442]</td>
<td>connector-bridge-mode [442], enable-connector-bridge-mode [442]</td>
<td>Enable the Tungsten Connector bridge mode</td>
</tr>
<tr>
<td>--connector-delete-user-map [442]</td>
<td>connector-delete-user-map [442]</td>
<td>Overwrite an existing user.map file</td>
</tr>
<tr>
<td>--connector-drop-after-max-connections [443]</td>
<td>connector-drop-after-max-connections [443]</td>
<td>Instantly drop connections that arrive after --connector-max-connections has been reached</td>
</tr>
<tr>
<td>--connector-keepalive-timeout [443]</td>
<td>connector-keepalive-timeout [443]</td>
<td>The Connector keep-alive SELECT statement is submitted if the time since the last activity reaches this timeout value.</td>
</tr>
<tr>
<td>--connector-listen-interface [443]</td>
<td>connector-listen-interface [443]</td>
<td>Listen interface to use for the connector</td>
</tr>
<tr>
<td>--connector-max-connections [443]</td>
<td>connector-max-connections [443]</td>
<td>The maximum number of connections the connector should allow at any time</td>
</tr>
<tr>
<td>--connector-max-slave-latency [443], --connector-max-applied-latency [443]</td>
<td>connector-max-slave-latency [443], connector-max-applied-latency [443]</td>
<td>The maximum applied latency for slave connections</td>
</tr>
<tr>
<td>--connector-readonly [443], --enable-connector-readonly [443]</td>
<td>connector-readonly [443], enable-connector-readonly [443]</td>
<td>Enable the Tungsten Connector read-only mode</td>
</tr>
<tr>
<td>--connector-ro-addresses [443]</td>
<td>connector-ro-addresses [443]</td>
<td>Connector addresses that should receive a r/o connection</td>
</tr>
</tbody>
</table>

427
<table>
<thead>
<tr>
<th>CmdLine Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repl-datasource-oracle-scan</td>
<td>datasource-oracle-scan</td>
<td>The default log directory for Oracle SCAN</td>
</tr>
<tr>
<td>repl-datasource-mysql-ssl-cert</td>
<td>datasource-mysql-ssl-cert</td>
<td>Defines whether SSL is used for MySQL SSL</td>
</tr>
<tr>
<td>repl-datasource-mysql-ssl-ca</td>
<td>datasource-mysql-ssl-ca</td>
<td>The default SSL CA file for MySQL SSL</td>
</tr>
<tr>
<td>repl-datasource-mysql-iblog-directory</td>
<td>datasource-mysql-iblog-directory</td>
<td>MySQL InnoDB data directory</td>
</tr>
<tr>
<td>repl-datasource-mysql-data-directory</td>
<td>datasource-mysql-data-directory</td>
<td>MySQL data directory</td>
</tr>
<tr>
<td>repl-datasource-mysql-ibdata-directory</td>
<td>datasource-mysql-ibdata-directory</td>
<td>MySQL InnoDB data directory</td>
</tr>
<tr>
<td>repl-datasource-mysql-log-pattern</td>
<td>datasource-mysql-log-pattern</td>
<td>MySQL log directory</td>
</tr>
<tr>
<td>repl-datasource-mysql-log-directory</td>
<td>datasource-mysql-log-directory</td>
<td>MySQL log directory</td>
</tr>
<tr>
<td>repl-datasource-mysql-ssl-capable</td>
<td>datasource-mysql-ssl-capable</td>
<td>Connector addresses that should receive session messages</td>
</tr>
<tr>
<td>repl-datasource-relay-enabled</td>
<td>datasource-relay-enabled</td>
<td>Make this dataservice the slave of another</td>
</tr>
<tr>
<td>repl-datasource-thl-port</td>
<td>datasource-thl-port</td>
<td>Port to use for THL operations</td>
</tr>
<tr>
<td>repl-datasource-use-relative-latency</td>
<td>datasource-use-relative-latency</td>
<td>Enable the cluster to operate on relative latency</td>
</tr>
<tr>
<td>repl-datasource-vip-enabled</td>
<td>datasource-vip-enabled</td>
<td>Is VIP management enabled?</td>
</tr>
<tr>
<td>repl-datasource-vip-ipaddress</td>
<td>datasource-vip-ipaddress</td>
<td>VIP IP address</td>
</tr>
<tr>
<td>repl-datasource-vip-netmask</td>
<td>datasource-vip-netmask</td>
<td>VIP netmask</td>
</tr>
<tr>
<td>repl-datasource-boot-script</td>
<td>datasource-boot-script</td>
<td>Database start script</td>
</tr>
<tr>
<td>repl-datasource-enable-ssl</td>
<td>datasource-enable-ssl</td>
<td>Enable SSL connection to DBMS server</td>
</tr>
<tr>
<td>repl-datasource-log-pattern</td>
<td>datasource-log-pattern</td>
<td>MySQL log filename pattern</td>
</tr>
<tr>
<td>repl-datasource-mysql-conf</td>
<td>datasource-mysql-conf</td>
<td>MySQL config file</td>
</tr>
<tr>
<td>repl-datasource-mysql-iblog-Directory</td>
<td>datasource-mysql-iblog-Directory</td>
<td>MySQL InnoDB log directory</td>
</tr>
<tr>
<td>repl-datasource-mysql-ibdata-Directory</td>
<td>datasource-mysql-ibdata-Directory</td>
<td>MySQL InnoDB data directory</td>
</tr>
<tr>
<td>repl-datasource-mysql-log-pattern</td>
<td>datasource-mysql-log-pattern</td>
<td>MySQL log filename pattern</td>
</tr>
<tr>
<td>repl-datasource-mysql-log-directory</td>
<td>datasource-mysql-log-directory</td>
<td>MySQL log directory</td>
</tr>
<tr>
<td>repl-datasource-mysql-ssl-capable</td>
<td>datasource-mysql-ssl-capable</td>
<td>Connector addresses that should receive session messages</td>
</tr>
<tr>
<td>repl-datasource-relay-enabled</td>
<td>datasource-relay-enabled</td>
<td>Make this dataservice the slave of another</td>
</tr>
<tr>
<td>repl-datasource-thl-port</td>
<td>datasource-thl-port</td>
<td>Port to use for THL operations</td>
</tr>
<tr>
<td>repl-datasource-use-relative-latency</td>
<td>datasource-use-relative-latency</td>
<td>Enable the cluster to operate on relative latency</td>
</tr>
<tr>
<td>repl-datasource-vip-enabled</td>
<td>datasource-vip-enabled</td>
<td>Is VIP management enabled?</td>
</tr>
<tr>
<td>repl-datasource-vip-ipaddress</td>
<td>datasource-vip-ipaddress</td>
<td>VIP IP address</td>
</tr>
<tr>
<td>repl-datasource-vip-netmask</td>
<td>datasource-vip-netmask</td>
<td>VIP netmask</td>
</tr>
<tr>
<td>repl-datasource-boot-script</td>
<td>datasource-boot-script</td>
<td>Database start script</td>
</tr>
<tr>
<td>repl-datasource-enable-ssl</td>
<td>datasource-enable-ssl</td>
<td>Enable SSL connection to DBMS server</td>
</tr>
<tr>
<td>repl-datasource-log-pattern</td>
<td>datasource-log-pattern</td>
<td>MySQL log filename pattern</td>
</tr>
<tr>
<td>repl-datasource-mysql-conf</td>
<td>datasource-mysql-conf</td>
<td>MySQL config file</td>
</tr>
<tr>
<td>repl-datasource-mysql-heartbeat</td>
<td>datasource-mysql-heartbeat</td>
<td>MySQL heartbeat data</td>
</tr>
<tr>
<td>repl-datasource-mysql-log-pattern</td>
<td>datasource-mysql-log-pattern</td>
<td>MySQL log filename pattern</td>
</tr>
<tr>
<td>repl-datasource-mysql-log-directory</td>
<td>datasource-mysql-log-directory</td>
<td>MySQL log directory</td>
</tr>
<tr>
<td>repl-datasource-mysql-ssl-capable</td>
<td>datasource-mysql-ssl-capable</td>
<td>Connector addresses that should receive session messages</td>
</tr>
<tr>
<td>repl-datasource-relay-enabled</td>
<td>datasource-relay-enabled</td>
<td>Make this dataservice the slave of another</td>
</tr>
<tr>
<td>repl-datasource-thl-port</td>
<td>datasource-thl-port</td>
<td>Port to use for THL operations</td>
</tr>
<tr>
<td>repl-datasource-use-relative-latency</td>
<td>datasource-use-relative-latency</td>
<td>Enable the cluster to operate on relative latency</td>
</tr>
<tr>
<td>repl-datasource-vip-enabled</td>
<td>datasource-vip-enabled</td>
<td>Is VIP management enabled?</td>
</tr>
<tr>
<td>repl-datasource-vip-ipaddress</td>
<td>datasource-vip-ipaddress</td>
<td>VIP IP address</td>
</tr>
<tr>
<td>repl-datasource-vip-netmask</td>
<td>datasource-vip-netmask</td>
<td>VIP netmask</td>
</tr>
<tr>
<td>repl-datasource-boot-script</td>
<td>datasource-boot-script</td>
<td>Database start script</td>
</tr>
<tr>
<td>repl-datasource-enable-ssl</td>
<td>datasource-enable-ssl</td>
<td>Enable SSL connection to DBMS server</td>
</tr>
<tr>
<td>repl-datasource-log-pattern</td>
<td>datasource-log-pattern</td>
<td>MySQL log filename pattern</td>
</tr>
<tr>
<td>repl-datasource-mysql-conf</td>
<td>datasource-mysql-conf</td>
<td>MySQL config file</td>
</tr>
<tr>
<td>repl-datasource-mysql-heartbeat</td>
<td>datasource-mysql-heartbeat</td>
<td>MySQL heartbeat data</td>
</tr>
<tr>
<td>repl-datasource-mysql-log-pattern</td>
<td>datasource-mysql-log-pattern</td>
<td>MySQL log filename pattern</td>
</tr>
<tr>
<td>repl-datasource-mysql-log-directory</td>
<td>datasource-mysql-log-directory</td>
<td>MySQL log directory</td>
</tr>
<tr>
<td>repl-datasource-mysql-ssl-capable</td>
<td>datasource-mysql-ssl-capable</td>
<td>Connector addresses that should receive session messages</td>
</tr>
<tr>
<td>repl-datasource-relay-enabled</td>
<td>datasource-relay-enabled</td>
<td>Make this dataservice the slave of another</td>
</tr>
<tr>
<td>repl-datasource-thl-port</td>
<td>datasource-thl-port</td>
<td>Port to use for THL operations</td>
</tr>
<tr>
<td>repl-datasource-use-relative-latency</td>
<td>datasource-use-relative-latency</td>
<td>Enable the cluster to operate on relative latency</td>
</tr>
<tr>
<td>repl-datasource-vip-enabled</td>
<td>datasource-vip-enabled</td>
<td>Is VIP management enabled?</td>
</tr>
<tr>
<td>repl-datasource-vip-ipaddress</td>
<td>datasource-vip-ipaddress</td>
<td>VIP IP address</td>
</tr>
<tr>
<td>repl-datasource-vip-netmask</td>
<td>datasource-vip-netmask</td>
<td>VIP netmask</td>
</tr>
<tr>
<td>repl-datasource-boot-script</td>
<td>datasource-boot-script</td>
<td>Database start script</td>
</tr>
<tr>
<td>repl-datasource-enable-ssl</td>
<td>datasource-enable-ssl</td>
<td>Enable SSL connection to DBMS server</td>
</tr>
<tr>
<td>repl-datasource-log-pattern</td>
<td>datasource-log-pattern</td>
<td>MySQL log filename pattern</td>
</tr>
<tr>
<td>repl-datasource-mysql-conf</td>
<td>datasource-mysql-conf</td>
<td>MySQL config file</td>
</tr>
<tr>
<td>repl-datasource-mysql-heartbeat</td>
<td>datasource-mysql-heartbeat</td>
<td>MySQL heartbeat data</td>
</tr>
</tbody>
</table>

The table above lists some of the options available in the *tpm* deployment command. Each option is described with a brief explanation.
## The `tpm` Deployment Command

<table>
<thead>
<tr>
<th>CmdLine Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-datasource-oracle-service [448], repl-datasource-oracle-service [448]</td>
<td>datasource-oracle-service [448], repl-datasource-oracle-service [448]</td>
<td>Oracle Service</td>
</tr>
<tr>
<td>-direct-datasource-user</td>
<td></td>
<td></td>
</tr>
<tr>
<td>repl-direct-datasource-user</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-datasource-oracle-service-group [448], repl-datasource-oracle-service-group [448]</td>
<td>datasource-oracle-service-group [448], repl-datasource-oracle-service-group [448]</td>
<td>Oracle service operating system group</td>
</tr>
<tr>
<td>-direct-datasource-host</td>
<td></td>
<td></td>
</tr>
<tr>
<td>repl-direct-datasource-host</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-datasource-oracle-type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>repl-datasource-oracle-type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>source-oracle-service</td>
<td></td>
<td>Oracle Service</td>
</tr>
<tr>
<td>repl-source-oracle-service</td>
<td></td>
<td></td>
</tr>
<tr>
<td>source-oracle-scan</td>
<td></td>
<td>Oracle SCAN</td>
</tr>
<tr>
<td>repl-source-oracle-scan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>source-oracle-sid</td>
<td></td>
<td>Oracle SID</td>
</tr>
<tr>
<td>repl-source-oracle-sid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-direct-datasource-oracle-service-group [450], repl-direct-datasource-oracle-service-group [450]</td>
<td>direct-datasource-oracle-service-group [450], repl-direct-datasource-oracle-service-group [450]</td>
<td>Oracle service operating system group</td>
</tr>
<tr>
<td>-direct-datasource-oracle-service-group [450], repl-direct-datasource-oracle-service-group [450]</td>
<td>direct-datasource-oracle-service-group [450], repl-direct-datasource-oracle-service-group [450]</td>
<td>Oracle service operating system group</td>
</tr>
<tr>
<td>--delete [449]</td>
<td>delete [449]</td>
<td>Delete the named data service from the configuration Data Service options:</td>
</tr>
<tr>
<td>--deploy-current-package [449]</td>
<td>deploy-current-package [449]</td>
<td>Deploy the current Tungsten package</td>
</tr>
<tr>
<td>--direct-datasource-password [451], repl-direct-datasource-password [451]</td>
<td>direct-datasource-password [451], repl-direct-datasource-password [451]</td>
<td>Password for datasource connection</td>
</tr>
<tr>
<td>--delete [451]</td>
<td>directory [451]</td>
<td>Set the directory of an existing installation used during fetching an existing configuration</td>
</tr>
</tbody>
</table>
The tpm Deployment Command

<table>
<thead>
<tr>
<th>CmdLine Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-disable-security-controls [451]</td>
<td>disable-security-controls [451]</td>
<td>Disables all forms of security, including SSL, TLS and authentication</td>
</tr>
<tr>
<td>-disable-slave-extractor [451], repl:--disable-slave-extractor [451]</td>
<td>disable-slave-extractor [451], repl:--disable-slave-extractor [451]</td>
<td>Should slave servers support the master role?</td>
</tr>
<tr>
<td>-drop-static-columns-in-updates [452]</td>
<td>drop-static-columns-in-updates [452]</td>
<td>This will modify UPDATE transactions in row-based replication and eliminate any columns that were not modified.</td>
</tr>
<tr>
<td>-enable-active-witnesses [452], repl:--enable-active-witnesses [452]</td>
<td>active-witnesses [452], enable-active-witnesses [452]</td>
<td>Enable active witness hosts</td>
</tr>
<tr>
<td>-enable-connector-client-ssl [452], --connector-client-ssl [452]</td>
<td>connector-client-ssl [452], enable-connector-client-ssl [452]</td>
<td>Enable SSL encryption of traffic from the client to the connector</td>
</tr>
<tr>
<td>-enable-connector-server-ssl [452], repl:--enable-connector-server-ssl [452]</td>
<td>connector-server-ssl [452], enable-connector-server-ssl [452]</td>
<td>Enable SSL encryption of traffic from the connector to the database</td>
</tr>
<tr>
<td>-enable-extracors [452], repl:--rmi-authentication [452]</td>
<td>enable-rmi-authentication [452], rmi-authentication [452]</td>
<td>Enable RMI authentication for the services running on this host</td>
</tr>
<tr>
<td>-enable-heterogeneous-master [452], ---enable-heterogeneous-service [453], --enable-heterogeneous-service [453]</td>
<td>enable-heterogeneous-master [452], enable-heterogeneous-master [452], enable-heterogeneous-service [453]</td>
<td>Enable heterogeneous operation for the master</td>
</tr>
<tr>
<td>-enable-heterogeneous-service [453]</td>
<td>enable-heterogeneous-service [453]</td>
<td>Enable heterogeneous operation</td>
</tr>
<tr>
<td>-enable-heterogeneous-service [453]</td>
<td>enable-heterogeneous-service [453]</td>
<td>Enable heterogeneous operation for the slave</td>
</tr>
<tr>
<td>-enable-heterogeneous-master [452], repl:--enable-heterogeneous-master [452], --enable-heterogeneous-master [452]</td>
<td>enable-heterogeneous-master [452]</td>
<td>Enable heterogeneous operation for the master</td>
</tr>
<tr>
<td>-enable-heterogeneous-service [453]</td>
<td>enable-heterogeneous-service [453]</td>
<td>Enable heterogeneous operation for the slave</td>
</tr>
<tr>
<td>-enable-rmi-authentication [454], repl:--rmi-authentication [454]</td>
<td>enable-rmi-authentication [454], rmi-authentication [454]</td>
<td>Enable SSL encryption of RMI communication on this host</td>
</tr>
<tr>
<td>-enable-sudo-access [454], repl:--root-command-prefix [454]</td>
<td>enable-sudo-access [454], root-command-prefix [454]</td>
<td>Run root commands using sudo</td>
</tr>
<tr>
<td>-enable-thl-listener [454], repl:--enable-thl-listener [454]</td>
<td>enable-thl-listener [454], repl:enable-thl-listener [454]</td>
<td>Should this service allow THL connections?</td>
</tr>
<tr>
<td>--host-name [455]</td>
<td>host-name [455]</td>
<td>DNS hostname</td>
</tr>
<tr>
<td>--hosts [455]</td>
<td>hosts [455]</td>
<td>Limit the command to the hosts listed You must use the hostname as it appears in the configuration.</td>
</tr>
<tr>
<td>--hub [455], --dataservice-hub-host [455]</td>
<td>dataservice-hub-host [455], hub [455]</td>
<td>What is the hub host for this all-masters dataservice?</td>
</tr>
</tbody>
</table>

The tpm Deployment Command

<table>
<thead>
<tr>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active-witnesses [452]</td>
<td>Enable active witness hosts</td>
</tr>
<tr>
<td>connector-client-ssl [452], enable-connector-client-ssl [452]</td>
<td>Enable SSL encryption of traffic from the client to the connector</td>
</tr>
<tr>
<td>connector-server-ssl [452], enable-connector-server-ssl [452]</td>
<td>Enable SSL encryption of traffic from the connector to the database</td>
</tr>
<tr>
<td>enable-rmi-authentication [452], rmi-authentication [452]</td>
<td>Enable SSL encryption of RMI communication on this host</td>
</tr>
<tr>
<td>enable-sudo-access [454], root-command-prefix [454]</td>
<td>Run root commands using sudo</td>
</tr>
<tr>
<td>enable-thl-listener [454], repl:enable-thl-listener [454]</td>
<td>Should this service allow THL connections?</td>
</tr>
<tr>
<td>host-name [455]</td>
<td>DNS hostname</td>
</tr>
<tr>
<td>hosts [455]</td>
<td>Limit the command to the hosts listed You must use the hostname as it appears in the configuration.</td>
</tr>
<tr>
<td>dataservice-hub-host [455], hub [455]</td>
<td>What is the hub host for this all-masters dataservice?</td>
</tr>
<tr>
<td>CmdLine Option</td>
<td>INI File Option</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>--hub-service [455], --dataservice-hub-service [455]</td>
<td>dataservice-hub-service [455], hub-service [455]</td>
</tr>
<tr>
<td>--install [455]</td>
<td>install [455]</td>
</tr>
<tr>
<td>--install-directory [456], --home-directory [456]</td>
<td>home-directory [456], install-directory [456]</td>
</tr>
<tr>
<td>--java-file-encoding [457], --repl-java-file-encoding [457]</td>
<td>java-file-encoding [457], repl-java-file-encoding [457]</td>
</tr>
<tr>
<td>--java-groups-key [457]</td>
<td>java-groups-key [457]</td>
</tr>
<tr>
<td>--java-jgroups-key [457]</td>
<td>java-jgroups-key [457]</td>
</tr>
<tr>
<td>--java-keystore-path [457]</td>
<td>java-keystore-path [457]</td>
</tr>
<tr>
<td>--java-keystore-path [457]</td>
<td>java-keystore-path [457]</td>
</tr>
<tr>
<td>--java-keystore-path [457]</td>
<td>java-keystore-path [457]</td>
</tr>
<tr>
<td>--java-keystore-path [457]</td>
<td>java-keystore-path [457]</td>
</tr>
<tr>
<td>--java-tls-key-lifetime [458]</td>
<td>java-tls-key-lifetime [458]</td>
</tr>
<tr>
<td>--java-truststore-password [458]</td>
<td>java-truststore-password [458]</td>
</tr>
<tr>
<td>--java-user-timezone [458], --repl-java-user-timezone [458]</td>
<td>java-user-timezone [458], repl-java-user-timezone [458]</td>
</tr>
<tr>
<td>--log [458]</td>
<td>log [458]</td>
</tr>
<tr>
<td>--log-slave-updates [459]</td>
<td>log-slave-updates [459]</td>
</tr>
<tr>
<td>--master [459], --datasevice-master-host [459], --mesters [459], --relay [459]</td>
<td>dataservice-master-host [459], master [459], masters [459], relay [459]</td>
</tr>
<tr>
<td>--master-services [459], --datasevice-master-services [459]</td>
<td>dataservice-master-services [459], master-services [459]</td>
</tr>
<tr>
<td>Description</td>
<td>INI File Option</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Hostnames for the dataservice members</td>
<td>dataser-vic-hosts [460], members [460]</td>
</tr>
<tr>
<td>Replicator metadata directory</td>
<td>metadata-directory [460], repl-metadata-directory [460]</td>
</tr>
<tr>
<td>Enable the Manager API</td>
<td>mgr-api [460]</td>
</tr>
<tr>
<td>Address for the Manager API</td>
<td>mgr-api-address [460]</td>
</tr>
<tr>
<td>Enable all Manager API commands. Only the status command will be enabled without it.</td>
<td>mgr-api-full-access [460]</td>
</tr>
<tr>
<td>Port for the Manager API</td>
<td>mgr-api-port [460]</td>
</tr>
<tr>
<td>Port to use for manager group communication</td>
<td>mgr-group-communication-port [460]</td>
</tr>
<tr>
<td>Java memory usage [MB] that will force a Manager restart</td>
<td>mgr-heap-threshold [460]</td>
</tr>
<tr>
<td>Listen interface to use for the manager</td>
<td>mgr-listen-interface [461]</td>
</tr>
<tr>
<td>Mechanism to use when identifying the liveness of other data-sources (ping, echo)</td>
<td>mgr-ping-method [461]</td>
</tr>
<tr>
<td>Manager policy mode</td>
<td>mgr-policy-mode [461]</td>
</tr>
<tr>
<td>Port to use for the manager RMI server</td>
<td>mgr-rmi-port [461]</td>
</tr>
<tr>
<td>Port to use for calling the remote manager RMI server</td>
<td>mgr-rmi-remote-port [461]</td>
</tr>
<tr>
<td>Make slaves read-only</td>
<td>mgr-ro-slave [462]</td>
</tr>
<tr>
<td>Path to the arp binary</td>
<td>mgr-vip-arp-path [462]</td>
</tr>
<tr>
<td>VIP network device</td>
<td>mgr-vip-device [462]</td>
</tr>
<tr>
<td>Path to the ifconfig binary</td>
<td>mgr-vip-ifconfig-path [462]</td>
</tr>
<tr>
<td>Wait for all datasources to be available before completing installation</td>
<td>mgr-wait-for-members [462]</td>
</tr>
<tr>
<td>For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility.</td>
<td>mysql-allow-intensive-checks [462]</td>
</tr>
<tr>
<td>Path to MySQL Connector/J</td>
<td>mysql-connectorj-path [462]</td>
</tr>
<tr>
<td>MySQL Driver Vendor</td>
<td>mysql-driver [463]</td>
</tr>
<tr>
<td>Enables ANSI_QUOTES mode for incoming events</td>
<td>mysql-enable-ansiquotes [463], repl-mysql-enable-ansiquotes [463]</td>
</tr>
<tr>
<td>Enable a filter to convert ENUM values to strings</td>
<td>mysql-enable-enumentstring [463], repl-mysql-enable-enumentstring [463]</td>
</tr>
<tr>
<td>Enables a filter to translate DELETE FROM ONLY to DELETE FROM and UPDATE ONLY to UPDATE.</td>
<td>mysql-enable-nonlykeywords [463], repl-mysql-enable-nonlykeywords [463]</td>
</tr>
<tr>
<td>Enable a filter to convert SET types to strings</td>
<td>mysql-enable-settostring [463], repl-mysql-enable-settostring [463]</td>
</tr>
<tr>
<td>Slaves are read-only</td>
<td>mysql-ro-slave [463], repl-mysql-ro-slave [463]</td>
</tr>
<tr>
<td>Explicitly set the MySQL server ID</td>
<td>mysql-server-id [464], repl-mysql-server-id [464]</td>
</tr>
<tr>
<td>Transfer strings as their byte representation</td>
<td>mysql-use-bytes-for-string [464], repl-mysql-use-bytes-for-string [464]</td>
</tr>
<tr>
<td>Directory to use for storing xtrabackup full &amp; incremental backups</td>
<td>mysql-xtrabackup-dir [464], repl-mysql-xtrabackup-dir [464]</td>
</tr>
</tbody>
</table>
The tpm Deployment Command

<table>
<thead>
<tr>
<th>CmdLine Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repl-native-slave-takeover</td>
<td>repl-native-slave-takeover</td>
<td>Takeover native replication</td>
</tr>
<tr>
<td>repl-standby</td>
<td>repl-standby</td>
<td>Cross-site replicators within a composite multimaster deployment can be configured to point to slaves by default, and to prefer slaves over masters during operation. In a standard deployment, cross-site replicators work via masters at each cluster site to read the remote information. To configure the service to use slaves in preference to masters, use the repl-native-slave-takeover option to tpm. Both masters and slaves remain in the list of possible hosts, so if no slaves are available during a switch or failover event, then a master will be used.</td>
</tr>
<tr>
<td>repl-postgresql-dname</td>
<td>repl-postgresql-dname</td>
<td>Name of the database to replicate</td>
</tr>
<tr>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>Enable MySQL to PostgreSQL DDL dialect converting filter placeholder</td>
</tr>
<tr>
<td>repl-postgresql-slonik</td>
<td>repl-postgresql-slonik</td>
<td>Path to the slonik executable</td>
</tr>
<tr>
<td>repl-postgresql-tables</td>
<td>repl-postgresql-tables</td>
<td>Tables to replicate in form: schema1.table1,schema2.table2,...</td>
</tr>
<tr>
<td>preferred-path</td>
<td>preferred-path</td>
<td>Additional command path</td>
</tr>
<tr>
<td>repl-native-slave-takeover</td>
<td>repl-native-slave-takeover</td>
<td>Takeover native replication</td>
</tr>
<tr>
<td>repl-standby</td>
<td>repl-standby</td>
<td>Cross-site replicators within a composite multimaster deployment can be configured to point to slaves by default, and to prefer slaves over masters during operation. In a standard deployment, cross-site replicators work via masters at each cluster site to read the remote information. To configure the service to use slaves in preference to masters, use the repl-native-slave-takeover option to tpm. Both masters and slaves remain in the list of possible hosts, so if no slaves are available during a switch or failover event, then a master will be used.</td>
</tr>
<tr>
<td>repl-postgresql-dname</td>
<td>repl-postgresql-dname</td>
<td>Name of the database to replicate</td>
</tr>
<tr>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>Enable MySQL to PostgreSQL DDL dialect converting filter placeholder</td>
</tr>
<tr>
<td>repl-postgresql-slonik</td>
<td>repl-postgresql-slonik</td>
<td>Path to the slonik executable</td>
</tr>
<tr>
<td>repl-postgresql-tables</td>
<td>repl-postgresql-tables</td>
<td>Tables to replicate in form: schema1.table1,schema2.table2,...</td>
</tr>
<tr>
<td>preferred-path</td>
<td>preferred-path</td>
<td>Additional command path</td>
</tr>
<tr>
<td>repl-native-slave-takeover</td>
<td>repl-native-slave-takeover</td>
<td>Takeover native replication</td>
</tr>
<tr>
<td>repl-standby</td>
<td>repl-standby</td>
<td>Cross-site replicators within a composite multimaster deployment can be configured to point to slaves by default, and to prefer slaves over masters during operation. In a standard deployment, cross-site replicators work via masters at each cluster site to read the remote information. To configure the service to use slaves in preference to masters, use the repl-native-slave-takeover option to tpm. Both masters and slaves remain in the list of possible hosts, so if no slaves are available during a switch or failover event, then a master will be used.</td>
</tr>
<tr>
<td>repl-postgresql-dname</td>
<td>repl-postgresql-dname</td>
<td>Name of the database to replicate</td>
</tr>
<tr>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>Enable MySQL to PostgreSQL DDL dialect converting filter placeholder</td>
</tr>
<tr>
<td>repl-postgresql-slonik</td>
<td>repl-postgresql-slonik</td>
<td>Path to the slonik executable</td>
</tr>
<tr>
<td>repl-postgresql-tables</td>
<td>repl-postgresql-tables</td>
<td>Tables to replicate in form: schema1.table1,schema2.table2,...</td>
</tr>
<tr>
<td>preferred-path</td>
<td>preferred-path</td>
<td>Additional command path</td>
</tr>
<tr>
<td>repl-native-slave-takeover</td>
<td>repl-native-slave-takeover</td>
<td>Takeover native replication</td>
</tr>
<tr>
<td>repl-standby</td>
<td>repl-standby</td>
<td>Cross-site replicators within a composite multimaster deployment can be configured to point to slaves by default, and to prefer slaves over masters during operation. In a standard deployment, cross-site replicators work via masters at each cluster site to read the remote information. To configure the service to use slaves in preference to masters, use the repl-native-slave-takeover option to tpm. Both masters and slaves remain in the list of possible hosts, so if no slaves are available during a switch or failover event, then a master will be used.</td>
</tr>
<tr>
<td>repl-postgresql-dname</td>
<td>repl-postgresql-dname</td>
<td>Name of the database to replicate</td>
</tr>
<tr>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>Enable MySQL to PostgreSQL DDL dialect converting filter placeholder</td>
</tr>
<tr>
<td>repl-postgresql-slonik</td>
<td>repl-postgresql-slonik</td>
<td>Path to the slonik executable</td>
</tr>
<tr>
<td>repl-postgresql-tables</td>
<td>repl-postgresql-tables</td>
<td>Tables to replicate in form: schema1.table1,schema2.table2,...</td>
</tr>
<tr>
<td>preferred-path</td>
<td>preferred-path</td>
<td>Additional command path</td>
</tr>
<tr>
<td>repl-native-slave-takeover</td>
<td>repl-native-slave-takeover</td>
<td>Takeover native replication</td>
</tr>
<tr>
<td>repl-standby</td>
<td>repl-standby</td>
<td>Cross-site replicators within a composite multimaster deployment can be configured to point to slaves by default, and to prefer slaves over masters during operation. In a standard deployment, cross-site replicators work via masters at each cluster site to read the remote information. To configure the service to use slaves in preference to masters, use the repl-native-slave-takeover option to tpm. Both masters and slaves remain in the list of possible hosts, so if no slaves are available during a switch or failover event, then a master will be used.</td>
</tr>
<tr>
<td>repl-postgresql-dname</td>
<td>repl-postgresql-dname</td>
<td>Name of the database to replicate</td>
</tr>
<tr>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>Enable MySQL to PostgreSQL DDL dialect converting filter placeholder</td>
</tr>
<tr>
<td>repl-postgresql-slonik</td>
<td>repl-postgresql-slonik</td>
<td>Path to the slonik executable</td>
</tr>
<tr>
<td>repl-postgresql-tables</td>
<td>repl-postgresql-tables</td>
<td>Tables to replicate in form: schema1.table1,schema2.table2,...</td>
</tr>
<tr>
<td>preferred-path</td>
<td>preferred-path</td>
<td>Additional command path</td>
</tr>
<tr>
<td>repl-native-slave-takeover</td>
<td>repl-native-slave-takeover</td>
<td>Takeover native replication</td>
</tr>
<tr>
<td>repl-standby</td>
<td>repl-standby</td>
<td>Cross-site replicators within a composite multimaster deployment can be configured to point to slaves by default, and to prefer slaves over masters during operation. In a standard deployment, cross-site replicators work via masters at each cluster site to read the remote information. To configure the service to use slaves in preference to masters, use the repl-native-slave-takeover option to tpm. Both masters and slaves remain in the list of possible hosts, so if no slaves are available during a switch or failover event, then a master will be used.</td>
</tr>
<tr>
<td>repl-postgresql-dname</td>
<td>repl-postgresql-dname</td>
<td>Name of the database to replicate</td>
</tr>
<tr>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>Enable MySQL to PostgreSQL DDL dialect converting filter placeholder</td>
</tr>
<tr>
<td>repl-postgresql-slonik</td>
<td>repl-postgresql-slonik</td>
<td>Path to the slonik executable</td>
</tr>
<tr>
<td>repl-postgresql-tables</td>
<td>repl-postgresql-tables</td>
<td>Tables to replicate in form: schema1.table1,schema2.table2,...</td>
</tr>
<tr>
<td>preferred-path</td>
<td>preferred-path</td>
<td>Additional command path</td>
</tr>
<tr>
<td>repl-native-slave-takeover</td>
<td>repl-native-slave-takeover</td>
<td>Takeover native replication</td>
</tr>
<tr>
<td>repl-standby</td>
<td>repl-standby</td>
<td>Cross-site replicators within a composite multimaster deployment can be configured to point to slaves by default, and to prefer slaves over masters during operation. In a standard deployment, cross-site replicators work via masters at each cluster site to read the remote information. To configure the service to use slaves in preference to masters, use the repl-native-slave-takeover option to tpm. Both masters and slaves remain in the list of possible hosts, so if no slaves are available during a switch or failover event, then a master will be used.</td>
</tr>
<tr>
<td>repl-postgresql-dname</td>
<td>repl-postgresql-dname</td>
<td>Name of the database to replicate</td>
</tr>
<tr>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>Enable MySQL to PostgreSQL DDL dialect converting filter placeholder</td>
</tr>
<tr>
<td>repl-postgresql-slonik</td>
<td>repl-postgresql-slonik</td>
<td>Path to the slonik executable</td>
</tr>
<tr>
<td>repl-postgresql-tables</td>
<td>repl-postgresql-tables</td>
<td>Tables to replicate in form: schema1.table1,schema2.table2,...</td>
</tr>
<tr>
<td>preferred-path</td>
<td>preferred-path</td>
<td>Additional command path</td>
</tr>
<tr>
<td>repl-native-slave-takeover</td>
<td>repl-native-slave-takeover</td>
<td>Takeover native replication</td>
</tr>
<tr>
<td>repl-standby</td>
<td>repl-standby</td>
<td>Cross-site replicators within a composite multimaster deployment can be configured to point to slaves by default, and to prefer slaves over masters during operation. In a standard deployment, cross-site replicators work via masters at each cluster site to read the remote information. To configure the service to use slaves in preference to masters, use the repl-native-slave-takeover option to tpm. Both masters and slaves remain in the list of possible hosts, so if no slaves are available during a switch or failover event, then a master will be used.</td>
</tr>
<tr>
<td>repl-postgresql-dname</td>
<td>repl-postgresql-dname</td>
<td>Name of the database to replicate</td>
</tr>
<tr>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>Enable MySQL to PostgreSQL DDL dialect converting filter placeholder</td>
</tr>
<tr>
<td>repl-postgresql-slonik</td>
<td>repl-postgresql-slonik</td>
<td>Path to the slonik executable</td>
</tr>
<tr>
<td>repl-postgresql-tables</td>
<td>repl-postgresql-tables</td>
<td>Tables to replicate in form: schema1.table1,schema2.table2,...</td>
</tr>
<tr>
<td>preferred-path</td>
<td>preferred-path</td>
<td>Additional command path</td>
</tr>
<tr>
<td>repl-native-slave-takeover</td>
<td>repl-native-slave-takeover</td>
<td>Takeover native replication</td>
</tr>
<tr>
<td>repl-standby</td>
<td>repl-standby</td>
<td>Cross-site replicators within a composite multimaster deployment can be configured to point to slaves by default, and to prefer slaves over masters during operation. In a standard deployment, cross-site replicators work via masters at each cluster site to read the remote information. To configure the service to use slaves in preference to masters, use the repl-native-slave-takeover option to tpm. Both masters and slaves remain in the list of possible hosts, so if no slaves are available during a switch or failover event, then a master will be used.</td>
</tr>
<tr>
<td>repl-postgresql-dname</td>
<td>repl-postgresql-dname</td>
<td>Name of the database to replicate</td>
</tr>
<tr>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>repl-postgresql-enable-mysql2pgddl</td>
<td>Enable MySQL to PostgreSQL DDL dialect converting filter placeholder</td>
</tr>
<tr>
<td>repl-postgresql-slonik</td>
<td>repl-postgresql-slonik</td>
<td>Path to the slonik executable</td>
</tr>
<tr>
<td>repl-postgresql-tables</td>
<td>repl-postgresql-tables</td>
<td>Tables to replicate in form: schema1.table1,schema2.table2,...</td>
</tr>
<tr>
<td>preferred-path</td>
<td>preferred-path</td>
<td>Additional command path</td>
</tr>
</tbody>
</table>
## The tm deployment command

### CmdLine Option

<table>
<thead>
<tr>
<th>Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--relay-source [469]</code></td>
<td><code>dataservice-relay-source [469]</code></td>
<td>Dataservice name to use as a relay source</td>
</tr>
<tr>
<td><code>--replace-tls-certificate [469]</code></td>
<td><code>replace-tls-certificate [469]</code></td>
<td>Replace the TLS certificate</td>
</tr>
<tr>
<td><code>--replication-host [469]</code></td>
<td><code>datasource-host [469]</code></td>
<td>Hostname of the datasource</td>
</tr>
<tr>
<td><code>--replication-password [469]</code></td>
<td><code>datasource-password [469]</code></td>
<td>Database password</td>
</tr>
<tr>
<td><code>--replication-port [469]</code></td>
<td><code>datasource-port [469]</code></td>
<td>Database network port</td>
</tr>
<tr>
<td><code>--replication-user [470]</code></td>
<td><code>datasource-user [470]</code></td>
<td>User for database connection</td>
</tr>
<tr>
<td><code>--reset [470]</code></td>
<td><code>reset [470]</code></td>
<td>Clear the current configuration before processing any arguments</td>
</tr>
<tr>
<td><code>--rmi-port [470], repl-rmi-port [470]</code></td>
<td><code>repl-rmi-port [470], rmi-port [470]</code></td>
<td>Replication RMI listen port</td>
</tr>
<tr>
<td><code>--rmi-user [470]</code></td>
<td><code>rmi-user [470]</code></td>
<td>The username for RMI authentication</td>
</tr>
<tr>
<td><code>--role [470], repl-role [470]</code></td>
<td><code>repl-role [470], role [470]</code></td>
<td>What is the replication role for this service?</td>
</tr>
<tr>
<td><code>--router-gateway-port [471]</code></td>
<td><code>router-gateway-port [471]</code></td>
<td>The router gateway port</td>
</tr>
<tr>
<td><code>--service-alias [471], repl-service-alias [471]</code></td>
<td><code>service-service-alias [471], repl-service-service-alias [471]</code></td>
<td>Replication alias of this dataservice</td>
</tr>
<tr>
<td><code>--service-name [471]</code></td>
<td><code>service-name [471]</code></td>
<td>Set the service name</td>
</tr>
<tr>
<td><code>--service-type [471], repl-service-type [471]</code></td>
<td><code>service-type [471], repl-service-type [471]</code></td>
<td>What is the replication service type?</td>
</tr>
<tr>
<td><code>--skip-statemap [472]</code></td>
<td><code>skip-statemap [472]</code></td>
<td>Do not copy the cluster-home/conf/statemap.properties from the previous install</td>
</tr>
<tr>
<td><code>--slave-privileged-updates [472]</code></td>
<td><code>slave-privileged-updates [472]</code></td>
<td>Does login for slave update have superuser privileges</td>
</tr>
<tr>
<td><code>--slaves [472], repl-slaves [472]</code></td>
<td><code>dataservice-slaves [472], slaves [472]</code></td>
<td>What are the slaves for this dataservice?</td>
</tr>
<tr>
<td><code>--start [472]</code></td>
<td><code>start [472]</code></td>
<td>Start the services after configuration</td>
</tr>
<tr>
<td><code>--start-and-report [472]</code></td>
<td><code>start-and-report [472]</code></td>
<td>Start the services and report out the status after configuration</td>
</tr>
<tr>
<td><code>--svc-allow-remote-service [472], repl-svc-allow-remote-service [472]</code></td>
<td><code>svc-allow-remote-service [472], repl-svc-allow-remote-service [472]</code></td>
<td>Replicate from any service</td>
</tr>
</tbody>
</table>

UPDATE does not affect any rows.

How should the replicator behave when a Row-Based Replication UPDATE does not affect any rows.
9.8.1 - tpm Options

`--connector-reset-when-affinity-back`
### Config File Options

#### connector-reset-when-affinity-back[435]

| Description | Forces reconnection of clients to datasources with the configured affinity when they become available |
| Default     | true |
| Valid Values | false | Allow connections to remain open even when correct datasource becomes available |
|             | true  | Disconnect clients when datasource becomes available |

When a site goes offline, connections to this site will be forced closed. Those connections will reconnect, as long as the site stays offline, they will be connected to remote site.

You can now enable an option so that when the site comes back online, the connector will disconnect all these connections that couldn't get to their preferred site so that they will then reconnect to the expected site with the appropriate affinity.

Note that this only applies to bridge mode. In proxy mode, relevancy of connected data source will be re-evaluated before every transaction.

When not enabled, connections will continue to use the server originally configured until they disconnect through normal attribution. This is the default option.

### 9.8.2. A tpm Options

#### --allow-bidi-unsafe

<table>
<thead>
<tr>
<th>Option</th>
<th>--allow-bidi-unsafe[436]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-allow-bidi-unsafe[436]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>allow-bidi-unsafe [436], repl-allow-bidi-unsafe [436]</td>
</tr>
<tr>
<td>Description</td>
<td>Allow unsafe SQL from remote service</td>
</tr>
<tr>
<td>Value Type</td>
<td>boolean</td>
</tr>
<tr>
<td>Valid Values</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>true</td>
</tr>
</tbody>
</table>

#### --api

<table>
<thead>
<tr>
<th>Option</th>
<th>--api [436]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-api [436]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>api [436], repl-api [436]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable the replication API</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --api-host

<table>
<thead>
<tr>
<th>Option</th>
<th>--api-host [436]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-api-host [436]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>api-host [436], repl-api-host [436]</td>
</tr>
<tr>
<td>Description</td>
<td>Hostname that the replication API should listen on</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --api-password

<table>
<thead>
<tr>
<th>Option</th>
<th>--api-password [436]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-api-password [436]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>api-password [436], repl-api-password [436]</td>
</tr>
<tr>
<td>Description</td>
<td>HTTP basic auth password for the replication API</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --api-port

<table>
<thead>
<tr>
<th>Option</th>
<th>--api-port</th>
</tr>
</thead>
</table>

436
## The tpm Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Config File Options</th>
<th>Aliases</th>
<th>Aliases</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--api-port</code></td>
<td>Port that the replication API should bind to</td>
<td>string</td>
<td><code>api-port</code>, <code>repl-api-port</code></td>
<td><code>--repl-api-port</code></td>
<td><code>--api-port</code></td>
</tr>
<tr>
<td><code>--api-user</code></td>
<td>HTTP basic auth username for the replication API</td>
<td>string</td>
<td><code>api-user</code>, <code>repl-api-user</code></td>
<td><code>--repl-api-user</code></td>
<td><code>--api-user</code></td>
</tr>
<tr>
<td><code>--application-password</code></td>
<td>Database password for the connector</td>
<td>string</td>
<td><code>application-password</code>, <code>connector-password</code></td>
<td><code>--connector-password</code></td>
<td><code>--application-password</code></td>
</tr>
<tr>
<td><code>--application-port</code></td>
<td>Port for the connector to listen on</td>
<td>string</td>
<td><code>application-port</code>, <code>connector-listen-port</code></td>
<td><code>--connector-listen-port</code></td>
<td><code>--application-port</code></td>
</tr>
<tr>
<td><code>--application-readonly-port</code></td>
<td>Port for the connector to listen for read-only connections on</td>
<td>string</td>
<td><code>application-readonly-port</code>, <code>connector-readonly-listen-port</code></td>
<td><code>--connector-readonly-listen-port</code></td>
<td><code>--application-readonly-port</code></td>
</tr>
<tr>
<td><code>--application-user</code></td>
<td>Database username for the connector</td>
<td>string</td>
<td><code>application-user</code>, <code>connector-user</code></td>
<td><code>--connector-user</code></td>
<td><code>--application-user</code></td>
</tr>
<tr>
<td><code>--auto-enable</code></td>
<td>Auto-enable services after start-up</td>
<td>string</td>
<td><code>auto-enable</code>, <code>repl-auto-enable</code></td>
<td><code>--repl-auto-enable</code></td>
<td><code>--auto-enable</code></td>
</tr>
</tbody>
</table>
The delay between the replicator identifying that autorecovery is needed, and autorecovery being attempted. For busy MySQL installations, larger numbers may be needed to allow time for MySQL servers to restart or recover from their failure.

--auto-recovery-max-attempts

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.

--auto-recovery-reset-interval

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.

### 9.8.3. B tpm Options

--backup-directory

Permanent backup storage directory

--backup-dump-directory

Offline backup directory
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--backup-method</code></td>
<td>Database backup method</td>
<td>string</td>
<td>ebs-snapshot, file-copy-snapshot, mysqlldump, none, script, xtrabackup, xtrabackup-full, xtrabackup-incremental</td>
</tr>
<tr>
<td><code>--backup-online</code></td>
<td>Does the backup script support backing up a datasource while it is ONLINE</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td><code>--backup-retention</code></td>
<td>Number of backups to retain</td>
<td>numeric</td>
<td></td>
</tr>
<tr>
<td><code>--backup-script</code></td>
<td>What is the path to the backup script</td>
<td>filename</td>
<td></td>
</tr>
<tr>
<td><code>--batch-enabled</code></td>
<td>Should the replicator service use a batch applier</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td><code>--batch-load-language</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Option: \(--\text{batch-load-language} \,[439]\)

**Config File Options**: \(\text{batch-load-language} \,[439]\)

**Description**: Which script language to use for batch loading

**Value Type**: string

**Valid Values**:
- js: JavaScript
- sql: SQL

### Option: \(--\text{batch-load-template} \,[440]\)

**Config File Options**: \(\text{batch-load-template} \,[440]\)

**Description**: Value for the loadBatchTemplate property

**Value Type**: string

### Option: \(--\text{buffer-size} \,[440]\)

**Aliases**: \(--\text{repl-buffer-size} \,[440]\), \(--\text{repl-svc-applier-buffer-size} \,[440]\), \(--\text{repl-svc-buffer-size} \,[440]\)

**Config File Options**: buffer-size \,[440], repl-buffer-size \,[440], repl-svc-applier-buffer-size \,[440], repl-svc-buffer-size \,[440]

**Description**: Replicator queue size between stages (min 1)

**Value Type**: numeric

### 9.8.4. C tpm Options

#### \(--\text{cctrl-column-width} \,[440]\)

**Config File Options**: cctrl-column-width \,[440]

**Description**: Sets the minimum column width for service names within cctrl

Sets the minimum column width for service names within cctrl

#### \(--\text{channels} \,[440]\)

**Aliases**: \(--\text{repl-channels} \,[440]\)

**Config File Options**: channels \,[440], repl-channels \,[440]

**Description**: Number of replication channels to use for services

**Value Type**: numeric

#### \(--\text{cluster-slave-auto-recovery-delay-interval} \,[440]\)

**Aliases**: \(--\text{cluster-slave-repl-auto-recovery-delay-interval} \,[440]\)

**Config File Options**: cluster-slave-auto-recovery-delay-interval \,[440], cluster-slave-repl-auto-recovery-delay-interval \,[440]

**Description**: Default value for \(--\text{auto-recovery-delay-interval} \,[440]\) when \(--\text{topology=cluster-slave} \,[440]\)

**Value Type**: string

#### \(--\text{cluster-slave-auto-recovery-max-attempts} \,[440]\)

**Aliases**: \(--\text{cluster-slave-repl-auto-recovery-max-attempts} \,[440]\)

**Config File Options**: cluster-slave-auto-recovery-max-attempts \,[440], cluster-slave-repl-auto-recovery-max-attempts \,[440]

**Description**: Default value for \(--\text{auto-recovery-max-attempts} \,[440]\) when \(--\text{topology=cluster-slave} \,[440]\)
### `--cluster-slave-auto-recovery-reset-interval`

**Option** 
`--cluster-slave-auto-recovery-reset-interval [441]`

**Aliases** 
`--cluster-slave-repl-auto-recovery-reset-interval [441]`

**Config File Options** 
`cluster-slave-auto-recovery-reset-interval [441], cluster-slave-repl-auto-recovery-reset-interval [441]`

**Description** 
Default value for `--auto-recovery-reset-interval` when `--topology=cluster-slave`

**Value Type** 
string

### `--composite-datasources`

**Option** 
`--composite-datasources [441]`

**Aliases** 
`--dataservice-composite-datasources [441]`

**Config File Options** 
`composite-datasources [441], dataservice-composite-datasources [441]`

**Description** 
Data services that should be added to this composite data service

**Value Type** 
string

### `--config-file-help`

**Option** 
`--config-file-help [441]`

**Config File Options** 
`config-file-help [441]`

**Description** 
Display help information for content of the config file

**Value Type** 
string

### `--conn-java-enable-concurrent-gc`

**Option** 
`--conn-java-enable-concurrent-gc [441]`

**Config File Options** 
`conn-java-enable-concurrent-gc [441]`

**Description** 
Connector Java uses concurrent garbage collection

**Value Type** 
string

### `--conn-java-mem-size`

**Option** 
`--conn-java-mem-size [441]`

**Config File Options** 
`conn-java-mem-size [441]`

**Description** 
Connector Java heap memory size used to buffer data between clients and databases

**Value Type** 
numeric

**Valid Values**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<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.

### `--conn-round-robin-include-master`

**Option** 
`--conn-round-robin-include-master [441]`

**Config File Options** 
`conn-round-robin-include-master [441]`

**Description** 
Should the Connector include the master in round-robin load balancing

**Value Type** 
string

### `--connector-affinity`

**Option** 
`--connector-affinity [441]`

**Config File Options** 
`connector-affinity [441]`

**Description** 
The default affinity for all connections
The `tpm` Deployment Command

<table>
<thead>
<tr>
<th>Value Type</th>
<th>string</th>
</tr>
</thead>
</table>

**--connector-autoreconnect**

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--connector-autoreconnect [442]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td><code>connector-autoreconnect [442]</code></td>
</tr>
<tr>
<td>Description</td>
<td>Enable auto-reconnect in the connector</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--connector-autoreconnect-killed-connections**

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--connector-autoreconnect-killed-connections [442]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td><code>connector-autoreconnect-killed-connections [442]</code></td>
</tr>
<tr>
<td>Description</td>
<td>Enable autoreconnect for connections killed within the connector</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td>false Do not retry killed connections</td>
</tr>
<tr>
<td></td>
<td>true</td>
</tr>
</tbody>
</table>

By default, the connector operates as follows:

- Reconnect closed connections
- Retry autocommitted reads

The behavior can be modified by using the `--connector-autoreconnect-killed-connections [442]`. Setting to `false` disables the reconnection or retry of a connection outside of a planned switch or automatic failover. The default is `true`, reconnecting and retrying all connections.

**--connector-bridge-mode**

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--connector-bridge-mode [442]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>--enable-connector-bridge-mode [442]</code></td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>connector-bridge-mode [442], enable-connector-bridge-mode [442]</code></td>
</tr>
<tr>
<td>Description</td>
<td>Enable the Tungsten Connector bridge mode</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--connector-default-schema**

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--connector-default-schema [442]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>--connector-forced-schema [442]</code></td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>connector-default-schema [442], connector-forced-schema [442]</code></td>
</tr>
<tr>
<td>Description</td>
<td>Default schema for the connector to use</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--connector-delete-user-map**

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--connector-delete-user-map [442]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td><code>connector-delete-user-map [442]</code></td>
</tr>
<tr>
<td>Description</td>
<td>Overwrite an existing user.map file</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--connector-disable-connection-warnings**

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--connector-disable-connection-warnings [442]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td><code>connector-disable-connection-warnings [442]</code></td>
</tr>
<tr>
<td>Description</td>
<td>Hide Connector warnings in log files</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>
## The tpm Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--connector-disconnect-timeout</code></td>
<td><code>connector-disconnect-timeout</code></td>
<td>Time (in seconds) to wait for active connection to disconnect before forcing them closed [default: 5]</td>
<td>boolean</td>
</tr>
<tr>
<td><code>--connector-drop-after-max-connections</code></td>
<td><code>connector-drop-after-max-connections</code></td>
<td>Instantly drop connections that arrive after <code>--connector-max-connections</code> has been reached</td>
<td>boolean</td>
</tr>
<tr>
<td><code>--connector-keepalive-timeout</code></td>
<td><code>connector-keepalive-timeout</code></td>
<td>The Connector keep-alive <code>SELECT</code> statement is submitted if the time since the last activity reaches this timeout value.</td>
<td>numeric</td>
</tr>
<tr>
<td><code>--connector-listen-interface</code></td>
<td><code>connector-listen-interface</code></td>
<td>Listen interface to use for the connector</td>
<td>string</td>
</tr>
<tr>
<td><code>--connector-max-connections</code></td>
<td><code>connector-max-connections</code></td>
<td>The maximum number of connections the connector should allow at any time</td>
<td>numeric</td>
</tr>
<tr>
<td><code>--connector-max-slave-latency</code></td>
<td><code>connector-max-applied-latency</code>, <code>connector-max-slave-latency</code></td>
<td>The maximum applied latency for slave connections</td>
<td>string</td>
</tr>
<tr>
<td><code>--connector-readonly</code></td>
<td><code>connector-readonly</code>, <code>enable-connector-readonly</code></td>
<td>Enable the Tungsten Connector read-only mode</td>
<td>string</td>
</tr>
<tr>
<td><code>--connector-ro-addresses</code></td>
<td><code>connector-ro-addresses</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## The tpm Deployment Command

### Description
Connector addresses that should receive a r/o connection

| Value Type | string |

### --connector-rw-addresses

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-rw-addresses [444]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>connector-rw-addresses [444]</td>
</tr>
<tr>
<td>Description</td>
<td>Connector addresses that should receive a r/w connection</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### --connector-rwsplitting

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-rwsplitting [444]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>connector-rwsplitting [444]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable DirectReads R/W splitting in the connector</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### --connector-smartscale

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-smartscale [444]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>connector-smartscale [444]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable SmartScale R/W splitting in the connector</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### --connector-smartscale-sessionid

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-smartscale-sessionid [444]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>connector-smartscale-sessionid [444]</td>
</tr>
<tr>
<td>Description</td>
<td>The default session ID to use with smart scale</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### --connector-ssl-capable

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-ssl-capable [444]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>connector-ssl-capable [444]</td>
</tr>
<tr>
<td>Description</td>
<td>Defines whether connector ports are advertised as being capable of SSL</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>false</td>
</tr>
<tr>
<td></td>
<td>true</td>
</tr>
</tbody>
</table>

When SSL is enabled, the Connector automatically advertises the ports and itself as SSL capable. With some clients, this triggers them to use SSL even if SSL has not been configured. This causes the connections to fail and not operate correctly.

### --connectors

<table>
<thead>
<tr>
<th>Option</th>
<th>--connectors [444]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--dataservice-connectors [444]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>connectors [444], dataservice-connectors [444]</td>
</tr>
<tr>
<td>Description</td>
<td>Hostnames for the dataservice connectors</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### --consistency-policy

| Option | --consistency-policy [444] |

---

444
### The tpm Deployment Command

<table>
<thead>
<tr>
<th>Aliases</th>
<th>--repl-consistency-policy [444]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>consistency-policy [444], repl-consistency-policy [444]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Should the replicator stop or warn if a consistency check fails?</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

### 9.8.5. D tpm Options

#### --dataservice-name

<table>
<thead>
<tr>
<th>Option</th>
<th>--dataservice-name [445]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>dataservice-name [445]</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>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

#### --dataservice-relay-enabled

<table>
<thead>
<tr>
<th>Option</th>
<th>--dataservice-relay-enabled [445]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>dataservice-relay-enabled [445]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Make this dataservice the slave of another</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

#### --dataservice-schema

<table>
<thead>
<tr>
<th>Option</th>
<th>--dataservice-schema [445]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>dataservice-schema [445]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>The db schema to hold dataservice details</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

#### --dataservice-thl-port

<table>
<thead>
<tr>
<th>Option</th>
<th>--dataservice-thl-port [445]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>dataservice-thl-port [445]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Port to use for THL operations</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

#### --dataservice-use-relative-latency

<table>
<thead>
<tr>
<th>Option</th>
<th>--dataservice-use-relative-latency [445]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--use-relative-latency [445]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>dataservice-use-relative-latency [445], use-relative-latency [445]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Enable the cluster to operate on relative latency</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

#### --dataservice-vip-enabled

<table>
<thead>
<tr>
<th>Option</th>
<th>--dataservice-vip-enabled [445]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>dataservice-vip-enabled [445]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Is VIP management enabled?</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

#### --dataservice-vip-ipaddress

<table>
<thead>
<tr>
<th>Option</th>
<th>--dataservice-vip-ipaddress [445]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>dataservice-vip-ipaddress [445]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td></td>
</tr>
</tbody>
</table>
## Config File Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Aliases</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--dataservice-vip-ipaddress</code></td>
<td>VIP IP address</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td><code>--dataservice-vip-netmask</code></td>
<td>VIP netmask</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td><code>--datasource-boot-script</code></td>
<td>Database start script</td>
<td>string</td>
<td><code>--repl-datasource-boot-script</code></td>
</tr>
<tr>
<td><code>--datasource-enable-ssl</code></td>
<td>Enable SSL connection to DBMS server</td>
<td>string</td>
<td><code>--repl-datasource-enable-ssl</code></td>
</tr>
<tr>
<td><code>--datasource-log-directory</code></td>
<td>Master log directory</td>
<td>string</td>
<td><code>--repl-datasource-log-directory</code></td>
</tr>
<tr>
<td><code>--datasource-log-pattern</code></td>
<td>Master log filename pattern</td>
<td>string</td>
<td><code>--repl-datasource-log-pattern</code></td>
</tr>
<tr>
<td><code>--datasource-mysql-conf</code></td>
<td>MySQL config file</td>
<td>string</td>
<td><code>--repl-datasource-mysql-conf</code></td>
</tr>
<tr>
<td><code>--datasource-mysql-data-directory</code></td>
<td>MySQL data directory</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>
### The tpms Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--datasource-mysql-data-directory</td>
<td>string</td>
</tr>
<tr>
<td>--datasource-mysql-ibdata-directory</td>
<td>string</td>
</tr>
<tr>
<td>--datasource-mysql-iblog-directory</td>
<td>string</td>
</tr>
<tr>
<td>--datasource-mysql-ssl-ca</td>
<td>string</td>
</tr>
<tr>
<td>--datasource-mysql-ssl-cert</td>
<td>string</td>
</tr>
<tr>
<td>--datasource-mysql-ssl-key</td>
<td>string</td>
</tr>
<tr>
<td>--datasource-oracle-scan</td>
<td>string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL data directory</td>
<td>string</td>
</tr>
<tr>
<td>MySQL InnoDB data directory</td>
<td>string</td>
</tr>
<tr>
<td>MySQL InnoDB log directory</td>
<td>string</td>
</tr>
<tr>
<td>MySQL SSL CA file</td>
<td>string</td>
</tr>
<tr>
<td>MySQL SSL certificate file</td>
<td>string</td>
</tr>
<tr>
<td>MySQL SSL key file</td>
<td>string</td>
</tr>
<tr>
<td>Oracle SCAN</td>
<td>string</td>
</tr>
</tbody>
</table>
The tpm Deployment Command

---

### --datasource-oracle-service

**Option**  
--datasource-oracle-service [448]

**Aliases**  
--repl-datasource-oracle-service [448]

**Config File Options**  
datasource-oracle-service [448], repl-datasource-oracle-service [448]

**Description**  
Oracle Service

**Value Type**  
string

---

### --datasource-oracle-service-group

**Option**  
--datasource-oracle-service-group [448]

**Aliases**  
--repl-datasource-oracle-service-group [448]

**Config File Options**  
 datasource-oracle-service-group [448], repl-datasource-oracle-service-group [448]

**Description**  
Oracle service operating system group

**Value Type**  
string

The name of the operating system group that Oracle users should be a member of. If this value cannot be determined or is not set properly, the Tungsten Clustering will be unable to access the necessary Oracle tools.

---

### --datasource-pg-archive

**Option**  
--datasource-pg-archive [448]

**Aliases**  
--repl-datasource-pg-archive [448]

**Config File Options**  
 datasource-pg-archive [448], repl-datasource-pg-archive [448]

**Description**  
PostgreSQL archive location

**Value Type**  
string

---

### --datasource-pg-conf

**Option**  
--datasource-pg-conf [448]

**Aliases**  
--repl-datasource-pg-conf [448]

**Config File Options**  
 datasource-pg-conf [448], repl-datasource-pg-conf [448]

**Description**  
Location of postgresql.conf

**Value Type**  
string

---

### --datasource-pg-home

**Option**  
--datasource-pg-home [448]

**Aliases**  
--repl-datasource-pg-home [448]

**Config File Options**  
 datasource-pg-home [448], repl-datasource-pg-home [448]

**Description**  
PostgreSQL data directory

**Value Type**  
string

---

### --datasource-pg-root

**Option**  
--datasource-pg-root [448]

**Aliases**  
--repl-datasource-pg-root [448]

**Config File Options**  
 datasource-pg-root [448], repl-datasource-pg-root [448]

**Description**  
Root directory for postgresql installation

**Value Type**  
string

---

### --datasource-systemctl-service

**Option**  
--datasource-systemctl-service [448]
The tpm Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>--datasource-type [449]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-datasource-type [449]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>datasource-type [449], repl-datasource-type [449]</td>
</tr>
<tr>
<td>Description</td>
<td>Database type</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Default</td>
<td>mysql</td>
</tr>
<tr>
<td>Valid Values</td>
<td>file, hdfs, kafka, mongodb, mysql, oracle, postgres, vertica</td>
</tr>
</tbody>
</table>

Specifies the command name or full path of the command that should be used to control the database service, including startup, shutdown and restart. This is used by the Tungsten Clustering to control the underlying database service. By default, this will be configured to the service according to your environment if it has been found during installation. For example, the services command, or `/etc/init.d/mysql`.

--delete

<table>
<thead>
<tr>
<th>Option</th>
<th>--delete [449]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>delete [449]</td>
</tr>
<tr>
<td>Description</td>
<td>Delete the named data service from the configuration Data Service options:</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--deploy-current-package

<table>
<thead>
<tr>
<th>Option</th>
<th>--deploy-current-package [449]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>deploy-current-package [449]</td>
</tr>
<tr>
<td>Description</td>
<td>Deploy the current Tungsten package</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--deploy-package-uri

<table>
<thead>
<tr>
<th>Option</th>
<th>--deploy-package-uri [449]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>deploy-package-uri [449]</td>
</tr>
<tr>
<td>Description</td>
<td>URL for the Tungsten package to deploy</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 [449]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-direct-datasource-log-directory [449]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>direct-datasource-log-directory [449], repl-direct-datasource-log-directory [449]</td>
</tr>
<tr>
<td>Description</td>
<td>Master log directory</td>
</tr>
<tr>
<td>Value Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>string</td>
<td>Master log filename pattern</td>
</tr>
<tr>
<td>string</td>
<td>Oracle SCAN</td>
</tr>
<tr>
<td>string</td>
<td>Oracle Service</td>
</tr>
<tr>
<td>string</td>
<td>Oracle SID</td>
</tr>
<tr>
<td>string</td>
<td>Database type</td>
</tr>
<tr>
<td>mysql</td>
<td></td>
</tr>
<tr>
<td>mysql</td>
<td></td>
</tr>
<tr>
<td>mysql</td>
<td></td>
</tr>
<tr>
<td>mysql</td>
<td></td>
</tr>
<tr>
<td>mysql</td>
<td></td>
</tr>
<tr>
<td>mysql</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Aliases</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--direct-datasource-log-pattern</td>
<td>--direct-datasource-log-pattern, repl-direct-datasource-log-pattern</td>
<td>Master log filename pattern</td>
</tr>
<tr>
<td>--direct-datasource-oracle-scan</td>
<td>--direct-datasource-oracle-scan, repl-direct-datasource-oracle-scan</td>
<td>Oracle SCAN</td>
</tr>
<tr>
<td>--direct-datasource-oracle-service</td>
<td>--direct-datasource-oracle-service, repl-direct-datasource-oracle-service</td>
<td>Oracle Service</td>
</tr>
<tr>
<td>--direct-datasource-oracle-sid</td>
<td>--direct-datasource-oracle-sid, repl-direct-datasource-oracle-sid</td>
<td>Oracle SID</td>
</tr>
<tr>
<td>--direct-datasource-type</td>
<td>--direct-datasource-type, repl-direct-datasource-type</td>
<td>Database type</td>
</tr>
<tr>
<td>--direct-replication-host</td>
<td>--direct-replication-host</td>
<td></td>
</tr>
</tbody>
</table>
### The tpm Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--direct-datasource-host</code></td>
<td>Database server hostname</td>
<td>string</td>
</tr>
<tr>
<td><code>--direct-replication-host</code></td>
<td></td>
<td>string</td>
</tr>
<tr>
<td><code>--repl-direct-datasource-host</code></td>
<td></td>
<td>string</td>
</tr>
<tr>
<td><code>--direct-datasource-password</code></td>
<td>Password for datasource connection</td>
<td>string</td>
</tr>
<tr>
<td><code>--direct-replication-password</code></td>
<td></td>
<td>string</td>
</tr>
<tr>
<td><code>--direct-datasource-port</code></td>
<td>Database server port</td>
<td>string</td>
</tr>
<tr>
<td><code>--direct-replication-port</code></td>
<td></td>
<td>string</td>
</tr>
<tr>
<td><code>--direct-datasource-user</code></td>
<td>Database login for Tungsten</td>
<td>string</td>
</tr>
<tr>
<td><code>--direct-replication-user</code></td>
<td></td>
<td>string</td>
</tr>
<tr>
<td><code>--directory</code></td>
<td>Set the directory of an existing installation used during fetching an existing configuration</td>
<td>string</td>
</tr>
<tr>
<td><code>--disable-relay-logs</code></td>
<td>Disable the use of relay-logs?</td>
<td>string</td>
</tr>
<tr>
<td><code>--disable-security-controls</code></td>
<td>Disables all forms of security, including SSL, TLS and authentication</td>
<td>string</td>
</tr>
<tr>
<td><code>--disable-slave-extractor</code></td>
<td></td>
<td>string</td>
</tr>
</tbody>
</table>
### 9.8.6. `tpm` Options

#### `--drop-static-columns-in-updates`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--drop-static-columns-in-updates</code> [452]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>drop-static-columns-in-updates</code> [452]</td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>drop-static-columns-in-updates</code> [452]</td>
</tr>
<tr>
<td>Description</td>
<td>This will modify UPDATE transactions in row-based replication and eliminate any columns that were not modified.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### `--enable-active-witnesses`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--enable-active-witnesses</code> [452]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>--active-witnesses</code> [452]</td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>active-witnesses</code> [452], <code>enable-active-witnesses</code> [452]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable active witness hosts</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### `--enable-connector-client-ssl`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--enable-connector-client-ssl</code> [452]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>--connector-client-ssl</code> [452]</td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>connector-client-ssl</code> [452], <code>enable-connector-client-ssl</code> [452]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable SSL encryption of traffic from the client to the connector</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### `--enable-connector-server-ssl`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--enable-connector-server-ssl</code> [452]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>--connector-server-ssl</code> [452]</td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>connector-server-ssl</code> [452], <code>enable-connector-server-ssl</code> [452]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable SSL encryption of traffic from the connector to the database</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### `--enable-connector-ssl`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--enable-connector-ssl</code> [452]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>--connector-ssl</code> [452]</td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>connector-ssl</code> [452], <code>enable-connector-ssl</code> [452]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable SSL encryption of connector traffic to the database</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### `--enable-heterogeneous-master`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--enable-heterogeneous-master</code> [452]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>--enable-heterogenous-master</code> [452]</td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>enable-heterogeneous-master</code> [452], <code>enable-heterogenous-master</code> [452]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable heterogeneous operation for the master</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>
### --enable-heterogeneous-service

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-heterogeneous-service [453]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--enable-heterogeneous-service [453]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>enable-heterogeneous-service [453], enable-heterogenous-service [453]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable heterogeneous operation</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

- **On a Master**
  - `--mysql-use-bytes-for-string [464]` 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 [464]` is set to true.
  - `pkey` filter is enabled (in the `q-to-dbms` stage).

### --enable-heterogeneous-slave

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-heterogeneous-slave [453]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--enable-heterogeneous-slave [453]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>enable-heterogeneous-slave [453], enable-heterogenous-slave [453]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable heterogeneous operation for the slave</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### --enable-heterogenous-master

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-heterogenous-master [452]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>enable-heterogenous-master [452]</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 [453]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>enable-heterogenous-service [453]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable heterogenous operation</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### --enable-heterogenous-slave

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-heterogenous-slave [453]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>enable-heterogenous-slave [453]</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-jgroups-ssl

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-jgroups-ssl [453]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--jgroups-ssl [453]</td>
</tr>
</tbody>
</table>
The `tpm Deployment Command`

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>enable-jgroups-ssl [453], jgroups-ssl [453]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Enable SSL encryption of JGroups communication on this host</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 [454]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--rmi-authentication [454]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>enable-rmi-authentication [454], rmi-authentication [454]</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 [454]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--rmi-ssl [454]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>enable-rmi-ssl [454], rmi-ssl [454]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable SSL encryption of RMI communication on this host</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--enable-slave-thl-listener

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-slave-thl-listener [454]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-enable-slave-thl-listener [454]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>enable-slave-thl-listener [454], repl-enable-slave-thl-listener [454]</td>
</tr>
<tr>
<td>Description</td>
<td>Should this service allow THL connections?</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--enable-sudo-access

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-sudo-access [454]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--root-command-prefix [454]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>enable-sudo-access [454], root-command-prefix [454]</td>
</tr>
<tr>
<td>Description</td>
<td>Run root commands using sudo</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--enable-thl-ssl

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-thl-ssl [454]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-enable-thl-ssl [454], --thl-ssl [454]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>enable-thl-ssl [454], repl-enable-thl-ssl [454], thl-ssl [454]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable SSL encryption of THL communication for this service</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--executable-prefix

<table>
<thead>
<tr>
<th>Option</th>
<th>--executable-prefix [454]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>executable-prefix [454]</td>
</tr>
<tr>
<td>Description</td>
<td>Adds a prefix to command aliases</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

When enabled, the supplied prefix is added to each command alias that is generated for a given installation. This enables multiple installations to co-exist and be accessible through a unique alias. For example, if the executable prefix is configured as `east`, then an alias for the installation to `trepctl` will be created as `east_trepctl`.
Alias information for executable prefix data is stored within the $CONTINUENT_ROOT/share/aliases.sh file for each installation.

9.8.7. Tpm Options

--file-protection-level

<table>
<thead>
<tr>
<th>Option</th>
<th>--file-protection-level [455]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>file-protection-level [455]</td>
</tr>
<tr>
<td>Description</td>
<td>Protection level for Continuent files</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--file-protection-umask

<table>
<thead>
<tr>
<th>Option</th>
<th>--file-protection-umask [455]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>file-protection-umask [455]</td>
</tr>
<tr>
<td>Description</td>
<td>Protection umask for Continuent files</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

9.8.8. Htpm Options

--host-name

<table>
<thead>
<tr>
<th>Option</th>
<th>--host-name [455]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>host-name [455]</td>
</tr>
<tr>
<td>Description</td>
<td>DNS hostname</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--hosts

<table>
<thead>
<tr>
<th>Option</th>
<th>--hosts [455]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>hosts [455]</td>
</tr>
<tr>
<td>Description</td>
<td>Limit the command to the hosts listed You must use the hostname as it appears in the configuration.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--hub

<table>
<thead>
<tr>
<th>Option</th>
<th>--hub [455]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--dataservice-hub-host [455]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>dataservice-hub-host [455], hub [455]</td>
</tr>
<tr>
<td>Description</td>
<td>What is the hub host for this all-masters dataservice?</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--hub-service

<table>
<thead>
<tr>
<th>Option</th>
<th>--hub-service [455]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--dataservice-hub-service [455]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>dataservice-hub-service [455], hub-service [455]</td>
</tr>
<tr>
<td>Description</td>
<td>The data service to use for the hub of a star topology</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

9.8.9. Itpm Options

--install

<table>
<thead>
<tr>
<th>Option</th>
<th>--install [455]</th>
</tr>
</thead>
</table>
The tpm Deployment Command

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--install [455]</td>
<td>Install service start scripts</td>
<td>string</td>
</tr>
</tbody>
</table>

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

9.8.10. J tpm Options

--java-connector-keystore-password

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--java-connector-keystore-password [456]</td>
<td>The password for unlocking the tungsten_connector_keystore.jks file in the security directory</td>
<td>string</td>
</tr>
</tbody>
</table>

--java-connector-keystore-path

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--java-connector-keystore-path [456]</td>
<td>Local path to the Java Connector Keystore file.</td>
<td>filename</td>
</tr>
</tbody>
</table>

--java-connector-truststore-password

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--java-connector-truststore-password [456]</td>
<td>The password for unlocking the tungsten_connector_truststore.jks file in the security directory</td>
<td>string</td>
</tr>
</tbody>
</table>

--java-connector-truststore-path

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--java-connector-truststore-path [456]</td>
<td>Local path to the Java Connector Truststore file.</td>
<td>filename</td>
</tr>
</tbody>
</table>

--java-enable-concurrent-gc

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--java-enable-concurrent-gc [456]</td>
<td>Replicator Java uses concurrent garbage collection</td>
<td>string</td>
</tr>
</tbody>
</table>

--java-external-lib-dir

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--java-external-lib-dir [456]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### The `tpm` Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--java-external-lib-dir</code></td>
<td>Directory for 3rd party Jar files required by replicator</td>
<td>string</td>
</tr>
<tr>
<td><code>--java-file-encoding</code></td>
<td>Java platform charset (esp. for heterogeneous replication)</td>
<td>string</td>
</tr>
<tr>
<td><code>--java-jgroups-key</code></td>
<td>The alias to use for the JGroups TLS key in the keystore.</td>
<td>string</td>
</tr>
<tr>
<td><code>--java-jgroups-keystore-path</code></td>
<td>Local path to the JGroups Java Keystore file.</td>
<td>filename</td>
</tr>
<tr>
<td><code>--java-jmxremote-access-path</code></td>
<td>Local path to the Java JMX Remote Access file.</td>
<td>filename</td>
</tr>
<tr>
<td><code>--java-keystore-password</code></td>
<td>The password for unlocking the tungsten_keystore.jks file in the security directory</td>
<td>string</td>
</tr>
<tr>
<td><code>--java-keystore-path</code></td>
<td>Local path to the Java Keystore file.</td>
<td>filename</td>
</tr>
<tr>
<td><code>--java-mem-size</code></td>
<td>Replicator Java heap memory size in Mb [min 128]</td>
<td>string</td>
</tr>
</tbody>
</table>
### Value Type

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--java-passwordstore-path</td>
<td>Local path to the Java Password Store file.</td>
</tr>
<tr>
<td>--java-tls-alias</td>
<td>The alias to use for the TLS key/certificate in the keystore and truststore.</td>
</tr>
<tr>
<td>--java-tls-key-lifetime</td>
<td>Lifetime for the Java TLS key</td>
</tr>
<tr>
<td>--java-tls-keystore-path</td>
<td>The keystore holding a certificate to use for all Continuent TLS encryption.</td>
</tr>
<tr>
<td>--java-truststore-password</td>
<td>The password for unlocking the tungsten_truststore.jks file in the security directory</td>
</tr>
<tr>
<td>--java-truststore-path</td>
<td>Local path to the Java Truststore file.</td>
</tr>
<tr>
<td>--java-user-timezone</td>
<td>Java VM Timezone (esp. for cross-site replication)</td>
</tr>
</tbody>
</table>

### Log Options

- `--log`
## The `tpm` Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--log</code></td>
<td><code>log</code></td>
<td>Write all messages, visible and hidden, to this file. You may specify a filename, 'pid' or 'timestamp'.</td>
<td>numeric</td>
</tr>
</tbody>
</table>

### `--log-slave-updates` Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--log-slave-updates</code></td>
<td><code>log-slave-updates</code></td>
<td>Should slaves log updates to binlog</td>
<td>string</td>
</tr>
</tbody>
</table>

### 9.8.12. `tpm` Options

#### `--master` Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--master</code></td>
<td><code>master</code></td>
<td>Hostname of the master host within this service</td>
<td>string</td>
</tr>
</tbody>
</table>

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

#### `--master-preferred-role` Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--master-preferred-role</code></td>
<td><code>master-preferred-role</code></td>
<td>Preferred role for master THL when connecting as a slave</td>
<td>string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Valid Values</th>
<th>Master role</th>
<th>Slave role</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>Master role</td>
<td>Master role</td>
</tr>
<tr>
<td>slave</td>
<td>Slave role</td>
<td>Slave role</td>
</tr>
</tbody>
</table>

#### `--master-services` Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--master-services</code></td>
<td><code>master-services</code></td>
<td>Data service names that should be used on each master</td>
<td>string</td>
</tr>
</tbody>
</table>

#### `--master-thl-host` Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--master-thl-host</code></td>
<td><code>master-thl-host</code></td>
<td>Master THL Hostname</td>
<td>string</td>
</tr>
</tbody>
</table>

#### `--master-thl-port` Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--master-thl-port</code></td>
<td><code>master-thl-port</code></td>
<td>Master THL Port</td>
<td>string</td>
</tr>
</tbody>
</table>
## The tpm Deployment Command

<table>
<thead>
<tr>
<th>Description</th>
<th>Master THL Port</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

### --members

<table>
<thead>
<tr>
<th>Option</th>
<th>--members [460]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td><strong>datasetservice-hosts [460]</strong></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><strong>datasetservice-hosts [460], members [460]</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Hostnames for the dataservice members</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

### --metadata-directory

<table>
<thead>
<tr>
<th>Option</th>
<th>--metadata-directory [460]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td><strong>repl-metadata-directory [460]</strong></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><strong>metadata-directory [460], repl-metadata-directory [460]</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Replicator metadata directory</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

### --mgr-api

<table>
<thead>
<tr>
<th>Option</th>
<th>--mgr-api [460]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td><strong>mgr-api [460]</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Enable the Manager API</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

### --mgr-api-address

<table>
<thead>
<tr>
<th>Option</th>
<th>--mgr-api-address [460]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td><strong>mgr-api-address [460]</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Address for the Manager API</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

### --mgr-api-full-access

<table>
<thead>
<tr>
<th>Option</th>
<th>--mgr-api-full-access [460]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td><strong>mgr-api-full-access [460]</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Enable all Manager API commands. Only the status command will be enabled without it.</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

### --mgr-api-port

<table>
<thead>
<tr>
<th>Option</th>
<th>--mgr-api-port [460]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td><strong>mgr-api-port [460]</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Port for the Manager API</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

### --mgr-group-communication-port

<table>
<thead>
<tr>
<th>Option</th>
<th>--mgr-group-communication-port [460]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td><strong>mgr-group-communication-port [460]</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Port to use for manager group communication</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

### --mgr-heap-threshold
### The `tpm` Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--mgr-heap-threshold</code></td>
<td>Java memory usage (MB) that will force a Manager restart</td>
<td>string</td>
</tr>
<tr>
<td><code>--mgr-java-enable-concurrent-gc</code></td>
<td>Manager Java uses concurrent garbage collection</td>
<td>string</td>
</tr>
<tr>
<td><code>--mgr-java-mem-size</code></td>
<td>Manager Java heap memory size in Mb (min 128)</td>
<td>numeric</td>
</tr>
<tr>
<td><code>--mgr-listen-interface</code></td>
<td>Listen interface to use for the manager</td>
<td>string</td>
</tr>
<tr>
<td><code>--mgr-ping-method</code></td>
<td>Mechanism to use when identifying the liveness of other datasources (ping, echo)</td>
<td>string</td>
</tr>
<tr>
<td><code>--mgr-policy-mode</code></td>
<td>Manager policy mode</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>Valid Values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>automatic</td>
<td>Automatic policy mode</td>
</tr>
<tr>
<td></td>
<td>maintenance</td>
<td>Maintenance policy mode</td>
</tr>
<tr>
<td></td>
<td>manual</td>
<td>Manual policy mode</td>
</tr>
<tr>
<td><code>--mgr-rmi-port</code></td>
<td>Port to use for the manager RMI server</td>
<td>string</td>
</tr>
<tr>
<td><code>--mgr-rmi-remote-port</code></td>
<td>Port to use for the manager RMI server</td>
<td>string</td>
</tr>
</tbody>
</table>
## The tpm Deployment Command

<table>
<thead>
<tr>
<th>Description</th>
<th>Port to use for calling the remote manager RMI server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### --mgr-ro-slave

<table>
<thead>
<tr>
<th>Option</th>
<th>--mgr-ro-slave [462]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mgr-ro-slave [462]</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 [462]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mgr-vip-arp-path [462]</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 [462]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mgr-vip-device [462]</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 [462]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mgr-vip-ifconfig-path [462]</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 [462]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mgr-wait-for-members [462]</td>
</tr>
<tr>
<td>Description</td>
<td>Wait for all datasources to be available before completing installation</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### --mysql-allow-intensive-checks

<table>
<thead>
<tr>
<th>Option</th>
<th>--mysql-allow-intensive-checks [462]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mysql-allow-intensive-checks [462]</td>
</tr>
<tr>
<td>Description</td>
<td>For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility. This includes checking each table definition individually for any unsupported data types.</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 [462]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mysql-connectorj-path [462]</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>
The tpm Deployment Command

Important

As of Tungsten Clustering v4.0.0, the MySQL Connector/J prerequisite has been removed. The JDBC interface now uses the Drizzle driver by default.

Note

tpm reverse will display the parameter --mysql-connectorj-path as long as any mysql-connector-java* file remains in /opt/continuent/share/

Warning

Do not use path /opt/continuent/share/ inside the value for --mysql-connectorj-path or tpm will abort with an error.

--mysql-driver

<table>
<thead>
<tr>
<th>Option</th>
<th>--mysql-driver [463]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mysql-driver [463]</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 [463]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>repl-mysql-enable-ansiquotes [463]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>mysql-enable-ansiquotes [463], repl-mysql-enable-ansiquotes [463]</td>
</tr>
<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-enumtostring

<table>
<thead>
<tr>
<th>Option</th>
<th>--mysql-enable-enumtostring [463]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>repl-mysql-enable-enumtostring [463]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>mysql-enable-enumtostring [463], repl-mysql-enable-enumtostring [463]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable a filter to convert ENUM values to strings</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 [463]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>repl-mysql-enable-noonlykeywords [463]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>mysql-enable-noonlykeywords [463], repl-mysql-enable-noonlykeywords [463]</td>
</tr>
<tr>
<td>Description</td>
<td>Enables a filter to translate DELETE FROM ONLY to DELETE FROM and UPDATE ONLY to 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 [463]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>repl-mysql-enable-settostring [463]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>mysql-enable-settostring [463], repl-mysql-enable-settostring [463]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable a filter to convert SET types to strings</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 [463]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>repl-mysql-ro-slave [463]</td>
</tr>
</tbody>
</table>
### The tpm Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--mysql-ro-slave</td>
<td>Slaves are read-only?</td>
<td>string</td>
</tr>
</tbody>
</table>

---

#### --mysql-server-id

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--mysql-server-id</td>
<td>Explicitly set the MySQL server ID</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>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--mysql-use-bytes-for-string</td>
<td>Transfer strings as their byte representation?</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --mysql-xtrabackup-dir

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--mysql-xtrabackup-dir</td>
<td>Directory to use for storing xtrabackup full &amp; incremental backups</td>
<td>string</td>
</tr>
</tbody>
</table>

9.8.13. N tpm Options

#### --native-slave-takeover

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--native-slave-takeover</td>
<td>Takeover native replication</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --no-deployment

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--no-deployment</td>
<td>Skip deployment steps that create the install directory</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --no-validation

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--no-validation</td>
<td>Skip validation checks that run on each host</td>
<td>string</td>
</tr>
</tbody>
</table>
9.8.14. `tpm` Options

```
--optimize-row-events
```

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--optimize-row-events</code> [465]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td><code>optimize-row-events</code> [465]</td>
</tr>
<tr>
<td>Description</td>
<td>Enables or disables optimized row updates</td>
</tr>
<tr>
<td>Value Type</td>
<td>boolean</td>
</tr>
<tr>
<td>Valid Values</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>true</td>
</tr>
</tbody>
</table>

Optimized row updates bundle multiple row-based updates into a single `INSERT` or `UPDATE` statement. This increases the throughput of large batches of row-based updates.

9.8.15. `tpm` Options

```
--pg-archive-timeout
```

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--pg-archive-timeout</code> [465]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>--repl-pg-archive-timeout</code> [465]</td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>pg-archive-timeout</code> [465], <code>repl-pg-archive-timeout</code> [465]</td>
</tr>
<tr>
<td>Description</td>
<td>Timeout for sending unfilled WAL buffers (data loss window)</td>
</tr>
<tr>
<td>Value Type</td>
<td>numeric</td>
</tr>
</tbody>
</table>

```
--pg-ctl
```

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--pg-ctl</code> [465]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>--repl-pg-ctl</code> [465]</td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>pg-ctl</code> [465], <code>repl-pg-ctl</code> [465]</td>
</tr>
<tr>
<td>Description</td>
<td>Path to the pg_ctl script</td>
</tr>
<tr>
<td>Value Type</td>
<td>filename</td>
</tr>
</tbody>
</table>

```
--pg-method
```

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--pg-method</code> [465]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>--repl-pg-method</code> [465]</td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>pg-method</code> [465], <code>repl-pg-method</code> [465]</td>
</tr>
<tr>
<td>Description</td>
<td>Postgres Replication method</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

```
--pg-standby
```

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--pg-standby</code> [465]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>--repl-pg-standby</code> [465]</td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>pg-standby</code> [465], <code>repl-pg-standby</code> [465]</td>
</tr>
<tr>
<td>Description</td>
<td>Path to the pg_standby script</td>
</tr>
<tr>
<td>Value Type</td>
<td>filename</td>
</tr>
</tbody>
</table>

```
--policy-relay-from-slave
```

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--policy-relay-from-slave</code> [465]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td><code>policy-relay-from-slave</code> [465]</td>
</tr>
<tr>
<td>Description</td>
<td>Cross-site replicators within a composite multimaster deployment can be configured to point to slaves by default, and to prefer slaves over masters during operation. In a standard deployment, cross-site replicators work via masters at each cluster site to read the remote information. To configure the service to use slaves in preference to</td>
</tr>
</tbody>
</table>
The tpm Deployment Command

masters, use the `--policy-relay-from-slave=true` option to tpm. Both masters and slaves remain in the list of possible hosts, so if no slaves are available during a switch or failover event, then a master will be used.

Cross-site replicators within a composite multimaster deployment can be configured to point to slaves by default, and to prefer slaves over masters during operation. In a standard deployment, cross-site replicators work via masters at each cluster site to read the remote information. To configure the service to use slaves in preference to masters, use the `--policy-relay-from-slave=true` option to tpm. Both masters and slaves remain in the list of possible hosts, so if no slaves are available during a switch or failover event, then a master will be used.

`--postgresql-dbname`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--postgresql-dbname</code></td>
<td>Name of the database to replicate</td>
</tr>
<tr>
<td><code>--repl-postgresql-dbname</code></td>
<td></td>
</tr>
</tbody>
</table>

| Config File Options         | postgresql-dbname [466], repl-postgresql-dbname [466] |
| Value Type                  | string                                                 |

`--postgresql-enable-mysql2pgddl`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--postgresql-enable-mysql2pgddl</code></td>
<td>Enable MySQL to PostgreSQL DDL dialect converting filter placeholder</td>
</tr>
<tr>
<td><code>--repl-postgresql-enable-mysql2pgddl</code></td>
<td></td>
</tr>
</tbody>
</table>

| Config File Options         | postgresql-enable-mysql2pgddl [466], repl-postgresql-enable-mysql2pgddl [466] |
| Value Type                  | string                                                 |
| Valid Values                | false                                                  |

`--postgresql-slonik`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--postgresql-slonik</code></td>
<td>Path to the slonik executable</td>
</tr>
<tr>
<td><code>--repl-postgresql-slonik</code></td>
<td></td>
</tr>
</tbody>
</table>

| Config File Options         | postgresql-slonik [466], repl-postgresql-slonik [466] |
| Value Type                  | filename                                                |

`--postgresql-tables`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--postgresql-tables</code></td>
<td>Tables to replicate in form: schemat1.table1,schema2.table2,...</td>
</tr>
<tr>
<td><code>--repl-postgresql-tables</code></td>
<td></td>
</tr>
</tbody>
</table>

| Config File Options         | postgresql-tables [466], repl-postgresql-tables [466] |
| Value Type                  | string                                                 |

`--preferred-path`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--preferred-path</code></td>
<td>Additional command path</td>
</tr>
</tbody>
</table>

| Config File Options         | preferred-path [466]                    |
| Value Type                  | filename                                |

Specifies one or more additional directories that will be added before the current PATH environment variable when external commands are run from within the backup environment. This affects all external tools used by Tungsten Clustering, including MySQL, Ruby, Java, and backup/restore tools such as Percona Xtrabackup.

One or more paths can be specified by separating each directory with a colon. For example:

```
shell> tpm ... --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.
## The tpm Deployment Command

### --prefetch-enabled

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>--prefetch-enabled</td>
<td>prefetch-enabled [467]</td>
<td>Should the replicator service be setup as a prefetch applier</td>
<td>string</td>
<td></td>
<td></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>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>--prefetch-max-time-ahead</td>
<td>prefetch-max-time-ahead [467]</td>
<td>Maximum number of seconds that the prefetch applier can get in front of the standard applier</td>
<td>numeric</td>
<td></td>
<td></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>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>--prefetch-min-time-ahead</td>
<td>prefetch-min-time-ahead [467]</td>
<td>Minimum number of seconds that the prefetch applier must be in front of the standard applier</td>
<td>numeric</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### --prefetch-schema

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>--prefetch-schema</td>
<td>prefetch-schema [467]</td>
<td>Schema to watch for timing prefetch progress</td>
<td>string</td>
<td>tungsten_</td>
<td>tungsten_</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>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>--prefetch-sleep-time</td>
<td>prefetch-sleep-time [467]</td>
<td>How long to wait when the prefetch applier gets too far ahead</td>
<td>string</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### --privileged-master

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>--privileged-master</td>
<td>privileged-master [467]</td>
<td>Does the login for the master database service have superuser privileges</td>
<td>string</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### --privileged-slave

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>--privileged-slave</td>
<td>privileged-slave [467]</td>
<td>Does the login for the slave database service have superuser privileges</td>
<td>string</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### --profile-script

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
</table>
### Config File Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>profile-script</code></td>
<td>Append commands to include env.sh in this profile script</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

#### --protect-configuration-files

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>protect-configuration-files</code></td>
<td>When enabled, configuration files are protected to be only readable and updateable by the configured user</td>
<td>string</td>
<td>false, true</td>
</tr>
</tbody>
</table>

When enabled (default), the configuration that contain user, password and other information are configured so that they are only readable by the configured user. For example:

```shell
cat /opt/continuent/tungsten/tungsten-replicator/conf/ | head
```

When disabled, the files are readable by all users:

```shell
cat /opt/continuent/tungsten/tungsten-replicator/conf/ | head
```

### 9.8.16. R tpm Options

#### --redshift-dbnme

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>redshift-dbnme</code></td>
<td>Name of the Redshift database to replicate into</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

#### --relay-directory

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>relay-directory</code></td>
<td></td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>
### The tpm Deployment Command

<table>
<thead>
<tr>
<th>Description</th>
<th>Directory for logs transferred from the master</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
<tr>
<td><strong>Default</strong></td>
<td>{home directory}/relay</td>
</tr>
<tr>
<td><strong>Valid Values</strong></td>
<td>{home directory}/relay</td>
</tr>
</tbody>
</table>

#### --relay-enabled

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th>--relay-enabled [469]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>relay-enabled [469]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Should the replicator service be setup as a relay master</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

#### --relay-source

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th>--relay-source [469]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--dataservice-relay-source [469], --master-dataservice [469]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>dataservice-relay-source [469], master-dataservice [469], relay-source [469]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Dataservice name to use as a relay source</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

#### --replace-tls-certificate

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th>--replace-tls-certificate [469]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>replace-tls-certificate [469]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Replace the TLS certificate</td>
</tr>
</tbody>
</table>

#### --replication-host

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th>--replication-host [469]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--datasource-host [469], --repl-datasource-host [469]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>datasource-host [469], repl-datasource-host [469], replication-host [469]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Hostname of the datasource</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

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

#### --replication-password

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th>--replication-password [469]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--datasource-password [469], --repl-datasource-password [469]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>datasource-password [469], repl-datasource-password [469], replication-password [469]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Database password</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

The password to be used when connecting to the database using the corresponding --replication-user [470].

#### --replication-port

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th>--replication-port [469]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--datasource-port [469], --repl-datasource-port [469]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>datasource-port [469], repl-datasource-port [469], replication-port [469]</td>
</tr>
<tr>
<td>Description</td>
<td>Database network port</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td>1521: Oracle Default</td>
</tr>
<tr>
<td></td>
<td>27017: Elasticsearch Default</td>
</tr>
<tr>
<td></td>
<td>27017: Kafka Default</td>
</tr>
<tr>
<td></td>
<td>27017: MongoDB Default</td>
</tr>
<tr>
<td></td>
<td>3306: MySQL Default</td>
</tr>
<tr>
<td></td>
<td>5432: PostgreSQL Default</td>
</tr>
<tr>
<td></td>
<td>5433: Vertica Default</td>
</tr>
<tr>
<td></td>
<td>5439: Redshift Default</td>
</tr>
<tr>
<td></td>
<td>8020: HDFS Default</td>
</tr>
</tbody>
</table>

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

```
--replication-user
```

<table>
<thead>
<tr>
<th>Description</th>
<th>User for database connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

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

```
--reset
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Clear the current configuration before processing any arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

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

```
--rmi-port
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Replication RMI listen port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

```
--rmi-user
```

<table>
<thead>
<tr>
<th>Description</th>
<th>The username for RMI authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

```
--role
```

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>
### Description

**What is the replication role for this service?**

<table>
<thead>
<tr>
<th>Value Type</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid Values</td>
<td>master, relay, slave</td>
</tr>
</tbody>
</table>

#### --router-gateway-port

<table>
<thead>
<tr>
<th>Option</th>
<th>--router-gateway-port [471]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>router-gateway-port [471]</td>
</tr>
<tr>
<td>Description</td>
<td>The router gateway port</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --router-jmx-port

<table>
<thead>
<tr>
<th>Option</th>
<th>--router-jmx-port [471]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>router-jmx-port [471]</td>
</tr>
<tr>
<td>Description</td>
<td>The router jmx port</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### 9.8.17. S tpm Options

#### --security-directory

<table>
<thead>
<tr>
<th>Option</th>
<th>--security-directory [471]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>security-directory [471]</td>
</tr>
<tr>
<td>Description</td>
<td>Storage directory for the Java security/encryption files</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --service-alias

<table>
<thead>
<tr>
<th>Option</th>
<th>--service-alias [471]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--dataservice-service-alias [471]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>dataservice-service-alias [471], service-alias [471]</td>
</tr>
<tr>
<td>Description</td>
<td>Replication alias of this dataservice</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --service-name

<table>
<thead>
<tr>
<th>Option</th>
<th>--service-name [471]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>service-name [471]</td>
</tr>
<tr>
<td>Description</td>
<td>Set the service name</td>
</tr>
</tbody>
</table>

Set the service name

#### --service-type

<table>
<thead>
<tr>
<th>Option</th>
<th>--service-type [471]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-service-type [471]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-service-type [471], service-type [471]</td>
</tr>
<tr>
<td>Description</td>
<td>What is the replication service type?</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td>local, remote</td>
</tr>
</tbody>
</table>
### The tpm Deployment Command

#### --skip-statemap

<table>
<thead>
<tr>
<th>Option</th>
<th>--skip-statemap [472]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>skip-statemap [472]</td>
</tr>
<tr>
<td>Description</td>
<td>Do not copy the cluster-home/conf/statemap.properties from the previous install</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --slave-privileged-updates

<table>
<thead>
<tr>
<th>Option</th>
<th>--slave-privileged-updates [472]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>slave-privileged-updates [472]</td>
</tr>
<tr>
<td>Description</td>
<td>Does login for slave update have superuser privileges</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --slaves

<table>
<thead>
<tr>
<th>Option</th>
<th>--slaves [472]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--dataservice-slaves [472]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>dataservice-slaves [472], slaves [472]</td>
</tr>
<tr>
<td>Description</td>
<td>What are the slaves for this dataservice?</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --start

<table>
<thead>
<tr>
<th>Option</th>
<th>--start [472]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>start [472]</td>
</tr>
<tr>
<td>Description</td>
<td>Start the services after configuration</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --start-and-report

<table>
<thead>
<tr>
<th>Option</th>
<th>--start-and-report [472]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>start-and-report [472]</td>
</tr>
<tr>
<td>Description</td>
<td>Start the services and report out the status after configuration</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --svc-allow-any-remote-service

<table>
<thead>
<tr>
<th>Option</th>
<th>--svc-allow-any-remote-service [472]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>repl-svc-allow-any-remote-service [472]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-svc-allow-any-remote-service [472], svc-allow-any-remote-service [472]</td>
</tr>
<tr>
<td>Description</td>
<td>Replicate from any service</td>
</tr>
<tr>
<td>Value Type</td>
<td>boolean</td>
</tr>
<tr>
<td>Valid Values</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>true</td>
</tr>
</tbody>
</table>

#### --svc-applier-block-commit-interval

<table>
<thead>
<tr>
<th>Option</th>
<th>--svc-applier-block-commit-interval [472]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>repl-svc-applier-block-commit-interval [472]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-svc-applier-block-commit-interval [472], svc-applier-block-commit-interval [472]</td>
</tr>
<tr>
<td>Description</td>
<td>Minimum interval between commits</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>
## TPM Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Valid Values</th>
<th>Aliases</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--svc-applier-block-commit-size</code></td>
<td>Applier block commit size (min 1)</td>
<td>numeric</td>
<td>disk, memory, none</td>
<td><code>--repl-svc-applier-block-commit-size</code></td>
</tr>
<tr>
<td><code>--svc-applier-filters</code></td>
<td>Replication service applier filters</td>
<td>string</td>
<td></td>
<td><code>--repl-svc-applier-filters</code></td>
</tr>
<tr>
<td><code>--svc-extractor-filters</code></td>
<td>Replication service extractor filters</td>
<td>string</td>
<td></td>
<td><code>--repl-svc-extractor-filters</code></td>
</tr>
<tr>
<td><code>--svc-fail-on-zero-row-update</code></td>
<td>How should the replicator behave when a Row-Based Replication UPDATE does not affect any rows.</td>
<td>string</td>
<td></td>
<td><code>--repl-svc-fail-on-zero-row-update</code></td>
</tr>
<tr>
<td><code>--svc-parallelization-type</code></td>
<td>Method for implementing parallel apply</td>
<td>string</td>
<td>disk, memory, none</td>
<td><code>--repl-svc-parallelization-type</code></td>
</tr>
<tr>
<td><code>--svc-remote-filters</code></td>
<td></td>
<td></td>
<td></td>
<td><code>--repl-svc-remote-filters</code></td>
</tr>
</tbody>
</table>
The tpm Deployment Command

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>repl-svc-remote-filters [473], svc-remote-filters [473]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Replication service remote download filters</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

```
--svc-reposition-on-source-id-change
```

<table>
<thead>
<tr>
<th>Option</th>
<th>--svc-reposition-on-source-id-change [474]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-svc-reposition-on-source-id-change [474]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-svc-reposition-on-source-id-change [474], svc-reposition-on-source-id-change [474]</td>
</tr>
<tr>
<td>Description</td>
<td>The master will come ONLINE from the current position if the stored source_id does not match the value in the static properties</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

```
--svc-shard-default-db
```

<table>
<thead>
<tr>
<th>Option</th>
<th>--svc-shard-default-db [474]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-svc-shard-default-db [474]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-svc-shard-default-db [474], svc-shard-default-db [474]</td>
</tr>
<tr>
<td>Description</td>
<td>Mode for setting the shard ID from the default db</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td>relaxed</td>
</tr>
<tr>
<td></td>
<td>stringent</td>
</tr>
</tbody>
</table>

```
--svc-table-engine
```

<table>
<thead>
<tr>
<th>Option</th>
<th>--svc-table-engine [474]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-svc-table-engine [474]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-svc-table-engine [474], svc-table-engine [474]</td>
</tr>
<tr>
<td>Description</td>
<td>Replication service table engine</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Default</td>
<td>innodb</td>
</tr>
<tr>
<td>Valid Values</td>
<td>innodb</td>
</tr>
<tr>
<td></td>
<td>innodb</td>
</tr>
</tbody>
</table>

```
--svc-thl-filters
```

<table>
<thead>
<tr>
<th>Option</th>
<th>--svc-thl-filters [474]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-svc-thl-filters [474]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-svc-thl-filters [474], svc-thl-filters [474]</td>
</tr>
<tr>
<td>Description</td>
<td>Replication service THL filters</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

9.8.18. T tpm Options

```
--target-dataservice
```

<table>
<thead>
<tr>
<th>Option</th>
<th>--target-dataservice [474]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--slave-dataservice [474]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>slave-dataservice [474], target-dataservice [474]</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
```

474
The `tpm` Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--temp-directory</code></td>
<td>Temporary Directory</td>
<td>string</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Config File Options**
- `temp-directory`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--template-file-help</code></td>
<td>Display the keys that may be used in configuration template files</td>
<td>string</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Config File Options**
- `template-file-help`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--template-search-path</code></td>
<td>Adds a new template search path for configuration file generation</td>
<td>filename</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Config File Options**
- `template-search-path`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--thl-directory</code></td>
<td>Replicator log directory</td>
<td>string</td>
<td><code>{home directory}/thl</code></td>
<td><code>{home directory}/thl</code></td>
</tr>
</tbody>
</table>

**Config File Options**
- `repl-thl-directory`
- `thl-directory`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--thl-do-checksum</code></td>
<td>Execute checksum operations on THL log files</td>
<td>string</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Config File Options**
- `repl-thl-do-checksum`
- `thl-do-checksum`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--thl-interface</code></td>
<td>Listen interface to use for THL operations</td>
<td>string</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Config File Options**
- `repl-thl-interface`
- `thl-interface`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--thl-log-connection-timeout</code></td>
<td>Number of seconds to wait for a connection to the THL log</td>
<td>numeric</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Config File Options**
- `repl-thl-log-connection-timeout`
- `thl-log-connection-timeout`
## The tpm Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Aliasases</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Valid Values</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>#d Number of days</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>#h Number of hours</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>#m Number of minutes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>#s Number of seconds</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7d 7 days</td>
<td></td>
</tr>
<tr>
<td>--thl-port</td>
<td>--repl-thl-port [476]</td>
<td>repl-thl-port [476], thl-port [476]</td>
<td>Port to use for THL Operations</td>
<td>string</td>
</tr>
<tr>
<td>--thl-protocol</td>
<td>--repl-thl-protocol [476]</td>
<td>repl-thl-protocol [476], thl-protocol [476]</td>
<td>Protocol to use for THL communication with this service</td>
<td>string</td>
</tr>
<tr>
<td>--topology</td>
<td>--dataservice-topology [476]</td>
<td>dataservice-topology [476], topology [476]</td>
<td>Replication topology for the dataservice Valid values are star,cluster-slave,master-slave,fan-in,clustered,cluster-alias,all-masters,direct</td>
<td>string</td>
</tr>
</tbody>
</table>
The tpm Deployment Command

<table>
<thead>
<tr>
<th>Value Type</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--track-schema-changes</code></td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td><code>--track-schema-changes</code> [477]</td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>track-schema-changes</code> [477]</td>
</tr>
<tr>
<td>Description</td>
<td>This will enable filters that track DDL statements and write the resulting change to files on slave hosts. The feature is intended for use in some batch deployments.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

9.8.19. U tpm Options

`--user`

| Option | `--user` [477] |
| Config File Options | `user` [477] |
| Description | System User |
| Value Type | string |

9.8.20. V tpm Options

`--vertica-dbname`

| Option | `--vertica-dbname` [477] |
| Aliases | `--repl-vertica-dbname` [477] |
| Config File Options | `repl-vertica-dbname` [477], `vertica-dbname` [477] |
| Description | Name of the database to replicate into |
| Value Type | string |

9.8.21. W tpm Options

`--witnesses`

| Option | `--witnesses` [477] |
| Aliases | `--dataservice-witnesses` [477] |
| Config File Options | `dataservice-witnesses` [477], `witnesses` [477] |
| Description | Witness hosts for the dataservice |
| Value Type | string |
Chapter 10. Replication Filters

Filtering operates by applying the filter within one, or more, of the stages configured within the replicator. Stages are the individual steps that occur within a pipeline, that take information from a source (such as MySQL binary log) and write that information to an internal queue, the transaction history log, or apply it to a database. Where the filters are applied ultimately affect how the information is stored, used, or represented to the next stage or pipeline in the system.

For example, a filter that removed out all the tables from a specific database would have different effects depending on the stage it was applied. If the filter was applied on the master before writing the information into the THL, then no slave could ever access the table data, because the information would never be stored into the THL to be transferred to the slaves. However, if the filter was applied on the slave, then some slaves could replicate the table and database information, while other slaves could choose to ignore them. The filtering process also has an impact on other elements of the system. For example, filtering on the master may reduce network overhead, albeit at a reduction in the flexibility of the data transferred.

In a standard replicator configuration with MySQL, the following stages are configured in the master, as shown in Figure 10.1, "Filters: Pipeline Stages on Masters".

Figure 10.1. Filters: Pipeline Stages on Masters

Where:

- **binlog-to-q Stage**
  
The `binlog-to-q` stage reads information from the MySQL binary log and stores the information within an in-memory queue.

- **q-to-thl Stage**
  
The in-memory queue is written out to the THL file on disk.

Within the slave, the stages configured by default are shown in Figure 10.2, "Filters: Pipeline Stages on Slaves".
Replication Filters

Figure 10.2. Filters: Pipeline Stages on Slaves

- **remote-to-thl Stage**
  Remote THL information is read from a master datasource and written to a local file on disk.

- **thl-to-q Stage**
  The THL information is read from the file on disk and stored in an in-memory queue.

- **q-to-dbms Stage**
  The data from the in-memory queue is written to the target database.

Filters can be applied during any configured stage, and where the filter is applied alters the content and availability of the information. The staging and filtering mechanism can also be used to apply multiple filters to the data, altering content when it is read and when it is applied.

Where more than one filter is configured for a pipeline, each filter is executed in the order it appears in the configuration. For example, within the following fragment:

```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.

### 10.1. Enabling/Disabling Filters

A number of standard filter configurations are created and defined by default within the static properties file for the Tungsten Replicator configuration.

Filters can be enabled through `tpm` to update the filter configuration:

- `--repl-svc-extractor-filters` [473]
  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` [474]
  Apply the filter between the internal queue and when the transactions are written to the THL on the master (`q-to-thl`).

- `--repl-svc-remote-filters` [473]
  Apply the filter between reading from the remote THL server and writing to the local THL files on the slave (`remote-to-thl`).

- `--repl-svc-applier-filters` [473]
  Apply the filter between reading from the internal queue and applying to the destination database (`q-to-dbms`).

Properties and options for an individual filter can be specified by setting the corresponding property value on the `tpm` command-line.

For example, to ignore a database schema on a slave, the `replicate` filter can be enabled, and the `replicator.filter.replicate.ignore` specifies the name of the schemas to be ignored. To ignore the schema `contacts`:
Replication Filters

A bad filter configuration will not stop the replicator from starting, but the replicator will be placed into the OFFLINE state.

To disable a previously enabled filter, empty the filter specification and (optionally) unset the corresponding property or properties. For example:

```
shell> ./tools/tpm update alpha --hosts=host1,host2,host3
   --remove-property=replicator.filter.replicate.ignore
```

Multiple filters can be applied on any stage, and the filters will be processes and called within the order defined within the configuration. For example, the following configuration:

```
shell> ./tools/tpm update alpha --hosts=host1,host2,host3
   --repl-svc-applier-filters=enumtostring,settostring,pkey
   --remove-property=replicator.filter.replicate.ignore
```

The filters are called in order:

1. enumtostring
2. settostring
3. pkey

The order and sequence can be important if operations are being performed on the data and they are relied on later in the stage. For example, if data is being filtered by a value that exists in a SET column within the source data, the settostring filter must be defined before the data is filtered, otherwise the actual string value will not be identified.

**Warning**

In some cases, the filter order and sequence can also introduce errors. For example, when using the pkey filter and the optimizeupdates filters together, pkey may remove KEY information from the THL before optimizeupdates attempts to optimize the ROW event, causing the filter to raise a failure condition.

The currently active filters can be determined by using the `trepctl status -name stages` command:

```
shell> trepctl status -name stages
Processing status command (stages)...
...                       NAME                 VALUE
...                       applier.class      : com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier
applier.name            : dbms
blockCommitRowCount     : 10
committedMinSeqno       : 3600
extractor.class         : com.continuent.tungsten.replicator.thl.THLParallelQueueExtractor
extractor.name          : parallel-q-extractor
filter.0.class          : com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter
filter.0.name           : mysqlsessions
filter.1.class          : com.continuent.tungsten.replicator.filter.PrimaryKeyFilter
filter.1.name           : pkey
filter.2.name           : bidiSlave
name                    : q-to-dbms
processedMinSeqno       : 1
taskCount               : 5
Finished status command (stages)...
```

The above output is from a standard slave replication installation showing the default filters enabled. The filter order can be determined by the number against each filter definition.

### 10.2. Enabling Additional Filters

The Tungsten Clustering configuration includes a number of filter configurations by default. However, not all filters are given a default configuration, and for some filters, multiple configurations may be needed to achieve more complex filtering requirements. Internally, filter configuration is defined through a property file that defines the filter name and corresponding parameters.

For example, the rename configuration is defined as follows:

```
replicator.filter.rename=com.continuent.tungsten.replicator.filter.RenameFilter
replicator.filter.rename.definitionsFile=${replicator.home.dir}/samples/extensions/java/rename.csv
```
The first line creates a new filter configuration using the corresponding Java class. In this case, the filter is named `rename`, as defined by the string `replicator.filter.rename`.

Configuration parameters for the filter are defined as values after the filter name. In this example, `definitionsFile` is the name of the property examined by the class to set the CSV file where the rename definitions are located.

To create an entirely new filter based on an existing filter class, a new property should created with the new filter definition in the configuration file.

Additional properties from this base should then be used. For example, to create a second rename filter definition called `custom`:

```shell
com.continuent.tungsten.replicator.filter.RenameFilter
com.continuent.tungsten.replicator.filter.RenameFilter
```

The filter can be enabled against the desired stage using the filter name `custom`:

```shell
../tools/tpm configure \
--repl-svc-applier-filters=custom
```

### 10.3. Filter Status

To determine which filters are currently being applied within a replicator, use the `trepctl status -name stages` command. This outputs a list of the current stages and their configuration. For example:

```shell
trepctl status -name stages
```

In the output, the filters applied to the applier stage are shown in the last block of output. Filters are listed in the order in which they appear within the configuration.

For information about the filter operation and any modifications or changes made, check the `trepsvc.log` log file.

### 10.4. Filter Reference

The different filter types configured and available within the replicate are designed to provide a number of different functionality and operations. Since the information exchanged through the THL system contains a copy of the statement or the row data that is being updated, the filters allow schemas, table and column names, as well as actual data to be converted at the stage in which they are applied.
Replication Filters

Filters are identified according to the underlying Java class that defines their operation. For different filters, further configuration and naming is applied according to the templates used when Tungsten Clustering is installed through `tpm`.

Tungsten Replicator also comes with a number of JavaScript filters that can either be used directly, or that can be modified and adapted to suit individual requirements. These filter scripts are located in `tungsten-replicator/support/filters-javascript`.

For the purposes of classification, the different filters have been categorised according to their main purpose:

- **Auditing**
  
  These filters provide methods for tracking database updates alongside the original table data. For example, in a financial database, the actual data has to be updated in the corresponding tables, but the individual changes that lead to that update must also be logged individually.

- **Content**
  
  Content filters modify or update the content of the transaction events. These may alter information, for the purposes of interoperability (such as updating enumerated or integer values to their string equivalents), or remove or filter columns, tables, and entire schemas.

- **Logging**
  
  Logging filters record information about the transactions into the standard replicator log, either for auditing or debugging purposes.

- **Optimization**
  
  The optimization filters are designed to simplify and optimize statements and row updates to improve the speed at which those updates can be applied to the destination dataserver.

- **Transformation**
  
  Transformation filters rename or reformat schemas and tables according to a set of rules. For example, multiple schemas can be merged to a single schema, or tables and column names can be updated.

- **Validation**
  
  Provide validation or consistency checking of either the data or the replication process.

- **Miscellaneous**
  
  Other filters that cannot be allocated to one of the existing filter classes.

The list of filters and their basic description are provided in the table below.

<table>
<thead>
<tr>
<th>Filter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BidiRemoteSlaveFilter</td>
<td>Content</td>
<td>Suppresses events that originated on the local service (required for correct slave operation)</td>
</tr>
<tr>
<td>BuildAuditTable</td>
<td>Auditing</td>
<td>Builds an audit table of changes for specified schemas and tables</td>
</tr>
<tr>
<td>BuildIndexTable</td>
<td>Transformation</td>
<td>Merges multiple schemas into a single schema</td>
</tr>
<tr>
<td>CaseMappingFilter</td>
<td>Transformation</td>
<td>Transforms schema, table and column names to upper or lower case</td>
</tr>
<tr>
<td>CDCMetadataFilter</td>
<td>Auditing</td>
<td>Records change data capture for transactions to a separate change table [auditing]</td>
</tr>
<tr>
<td>ColumnNameFilter</td>
<td>Validation</td>
<td>Adds column name information to row-based replication events</td>
</tr>
<tr>
<td>ConsistencyCheckFilter</td>
<td>Validation</td>
<td>Adds consistency checking to events</td>
</tr>
<tr>
<td>DatabaseTransformFilter</td>
<td>Transformation</td>
<td>Transforms database or table names using regular expressions</td>
</tr>
<tr>
<td>DummyFilter</td>
<td>Miscellaneous</td>
<td>Allows for confirmation of filter configuration</td>
</tr>
<tr>
<td>EnumToStringFilter</td>
<td>Content</td>
<td>Updates enumerated values to their string-based equivalent</td>
</tr>
<tr>
<td>EventMetadataFilter</td>
<td>Content</td>
<td>Filters events based on metadata; used by default within sharding and multi-master topologies</td>
</tr>
<tr>
<td>HeartbeatFilter</td>
<td>Validation</td>
<td>Detects heartbeat events on masters or slaves</td>
</tr>
<tr>
<td>JavaScriptFilter</td>
<td>Miscellaneous</td>
<td>Enables filtering through custom JavaScripts</td>
</tr>
<tr>
<td>LoggingFilter</td>
<td>Logging</td>
<td>Logs filtered events through the standard replicator logging mechanism</td>
</tr>
<tr>
<td>MySQLSessionSupportFilter</td>
<td>Content</td>
<td>Filters transactions for session specific temporary tables and variables</td>
</tr>
</tbody>
</table>
### Replication Filters

<table>
<thead>
<tr>
<th>Filter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OptimizeUpdates-Filter</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>ReplicateColumns-Filter</td>
<td>Content</td>
<td>Removes selected columns from row-based transaction data</td>
</tr>
<tr>
<td>ReplicateFilter</td>
<td>Content</td>
<td>Selects or ignores specification schemas and/or databases</td>
</tr>
<tr>
<td>SetToStringFilter</td>
<td>Content</td>
<td>Converts integer values in <code>SET</code> datatypes to string values</td>
</tr>
<tr>
<td>ShardFilter</td>
<td>Content</td>
<td>Used to enforce database schema sharding between specific masters</td>
</tr>
<tr>
<td>TimeDelayFilter</td>
<td>Miscellaneous</td>
<td>Delays transactions until a specific point in time has passed</td>
</tr>
</tbody>
</table>

In the following reference sections:

- **Pre-configured filter name** is the filter name that can be used against a stage without additional configuration.
- **Property prefix** is the prefix string for the filter to be used when assigning property values.
- **Classname** is the Java class name of the filter.
- **Parameter** is the name of the filter parameter can be set as a property within the configuration.
- **Data compatibility** indicates whether the filter is compatible with row-based events, statement-based events, or both.

#### 10.4.1. `ansiquotes.js` Filter

The `ansiquotes` filter operates by inserting an SQL mode change to `ANSI_QUOTES` into the replication stream before a statement is executed, and returning to an empty SQL mode.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>JavaScript Filter File</th>
<th>Property prefix</th>
<th>Stage compatibility</th>
<th><code>tpm</code> Option compatibility</th>
<th>Data compatibility</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ansiquotes</code></td>
<td>tungsten-replicator/support/filters-javascript/ansiquotes.js</td>
<td><code>replicator.filter.ansiquotes</code></td>
<td><code>binlog-to-q</code></td>
<td><code>--svc-extractor-filters [473]</code></td>
<td>Any event</td>
<td></td>
</tr>
</tbody>
</table>

This changes a statement such as:

```sql
INSERT INTO notepad VALUES ('message',0);
```

To:

```sql
SET sql_mode='ANSI_QUOTES';
INSERT INTO notepad VALUES ('message',0);
SET sql_mode='';
```

This is achieved within the JavaScript by processing the incoming events and adding a new statement before the first `DBMSData` Object in each event:

```javascript
query = "SET sql_mode='ANSI_QUOTES'";
newStatement = new com.continuent.tungsten.replicator.dbms.StatementData(
    query,
    null,
    null,
    true);
data.add(0, newStatement);
```

A corresponding statement is appended to the end of the event:

```javascript
query = "SET sql_mode=''";
```
10.4.2. BidiRemoteSlave [BidiSlave] Filter

The BidiRemoteSlaveFilter is used by Tungsten Replicator to prevent statements that originated from this service (i.e. where data was extracted), being re-applied to the database. This is a requirement for replication to prevent data that may be transferred between hosts being re-applied, particularly in multi-master and other bi-directional replication deployments.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>bidiSlave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property prefix</td>
<td>replicator.filter.bidiSlave</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>tpms Option compatibility</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>
<tr>
<td>localServiceName</td>
<td>string</td>
</tr>
<tr>
<td>allowBidiUnsafe</td>
<td>boolean</td>
</tr>
<tr>
<td>allowAnyRemoteService</td>
<td>boolean</td>
</tr>
</tbody>
</table>

The filter works by comparing the server ID of the THL event that was created when the data was extracted against the server ID of the current server.

When deploying through the `tpm` service the filter is automatically enabled for remote slaves. For complex deployments, particularly those with bi-directional replication (including multi-master), the `allowBidiUnsafe` parameter may need to be enabled to allow certain statements to be re-executed.

10.4.3. breadcrumbs.js Filter

The `breadcrumbs` filter records regular 'breadcrumb' points into a MySQL table for systems that do not have global transaction IDs. This can be useful if recovery needs to be made to a specific point. The example also shows how metadata information for a given event can be updated based on the information from a table.

<p>| Pre-configured filter name | ansiquotes |
| JavaScript Filter File     | tungsten-replicator/support/filters-javascript/breadcrumbs.js |
| Property prefix            | replicator.filter.breadcrumbs |
| Stage compatibility        | binlog-to-q |
| tpms Option compatibility  | <code>--svc-extractor-filters</code> |
| Data compatibility         | Any event |</p>
<table>
<thead>
<tr>
<th>Parameters</th>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>server_id</td>
<td>numeric</td>
<td>[none]</td>
<td>MySQL server ID of the current host</td>
<td></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 works in two stages, and operates because the JavaScript instance is persistent as long as the Replicator is running. This means that data extracted during replication stays in memory and can be applied to later transactions. Hence the breadcrumb ID and offset information can be identified and used on each call to the filter function.

The first part of the filter event identifies the breadcrumb table and extracts the identified breadcrumb counter:

```java
if (table.compareToIgnoreCase("breadcrumbs") == 0) {
    columnValues = oneRowChange.getColumnValues();
    for (row = 0; row < columnValues.size(); row++) {
        values = columnValues.get(row);
        server_id_value = values.get(0);
        if (server_id == null || server_id == server_id_value.getValue()) {
            counter_value = values.get(1);
            breadcrumb_counter = counter_value.getValue();
            breadcrumb_offset = 0;
        }
    }
}
```

The second part updates the event metadata using the extracted breadcrumb information:

```java
topLevelEvent = event.getDBMSEvent();
if (topLevelEvent != null) {
    xact_server_id = topLevelEvent.getMetadataOptionValue("mysql_server_id");
    if (server_id == xact_server_id) {
        topLevelEvent.setMetaDataOption("breadcrumb_counter", breadcrumb_counter);
        topLevelEvent.setMetaDataOption("breadcrumb_offset", breadcrumb_offset);
    }
}
```

To calculate the offset (i.e. the number of events since the last breadcrumb value was extracted), the filter determines if the event was the last fragment processed, and updates the offset counter:

```java
if (event.getLastFrag()) {
    breadcrumb_offset = breadcrumb_offset + 1;
}
```

### 10.4.4. BuildAuditTable Filter

The BuildAuditTable filter populates a table with all the changes to a database so that the information can be tracked for auditing purposes.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>Not defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.BuildAuditTable</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.bidiSlave</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>replicator.filter.bidiSlave</td>
</tr>
<tr>
<td>tpn Option compatibility</td>
<td>replicator.filter.bidiSlave</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targetTableName</td>
<td>string</td>
<td></td>
<td>Name of the table where audit information will be stored</td>
</tr>
</tbody>
</table>
10.4.5. BuildIndexTable Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>buildindextable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>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>tpim Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</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>

10.4.6. CaseMapping [CaseTransform] 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>tpim Option 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>

10.4.7. CDCMetadata [CustomCDC] Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>customcdc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>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>tpim Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</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>
<tr>
<td>tableNameSuffix</td>
<td>string</td>
<td></td>
<td>Specifies the table name suffix for the CDC tables. If the schema suffix is not specified, this allows CDC tables to be created within the same schema</td>
</tr>
<tr>
<td>toSingleSchema</td>
<td>string</td>
<td></td>
<td>Creates and writes CDC data within a single schema</td>
</tr>
<tr>
<td>sequenceBeginning</td>
<td>numeric</td>
<td>1</td>
<td>Sets the sequence number of the CDC data. The sequence is used to identify individual changesets in the CDC</td>
</tr>
</tbody>
</table>
10.4.8. ColumnName Filter

The `ColumnNameFilter` loads the table specification information for tables and adds this information to the THL data for information extracted using row-base replication.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>colnames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.ColumnNameFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.colnames</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q</td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td>--svc-extractor-filters [473]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
<tr>
<td>Keeps Cached Data</td>
<td>Yes</td>
</tr>
<tr>
<td>Cached Refreshed When?</td>
<td>Emptied when going OFFLINE; Updated when ALTER statement seen</td>
</tr>
<tr>
<td>Metadata Updated</td>
<td>Yes; tungsten_filter_columnname=true</td>
</tr>
</tbody>
</table>

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>string</td>
<td><code>${replicator.global.extract.db.user}</code></td>
<td>The username for the connection to the database for looking up column definitions</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td><code>${replicator.global.extract.db.password}</code></td>
<td>The password for the connection to the database for looking up column definitions</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>jdbc:mysql:thin://${replicator.global.extract.db.host}:${replicator.global.extract.db.port}/${replicator.schema}?createDatabase=true</td>
<td>JDBC URL of the database connection to use for looking up column definitions</td>
</tr>
<tr>
<td>addSignedFlag</td>
<td>boolean</td>
<td>true</td>
<td>Determines whether the signed flag information for columns should be added to the metadata for each column.</td>
</tr>
<tr>
<td>ignoreMissingTables</td>
<td>boolean</td>
<td>true</td>
<td>When true, tables that do not exist will not trigger metadata and column names to be added to the THL data.</td>
</tr>
</tbody>
</table>

**Note**

This filter is designed to be used for testing and with heterogeneous replication where the field name information can be used to construct and build target data structures.

The filter is required for the correct operation of heterogeneous replication, for example when replicating to MongoDB. The filter works by using the replicator username and password to access the underlying database and obtain the table definitions. The table definition information is cached within the replication during operation to improve performance.

When extracting data from the binary log using row-based replication, the column names for each row of changed data are added to the THL.

Enabling this filter changes the THL data from the following example, shown without the column names:

```
SEQ# = 27 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 18:29:38.0
- EVENTID = mysql-bin:000012:00000000000004369:0
- 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) = 1
  - COL(2:1) = 23
  - COL(3:1) = 45
  - COL(4:1) = 45000.08
```

To a version where the column names are included as part of the THL record:
When the row-based data is applied to a non-MySQL database the column name information is used by the applier to specify the column, or they key when the column and value is used as a key/value pair in a document-based store.

### 10.4.9. ConsistencyCheck Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>Not defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.consistency.ConsistencyCheckFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>Not defined</td>
</tr>
</tbody>
</table>

#### Stage compatibility
tpm

#### Option compatibility

#### Data compatibility
Row events only

#### Parameters

<table>
<thead>
<tr>
<th>definitionsFile</th>
<th>string</th>
<th>support/filters-config/convertstringfrommysql.json</th>
<th>JSON file containing the definition of which events and which tables to skip</th>
</tr>
</thead>
</table>

### 10.4.10. ConvertStringFromMySQL Filter

The ConvertStringFromMySQLFilter is designed to be used in replicators that are used in conjunction either with existing native MySQL to MySQL replication deployments, or clustering deployments where the replication has been configured to use native MySQL byte storage for strings. These are incompatible with heterogeneous deployments as the string is stored internally and in the THL in a format that is useful only within similarly configured replicators.

Conversion can be selected to happen for all valid columns (VARCHAR or CHAR column types only), or for selected columns within specific tables and schemas. All conversions are made with the relevant character set for the table and THL event.

#### Note

Conversion will not occur on incompatible columns. For example, conversion will not be applied to INT columns. This is the case even if the column has been explicitly set to convert the column.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>convertstringfrommysql</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.ConvertStringFromMySQLFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>Not defined</td>
</tr>
</tbody>
</table>

#### Stage compatibility
tpm

#### Option compatibility

#### Data compatibility
Row events only

#### Parameters

<table>
<thead>
<tr>
<th>definitionsFile</th>
<th>string</th>
<th>support/filters-config/convertstringfrommysql.json</th>
<th>JSON file containing the definition of which events and which tables to skip</th>
</tr>
</thead>
</table>

Configuration of the filter is made using the generic JSON file, which supports both default options to happen for all tables not otherwise explicitly specified. The default JSON file converts all valid (VARCHAR or CHAR) column types only:

```json
{
  "__default": {
    "*": "true",
  },
  "SCHEMA": {
```
Warning

For column specific selection to work, the column names must be included within the THL. The `colnames` filter must have been enabled either before this filter, or on the extractor where the data was originally extracted.

The default section handles the default response when an explicit schema or table name does not appear. Further sections are then organised by schema, table and column name. Where the setting is `true`, conversion will take place. A `false` disables conversion.

To enable conversion on a single column `DESCRIPTION` within the `SALES.INVOICE` schema/table while disabling conversion on all other columns:

```json
{
  "__default": {
    "*" : "false",
  },
  "SALES": {
    "INVOICE": {
      "DESCRIPTION": "true",
    },
  }
}
```

To convert all compatible columns in all tables within a schema:

```json
{
  "__default": {
    "*" : "false",
  },
  "SALES": {
    "*": {
      "*": "true",
    },
  }
}
```

A primary use case for this filter is for cluster-slave replication from a cluster to a datawarehouse. For more details, please see Section 3.9, "Replicating from a Cluster to a Datawarehouse".

Source Cluster Example

For cluster-slave replication to a datawarehouse, the source cluster nodes must use ROW-based MySQL binary logging, and also must have two extractor filters enabled, `colanmes` and `pkey`.

For example, on every cluster node the lines below would be added to the `/etc/tungsten/tungsten.ini` file in the service stanza, then `tpm update` would be executed:

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

For staging deployments, prepend two hyphens to each line and include on the command line.

For more details about configuring the source cluster, please see Section 3.8, "Replicating Data Out of a Cluster".

Target Cluster-Slave Example

On the replication slave node, copy the `convertstringfrommysql.json` filter configuration sample file into the `/opt/replicator/share` directory then edit it to suit:

```
cp /opt/replicator/tungsten/tungsten-replicator/support/filters-config/convertstringfrommysql.json /opt/replicator/share/
vi /opt/replicator/share/convertstringfrommysql.json
```

Once the `convertstringfrommysql.json` configuration file has been edited, update the `/etc/tungsten/tungsten.ini` file to add and configure the `convertstringfrommysql` filter.

For example, configure a service named `omega` on `host6` to read from the cluster nodes defined by cluster-alias `alpha`.

```
[alpha]
topology=cluster-alias
master=host1
members=host1,host2,host3
thl-port=2112
[omega]
```
Replication Filters

topology=cluster-slave
relay-host=host6
relay-source=alpha
replicator.filter.convertstringfrommysql.
definitionsFile=/opt/replicator/share/convertstringfrommysql.json

For more details about configuring the target cluster-slave node, please see Section 3.8, “Replicating Data Out of a Cluster”.

10.4.11. DatabaseTransform (dbtransform) Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>dbtransform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.DatabaseTransformFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.dbtransform</td>
</tr>
</tbody>
</table>

Stage compatibility and ipm Option compatibility are not specified.

Data compatibility

Any event

<table>
<thead>
<tr>
<th>Parameters</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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>from_regex1</th>
<th>string</th>
<th>foo</th>
<th>The search regular expression to use when renaming databases or tables [group 1]; corresponds to to_regex1</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>from_regex2</th>
<th>string</th>
<th>The search regular expression to use when renaming databases or tables [group 2]; corresponds to to_regex1</th>
</tr>
</thead>
<tbody>
<tr>
<td>to_regex2</td>
<td>string</td>
<td>The replace regular expression to use when renaming databases or tables [group 2]; corresponds to from_regex1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>from_regex3</th>
<th>string</th>
<th>The search regular expression to use when renaming databases or tables [group 3]; corresponds to to_regex1</th>
</tr>
</thead>
<tbody>
<tr>
<td>to_regex3</td>
<td>string</td>
<td>The replace regular expression to use when renaming databases or tables [group 3]; corresponds to from_regex1</td>
</tr>
</tbody>
</table>

10.4.12. dbrename.js Filter

The dbrename JavaScript filter renames database [schemas] using two parameters from the properties file, the dbsource and dbtarget. Each event is then processed, and the statement or row based schema information is updated to dbtarget when the dbsource schema is identified.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>dbrename</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/dbrename.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.dbrename</td>
</tr>
</tbody>
</table>

Stage compatibility

binlog-to-q

Data compatibility

Any event

<table>
<thead>
<tr>
<th>Parameters</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
The operation of the filter is straightforward, because the schema name is exposed and settable within the statement and row change objects:

```javascript
function filter(event) {
    sourceName = filterProperties.getString("dbsource");
    targetName = filterProperties.getString("dbtarget");
    data = event.getData();
    for(i=0;i<data.size();i++) {
        d = data.get(i);
        if(d instanceof com.continuent.tungsten.replicator.dbms.StatementData) {
            if(d.getDefaultSchema() != null &&
                d.getDefaultSchema().compareTo(sourceName)==0) {
                d.setDefaultSchema(targetName);
            }
        } else if(d instanceof com.continuent.tungsten.replicator.dbms.RowChangeData) {
            rowChanges = data.get(i).getRowChanges();
            for(j=0;j<rowChanges.size();j++) {
                oneRowChange = rowChanges.get(j);
                if(oneRowChange.getSchemaName().compareTo(sourceName)==0) {
                    oneRowChange.setSchemaName(targetName);
                }
            }
        }
    }
}
```

### 10.4.13. `dbselector.js` Filter

Filtering only a single database schema can be useful when you want to extract a single schema for external processing, or for sharding information across multiple replication targets. The `dbselector` filter deletes all statement and row changes, except those for the selected table. To configure, the `db` parameter to the filter configuration specifies the schema to be replicated.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>dbselector</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/dbselector.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.dbselector</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q, q-to-thl, q-to-dbms</td>
</tr>
<tr>
<td><code>rpm</code> Option compatibility</td>
<td>--svc-extractor-filters [473], --svc-applier-filters [473]</td>
</tr>
</tbody>
</table>

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>db</td>
<td>string</td>
<td>[none]</td>
<td>Database to be selected</td>
</tr>
</tbody>
</table>

Within the filter, statement changes look for the schema in the `StatementData` object and remove it from the array:

```javascript
if (d instanceof com.continuent.tungsten.replicator.dbms.StatementData) {
    if(d.getDefaultSchema().compareTo(db)!=0) {
        data.remove(i);
        i--;
    }
}
```

Because entries are being removed from the list of statements, the iterator used to process each item must be explicitly decremented by 1 to reset the counter back to the new position.

Similarly, when looking at row changes in the `RowChangeData`:
Replication Filters

```java
else if(d instanceof com.continuent.tungsten.replicator.dbms.RowChangeData)
{
    rowChanges = data.get().getRowChanges();
    for(j=0;j<rowChanges.size();j++)
    {
        oneRowChange = rowChanges.get(j);
        if(oneRowChange.getSchemaName().compareTo(db)!=0)
        {
            rowChanges.remove(j);
            j--;
        }
    }
}
```

### 10.4.14. `dbupper.js` Filter

The `dbupper` filter changes the case of the schema name for all schemas to uppercase. The schema information is easily identified in the statement and row based information, and therefore easy to update.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>dbupper</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/dbupper.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.dbupper</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q</td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td><code>--svc-extractor-filters</code>[473], <code>--svc-applier-filters</code>[473]</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><code>from</code></td>
<td>string</td>
<td>(none)</td>
<td>Database name to be converted to uppercase</td>
</tr>
</tbody>
</table>

For example, within statement data:

```java
from = d.getDefaultSchema();
if (from != null)
{
    to   = from.toUpperCase();
    d.setDefaultSchema(to);
}
```

### 10.4.15. `dropcolumn.js` Filter

The `dropcolumn` filter enables columns in the THL to be dropped. This can be useful when replicating Personal Identification Information, such as email addresses, phone number, personal identification numbers and others are within the THL but need to be filtered out on the slave.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>dropcolumn</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/dropcolumn.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.dropcolumn</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q, <code>q-to-dbs</code></td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td><code>--svc-extractor-filters</code>[473], <code>--svc-applier-filters</code>[473]</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><code>definitionsFile</code></td>
<td>Filename</td>
<td>~/dropcolumn.json</td>
<td>Location of the definitions file for dropping columns</td>
</tr>
<tr>
<td><code>fillGaps</code></td>
<td>Boolean</td>
<td>false</td>
<td>When true, re-order THL Column Index IDs to sequential numbers</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:

```
shell> tpm update --svc-extractor-filters=colnames,dropcolumn \
    --property=replicator.filter.dropcolumn.definitionsFile=/opt/continuent/share/dropcolumn.json
```
Replication Filters

A sample configuration file is provided in `/opt/continuent/share/dropcolumn.json`. The format of the file is a JSON array of schema/table/column specifications:

```
[  
  {  
    "schema": "vip",  
    "table": "clients",  
    "columns": [  
      "personal_code",  
      "birth_date",  
      "email"
    ]
  },  
  ...
]
```

Where:

- **schema** specifies the name of the schema on which to apply the filtering. If `*` is given, all schemas are matched.
- **table** specifies the name of the table on which to apply the filtering. If `*` is given, all tables are matched.
- **columns** is an array of column names to be matched.

For example:

```
[  
  {  
    "schema": "vip",  
    "table": "clients",  
    "columns": [  
      "personal_code",  
      "birth_date",  
      "email"
    ]
  },  
  ...
]
```

Filters the columns `email`, `birth_date`, and `personal_code` within the `clients` table in the `vip` schema.

To filter the `telephone` column in any table and any schema:

```
[  
  {  
    "schema": "*",  
    "table": "*",  
    "columns": [  
      "telephone"
    ]
  }
]
```

Care should be taken when dropping columns on the slave and master when the column order is different or when the names of the column differ:

- If the column order is same, even if dropcolumn.js is used, leave the default setting for the property `replicator.applier.dbms.getColumnMetadataFromDB=true`.
- If the column order is different on the master and slave, set `replicator.applier.dbms.getColumnMetadataFromDB=false`.
- If slave's column names are different, regardless of differences in the order, use the default property setting `replicator.applier.dbms.getColumnMetadataFromDB=true`.

If the filter is enabled on the extractor, the columns will be removed from the THL and the resulting THL Index will appear to have gaps in the THL column index ID, for example:

```
...
- SQL(0) = 
  - ACTION = INSERT 
  - SCHEMA = sample 
  - TABLE = demotable 
  - ROW = 0 
  - COL(1: id) = 11 
  - COL(2: col_a) = UK 
  - COL(4: col_d) = 5678 
  - COL(5: col_d) = ABC 
  - COL(7: col_g) = 2019-09-05 07:21:48.0 
  - KEY(1: id) = NULL 
...
```
Replication Filters

For JDBC targets, this is expected and required to ensure accurate column mapping, however for Batch Appliers the gap in the Index ID's will cause the applier to fail with a CSV Column Mismatch error, therefore, if your target is a Batch target (eg Hadoop, Redshift, Vertica) you need to add the following property to your configuration:

```
property=replicator.filter.dropcolumn.fillGaps=true
```

With this property in place, the resulting THL from the above example would now look like the following:

```
... - SQL(0) = - ACTION = INSERT - SCHEMA = sample - TABLE = demotable - ROW# = 0 - COL(1: id) = 11 - COL(2: col_a) = UK - COL(3: col_c) = 5678 - COL(4: col_d) = ABC - COL(5: col_g) = 2019-09-05 07:21:48.0 - KEY(1: id) = NULL ...
```

10.4.16. dropcomments.js Filter

The dropcomments filter removes comments from statements within the event data.

```
Row changes do not have comments, so the filter only has to change the statement information, which is achieved by using a regular expression:

sqlOriginal = d.getQuery();
sqlNew = sqlOriginal.replaceAll(/\*(?:.\r\n)*?\*/,'');
d.setQuery(sqlNew);
```

To handle the case where the statement could only be a comment, the statement is removed:

```
if(sqlNew.trim().length()==0)
{
    data.remove(i);
    i--;
}
```

10.4.17. dropddl.js Filter

Within certain replication deployments, there may be occasions where you do not require specific DDL statements to be replicated. For example, you may NOT want to allow TRIGGERS or VIEWS to be replicated, but you do want TABLES, INDEXES, FUNCTIONS and PROCEDURES.

The dropddl filter allows you to configure which, if any, ddl statements to drop from THL and not be replicated.

```
Row changes do not have comments, so the filter only has to change the statement information, which is achieved by using a regular expression:

sqlOriginal = d.getQuery();
sqlNew = sqlOriginal.replaceAll(/\*(?:.\r\n)*?\*/,'');
d.setQuery(sqlNew);
```

To handle the case where the statement could only be a comment, the statement is removed:

```
if(sqlNew.trim().length()==0)
{
    data.remove(i);
    i--;
}
```
Replication Filters

<table>
<thead>
<tr>
<th>Filter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dropTableDDL</td>
<td>boolean</td>
<td>false</td>
<td>Drop CREATE TABLE and DROP TABLE statements</td>
</tr>
<tr>
<td>dropViewDDL</td>
<td>boolean</td>
<td>false</td>
<td>Drop CREATE VIEW and DROP VIEWS statements</td>
</tr>
<tr>
<td>dropIndexDDL</td>
<td>boolean</td>
<td>false</td>
<td>Drop CREATE INDEX and DROP INDEX statements</td>
</tr>
<tr>
<td>dropTriggerDDL</td>
<td>boolean</td>
<td>false</td>
<td>Drop CREATE TRIGGER and DROP TRIGGER statements</td>
</tr>
<tr>
<td>dropFunctionDDL</td>
<td>boolean</td>
<td>false</td>
<td>Drop CREATE FUNCTION and DROP FUNCTION statements</td>
</tr>
<tr>
<td>dropProcedureDDL</td>
<td>boolean</td>
<td>false</td>
<td>Drop CREATE PROCEDURE and DROP PROCEDURE statements</td>
</tr>
</tbody>
</table>

Example:

If you wish to drop VIEW and TRIGGER DDL on the applier, the following options need to be added to your configuration:

```sql
svc-applier-filters=dropddl
property=replicator.filter.dropddl.dropViewDDL=true
property=replicator.filter.dropddl.dropTriggerDDL=true
```

10.4.18. dropmetadata.js Filter

All events within the replication stream contain metadata about each event. This information can be individually processed and manipulated. The `dropmetadata` filter removes specific metadata from each event, configured through the `option` parameter to the filter.

**Pre-configured filter name**: dropmetadata

**JavaScript Filter File**: tungsten-replicator/support/filters-javascript/dropmetadata.js

**Property prefix**: replicator.filter.ansiquotes

**Stage compatibility**: binlog-to-q, q-to-dbms

**Option compatibility**: --svc-extractor-filters, --svc-applier-filters

**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>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:

```javascript
for (n = 0; n < metaData.size(); m++) {
    option = metaData.get(n);
    if (option.getOptionName().compareTo(optionName) == 0) {
        metaData.remove(n);
        break;
    }
}
```

10.4.19. dropstatementdata.js Filter

Within certain replication deployments, enforcing that only row-based information is replicated is important to ensure that the row data is replicated properly. For example, when replicating to databases that do not accept statements, these events must be filtered out.

**Pre-configured filter name**: dropstatementdata

**JavaScript Filter File**: tungsten-replicator/support/filters-javascript/dropstatementdata.js

**Property prefix**: replicator.filter.dropstatementdata

**Stage compatibility**: binlog-to-q, q-to-dbms

**Option compatibility**: --svc-extractor-filters, --svc-applier-filters

**Data compatibility**: Any event

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

This is achieved by checking for statements, and then removing them from the event:
10.4.20. Dummy Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>dummy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td><code>com.continuent.tungsten.replicator.filter.DummyFilter</code></td>
</tr>
<tr>
<td>Property prefix</td>
<td><code>replicator.filter.dummy</code></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Parameters</td>
<td>[none]</td>
</tr>
</tbody>
</table>

10.4.21. 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><code>enumtoString</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td><code>com.continuent.tungsten.replicator.filter.EnumToStringFilter</code></td>
</tr>
<tr>
<td>Property prefix</td>
<td><code>replicator.filter.enumtoString</code></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td><code>binlog-to-q</code></td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td><code>--repl-svc-extractor-filters [473]</code></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
<tr>
<td>Metadata Updated</td>
<td>Yes; <code>tungsten_filter_enumtoString=true</code></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>user</code></td>
<td>string</td>
<td><code>${replicator.global.extract.db.user}</code></td>
<td>The username for the connection to the database for looking up column definitions</td>
</tr>
<tr>
<td><code>password</code></td>
<td>string</td>
<td><code>${replicator.global.extract.db.password}</code></td>
<td>The password for the connection to the database for looking up column definitions</td>
</tr>
<tr>
<td><code>url</code></td>
<td>string</td>
<td><code>jdbc:mysql:thin://${replicator.global.extract.db.host}:${replicator.global.extract.db.port}/${replicator.schema}?createDatabase=true</code></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>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>city</td>
<td>int(11)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesman</td>
<td>set('Alan','Zachary')</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>decimal(10,2)</td>
<td>YES</td>
<td>NULL</td>
<td></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.

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:

The information is critical when applying the data to a dataserver that is not aware of the table definition, such as when replicating to Oracle or MongoDB.

The examples here also show the Section 10.4.38, “SetToString Filter” and Section 10.4.8, “ColumnName Filter” filters.

### 10.4.22. EventMetadata Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>eventmetadata</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.EventMetadataFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.eventmetadata</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>Row events</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Metadata Updated</td>
</tr>
<tr>
<td></td>
<td>Yes; tungsten_filter_settostring=true</td>
</tr>
<tr>
<td>Parameters</td>
<td>[none]</td>
</tr>
</tbody>
</table>

### 10.4.23. foreignkeychecks.js Filter

The `foreignkeychecks` filter switches off foreign key checks for statements using the following statements:
Replication Filters

<table>
<thead>
<tr>
<th>JavaScript Filter File</th>
<th>tungsten-replicator/support/filters-javascript/foreignkeychecks.js</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property prefix</td>
<td>replicator.filter.foreignkeychecks</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 [473], --svc-applier-filters [473]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
</tbody>
</table>

The process checks the statement data and parses the content of the SQL statement by first trimming any extraneous space, and then converting the statement to upper case:

```java
upCaseQuery = d.getQuery().trim().toUpperCase();
```

Then comparing the string for the corresponding statement types:

```java
if(upCaseQuery.startsWith("CREATE TABLE") ||
   upCaseQuery.startsWith("DROP TABLE") ||
   upCaseQuery.startsWith("ALTER TABLE") ||
   upCaseQuery.startsWith("RENAME TABLE")
)
```

If they match, a new statement is inserted into the event that disables foreign key checks:

```java
query = "SET foreign_key_checks=0";
newStatement = new com.continuent.tungsten.replicator.dbms.StatementData(
   d.getDefaultSchema(),
   null,
   query
);
data.add(0, newStatement);
i++;
```

The use of `a` in the `add()` method inserts the new statement before the others within the current event.

### 10.4.24. Heartbeat Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>(none)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.HeartbeatFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>(none)</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>heartbeatInterval</td>
<td>numeric</td>
<td>3000</td>
<td>Interval in milliseconds when a heartbeat event is inserted into the THL</td>
</tr>
</tbody>
</table>

### 10.4.25. insertonly.js Filter

The `insertonly` filter filters events to only include ROW-based events using INSERT.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>insertonly</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/insertonly.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.insertonly</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>q-to-dbms</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>--svc-applier-filters [473]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

498
This is achieved by examining each row and removing row changes that do not match the `INSERT` action type:

```java
if(oneRowChange.getAction() != "INSERT")
{
    rowChanges.remove(j);
    j--;
}
```

### 10.4.26. Logging Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>logger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td><code>com.continuent.tungsten.replicator.filter.LoggingFilter</code></td>
</tr>
<tr>
<td>Property prefix</td>
<td><code>replicator.filter.logger</code></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td><code>--svc-extractor-filters</code>[473], <code>--svc-thl-filters</code>[474], <code>--svc-applier-filters</code>[473]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Parameters</td>
<td>(none)</td>
</tr>
</tbody>
</table>

### 10.4.27. MySQLSessionSupport (mysqlsessions) Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>mysqlsessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td><code>com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter</code></td>
</tr>
<tr>
<td>Property prefix</td>
<td><code>replicator.filter.mysqlsession</code></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td><code>--svc-extractor-filters</code>[473], <code>--svc-thl-filters</code>[474], <code>--svc-applier-filters</code>[473]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Parameters</td>
<td>(none)</td>
</tr>
</tbody>
</table>

### 10.4.28. 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><code>com.continuent.tungsten.replicator.filter.NetworkClientFilter</code></td>
</tr>
<tr>
<td>Property prefix</td>
<td><code>replicator.filter.networkclient</code></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>Any</td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td><code>--svc-extractor-filters</code>[473], <code>--svc-thl-filters</code>[474], <code>--svc-applier-filters</code>[473]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
</tr>
<tr>
<td><code>definitionsFile</code></td>
<td>pathname</td>
</tr>
<tr>
<td><code>serverPort</code></td>
<td>number</td>
</tr>
<tr>
<td><code>timeout</code></td>
<td>number</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.
Replication Filters

The filter operation works as follows:

- All filtered data will be sent to a single network server, at the configured port.
- A single network server can be used to provide multiple transformations.
- The JSON configuration file for the filter supports multiple types and multiple column definitions.
- The protocol used by the network filter must be followed to effectively process the information. A failure in the network server or communication will cause the replicator to raise an error and replication to go **OFFLINE**.
- The network server must be running before the replicator is started. If the network server cannot be found, replication will go **OFFLINE**.

Correct operation requires building a suitable network filter using the defined protocol, and creating the JSON configuration file. A sample filter is provided for reference.

### 10.4.28.1. Network Client Configuration

The format of the configuration file defines the translation operation to be requested from the network client, in addition to the schema, table and column name. The format for the file is JSON, with the top-level hash defining the operation, and an array of field selections for each field that should be processed accordingly. For example:

```json
{
  "String_to_HEX_v1": [
    {
      "table": "hextable",
      "schema": "hexdb",
      "columns": [
        "hexcol"
      ]
    }
  ]
}
```

The operation in this case is **String_to_HEX_v1**; this will be sent to the network server as part of the request. The column definition follows.

To send multiple columns from different tables to the same translation:

```json
{
  "String_to_HEX_v1": [
    {
      "table": "hextable",
      "schema": "hexdb",
      "columns": [
        "hexcol"
      ]
    },
    {
      "table": "hexagon",
      "schema": "sourcetext",
      "columns": [
        "itemtext"
      ]
    }
  ]
}
```

Alternatively, to configure different operations for the same two tables:

```json
{
  "String_to_HEX_v1": [
    {
      "table": "hextable",
      "schema": "hexdb",
      "columns": [
        "hexcol"
      ]
    }
  ],
  "HEX_to_String_v1": [
    {
      "table": "hexagon",
      "schema": "sourcetext",
      "columns": [
        "itemtext"
      ]
    }
  ]
}
```
10.4.28.2. Network Filter Protocol

The network filter protocol has been designed to be both lightweight and binary data compatible, as it is designed to work with data that may be heavily encoded, binary, or compressed in nature.

The protocol operates through a combined JSON and optional binary payload structure that communicates the information. The JSON defines the communication type and metadata, while the binary payload contains the raw or translated information.

The filter communicates with the network server using the following packet types:

- **prepare**

  The `prepare` message is called when the filter goes online, and is designed to initialize the connection to the network server and confirm the supported filter types and operation. The format of the connection message is:

  ```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" : "firststep",
    "type" : "filter",
    "payload" : 22,
    "row" : 8,
    "column" : "hexcol",
    "table" : "hextable",
    "seqno" : 145196,
    "fragments" : 1,
    "protocol" : "v0_9",
    "fragment" : 1
  }
  ``

  Where:

  - **protocol**
    
    The protocol version.

  - **service**
    
    The service name the requested the filter.

  - **type**
    
    The message type, in this case, *filter*.

  - **row**
    
    The row of the source information from the THL that is being filtered.

  - **schema**
    
    The schema of the source information from the THL that is being filtered.

  - **table**
    
    The table of the source information from the THL that is being filtered.

  - **column**
    
    The column of the source information from the THL that is being filtered.

  - **seqno**
    
    The sequence number of the event from the THL that is being filtered.

  - **fragments**
    
    The number of fragments in the THL that is being filtered.

  - **fragment**
    
    The fragment number within the THL that is being filtered. The fragments may be sent individually and sequentially to the network server, so they may need to be retrieved, merged, and reconstituted depending on the nature of the source data and the filter being applied.

  - **transformation**
    
    The transformation to be performed on the supplied payload data. A single network server can support multiple transformations, so this information is provided to perform the corrupt operation. The actual transformation to be performed is taken from the JSON configuration file for the filter.

  - **payload**
    
    The payload.
The length, in bytes, of the payload data that will immediately follow the JSON filter request.

The payload that immediately follows the JSON block is the data from the column that should be processed by the network filter.

The response package should contain a copy of the supplied information from the requested filter, with the payload size updated to the size of the returned information, the message type changed to `filtered`, and the payload containing the translated data. For example:

```json
{
  "transformation" : "String_to_HEX_v1",
  "fragments" : 1,
  "type" : "filtered",
  "fragment" : 1,
  "return" : 0,
  "schema" : "hexdb",
  "table" : "hextable",
  "service" : "firstrep",
  "protocol" : "v0_9",
  "payload" : 8,
  "column" : "hexcol",
  "row" : 0
}
```

### 10.4.28.3. Sample Network Client

The following sample network server script is written in Perl, and is designed to translated packed hex strings [two-hex characters per byte] from their hex representation into their character representation.

```perl
#!/usr/bin/perl
use Switch;
use IO::Socket::INET;
use JSON qw( decode_json encode_json);
use Data::Dumper;
# auto-flush on socket
$| = 1;
my $serverName = "Perl_BLOB_to_String_v1";
while(1)
{
  # creating a listening socket
  my $socket = new IO::Socket::INET (LocalHost => '0.0.0.0',
    LocalPort => '3112',
    Proto => 'tcp',
    Listen => 5,
    Reuse => 1);
  die "Cannot create socket $!

  print "********
Server waiting for client connection on port 3112
******


While(1)
{
  # waiting for a new client connection
  my $client_socket = $socket->accept();
  my $client_address = $client_socket->peerhost();
  my $client_port = $client_socket->peerport();
  print "Connection from $client_address:$client_port

  my $data = ""
  while( $data = $client_socket->getline())
  {
    # Read up to 1024 characters from the connected client
    chomp($data);,
    print "$inReceived: <$data>

    # Decode the JSON part
    my $msg = decode_json($data);

    # Extract payload
    my $payload = undef;
    if ($msg->{payload} > 0)
    {
      print STDERR "Reading $msg->{payload} bytes\n";
      $client_socket->read($payload,$msg->{payload});
      print "$payload\n";
```
10.4.29. **nocreatedbifnotexists.js** Filter

The **nocreatedbifnotexists** filter removes statements that start with:

```
CREATE DATABASE IF NOT EXISTS
```

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>nocreatedbifnotexists</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/nocreatedbifnotexists.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.nocreatedbifnotexists</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>q-to-dbms</td>
</tr>
<tr>
<td>ipm Option compatibility</td>
<td>--svc-applier-filters [473]</td>
</tr>
</tbody>
</table>
Replication Filters

<table>
<thead>
<tr>
<th>Data compatibility</th>
<th>Any event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
</tr>
</tbody>
</table>

This can be useful in heterogeneous replication where Tungsten Replicator specific databases need to be removed from the replication stream.

The filter works in two phases. The first phase creates a global variable within the `prepare()` function that defines the string to be examined:

```java
function prepare()
{
    beginning = "CREATE DATABASE IF NOT EXISTS";
}
```

Row based changes can be ignored, but for statement based events, the SQL is examine and the statement removed if the SQL starts with the text in the `beginning` variable:

```java
sql = d.getQuery();
if(sql.startsWith(beginning))
{
    data.remove(i);
    i--;
}
```

### 10.4.30. OptimizeUpdates Filter

The `optimizeupdates` filter works with row-based events to simplify the update statement and remove columns/values that have not changed. This reduces the workload and row data exchanged between replicators.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>optimizeupdates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.OptimizeUpdatesFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.optimizeupdates</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>Row events</td>
</tr>
<tr>
<td>lpm Option compatibility</td>
<td>[none]</td>
</tr>
</tbody>
</table>

The filter operates by removing column values for keys in the update statement that do not change. For example, when replicating the row event from the statement:

```
mysql> update testopt set msg = 'String1', string = 'String3' where id = 1;
```

Generates the following THL event data:

```
- SQL(0) =
  - ACTION = UPDATE
  - SCHEMA = test
  - TABLE = testopt
  - ROW# = 0
  - COL(1: id) = 1
  - COL(2: msg) = String1
  - COL(3: string) = String3
  - KEY(1: id) = 1
```

Column 1 `[id]` in this case is automatically implied by the KEY entry required for the update.

With the `optimizeupdates` filter enabled, the data in the THL is simplified to:

```
- SQL(0) =
  - ACTION = UPDATE
  - SCHEMA = test
  - TABLE = testopt
  - ROW# = 0
  - COL(2: msg) = String1
  - COL(3: string) = String4
  - KEY(1: id) = 1
```

In tables where there are multiple keys the stored THL information can be reduced further.
### Warning

The filter works by comparing the value of each KEY and COL entry in the THL and determining whether the value has changed or not. If the number of keys and columns do not match then the filter will fail with the following error message:

```plaintext
Caused by: java.lang.Exception: Column and key count is different in this event! Cannot filter
```

This may be due to a filter earlier within the filter configuration that has optimized or simplified the data. For example, the `pkey` filter removes KEY entries from the THL that are not primary keys, or `dropcolumn` which drops column data.

### 10.4.31. 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>lpm Option compatibility</td>
<td>repl.svc-extractor-filters</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
<tr>
<td>Keeps Cached Data</td>
<td>Yes</td>
</tr>
<tr>
<td>Cached Refreshed When?</td>
<td>Emptied when going OFFLINE; Updated when ALTER statement seen</td>
</tr>
<tr>
<td>Metadata Updated</td>
<td>Yes; tungsten_filter_primarykey=true</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
</tr>
<tr>
<td>user</td>
<td>string</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
</tr>
<tr>
<td>addPkeyToInsert</td>
<td>boolean</td>
</tr>
<tr>
<td>addColumnsToDeletes</td>
<td>boolean</td>
</tr>
<tr>
<td>custompkey</td>
<td>filename</td>
</tr>
</tbody>
</table>

**Note**

This filter is designed to be used for testing and with heterogeneous replication where the field name information can be used to construct and build target data structures.

For example, in the following THL fragment, the key information includes data for all columns, which is the default behavior for UPDATE and DELETE Operations.

```
SEQ# = 142 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:31:04.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:0000000000022187:0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
```

506
Replication Filters

- 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:

SEQ# = 142 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:31:04.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:0000000000022187:0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
- ACTION = UPDATE
- SCHEMA = test
- TABLE = salesadv
- ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 1
  - COL(3: city) = 8374
  - COL(4: salesman) = 1
  - COL(5: value) = 89000.00
- KEY(1: id) = 2

The final line shows the addition of the primary key td added to THL event.

Important

The filter determines primary key information by examining the DDL for the table, and keeping that information in an internal cache. If the DDL for a table is not known, or an ALTER TABLE statement is identified, the cache information is updated before the THL is then modified with the primary key information.

In the situation where you enable the filter, but have not create primary key information on the tables, it is possible that creating or adding other index types [such as UNIQUE] on a table, could lead to the incorrect primary key information being updated in the THL, particularly if there are active transactions taking place during and/or immediately after the ALTER statement.

The safest way to perform an index update in case remains the same as for any safe DDL update:

- Put the replicator offline
- Change the DDL for the table or tables
- Put the replicator online

The two options, addPkeyToInsert and addColumnsToDeletes add the primary key information to INSERT and DELETE operations respectively. In a heterogeneous environment, these options should be enabled to prevent full-table scans during update and deletes.

10.4.31.1. Setting Custom Primary Key Definitions

6.0.0 or later. Custom primary key configuration support available in 6.0.0 or later.

Not all tables and databases set or provide explicit primary key information, and in some cases, it is not possible to change the index definition on the source table to include primary key information. Without primary key information in the THL, replicating data into heterogeneous targets can fail, because there is no way for the target environment to correctly identify the primary key information, and therefore the specific record or records to be updated.

The pkeys filter supports defining custom columns to make up a primary key in this situation, avoiding the need to explicitly set index information within the database.
The custom primary key setting works as follows:

- When processing a THL entry, if the custom primary key configuration file has been set and exists, and the incoming schema/table name has been defined in the configuration file, the custom pkey configuration is used.
- Otherwise, the primary key information is taken from the database if it exists.

If neither of these conditions is met, then no primary key data is added to the THL during process.

To configure a custom primary key for one or more tables:

1. Copy the sample primary key configuration file, located within the distribution as `tungsten/tungsten-replicator/support/filters-config/custompkey.json` into the `share` directory within the installation. For example, `/opt/continuent/share`.

2. Update the configuration to include the specific primary key settings for the incoming table. The format of the file is JSON, using a structured layout based on the common JSON filter configuration format (see Section 10.5, “Standard JSON Filter Configuration”). The sample file contains an example for the schema `test` and the table `msg`:

   ```json
   {
     "__default": {
       "IGNORE": "pkey"
     },
     "test": {
       "msg": {
         "msg": "pkey"
       }
     }
   }
   ``

   In the above example, `msg` is the name of the column to be specified as the primary key. The `pkey` indicates that this column must be a primary key field. You can also specify multiple columns:

   ```json
   {
     "__default": {
       "IGNORE": "pkey"
     },
     "test": {
       "msg": {
         "id": "pkey",
         "msg": "pkey"
       }
     }
   }
   ``

   To include another table within the same schema:

   ```json
   {
     "__default": {
       "IGNORE": "pkey"
     },
     "test": {
       "msg": {
         "msg": "pkey"
       },
       "orders": {
         "orderid": "pkey"
       }
     }
   }
   ```

   **Note**

   The `__default` section must remain in place, although it has no impact on processing, it is used to ensure the file is valid for the filter configuration.

3. Update the configuration of your installation through `tpm` to specify the custom primary key file in the properties:

   ```shell
   tpm update alpha --property=replicator.filter.pkey.custompkey=/opt/continuent/share/custompkey.json
   ```

   Once this process has been completed, the replicator will add the custom primary key fields to the THL during processing. For example:

   ```sql
   SQL(0) =
   ACTION = INSERT
   SCHEMA = test
   TABLE = msg
   ROW# = 0
   COL(1: id) = 6
   COL(2: msg) = new test
   ```
10.4.32. PrintEvent Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>printevent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.PrintEventFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.printevent</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>tpmp Option compatibility</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Parameters</td>
<td>(none)</td>
</tr>
</tbody>
</table>

10.4.33. Rename Filter

The rename filter enables schemas to be renamed at the database, table and column levels, and for complex combinations of these renaming operations. Configuration is through a CSV file that defines the rename parameters. A single CSV file can contain multiple rename definitions. The rename operations occur only on ROW based events.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>rename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.RenameFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.rename</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>Row events; Schema names of Statement events in 2.2.1 and later.</td>
</tr>
<tr>
<td>tpmp Option compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>(none)</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 `tpmp update`] must be initiated to force reconfiguration.

To enable using the default CSV file:

```
shell> ./tools/tpmp update alpha --svc-applier-filters=rename
```

The CSV consists of multiple lines, one line for each rename specification. Comments are supposed using the `#` character.

The format of each line of the CSV is:

```
originalSchema,originalTable,originalColumn,newSchema,newTable,newColumn
```

Where:

- `originalSchema`, `originalTable`, `originalColumn` define the original schema, table and column.
- Definition can either be:
  - Explicit schema, table or column name
  - `*` character, which indicates that all entries should match.
- `newSchema`, `newTable`, `newColumn` define the new schema, table and column for the corresponding original specification.
- Definition can either be:
  - Explicit schema, table or column name
  - `*` character, which indicates that the corresponding object should not be updated.
For example, the specification:

```
*,chicago,*,newyork,-
```

Would rename the table `chicago` in every database schema to `newyork`. The schema and column names are not modified.

The specification:

```
*,chicago,destination,-,-,source
```

Would match all schemas, but update the column `destination` in the table `chicago` to the column name `source`, without changing the schema or table name.

Processing of the individual rules is executed in a specific order to allow for complex matching and application of the rename changes.

- Rules are case sensitive.
- Schema names are looked up in the following order:
  1. `schema.table` (explicit schema/table)
  2. `schema.*` (explicit schema, wildcard table)
- Table names are looked up in the following order:
  1. `schema.table` (explicit schema/table)
  2. `*.table` (wildcard schema, explicit table)
- Column names are looked up in the following order:
  1. `schema.table` (explicit schema/table)
  2. `schema.*` (explicit schema, wildcard table)
  3. `*.table` (wildcard schema, explicit table)
  4. `*.*` (wildcard schema, wildcard table)
- Rename operations match the first specification according to the above rules, and only one matching rule is executed.

### 10.4.33.1. Rename Filter Examples

When processing multiple entries that would match the same definition, the above ordering rules are applied. For example, the definition:

```
asia,*,-,america,-,-
asia,shanghai,*,europe,-,-
```

Would rename `asia.shanghai` to `europe.shanghai`, while renaming all other tables in the schema `asia` to the schema `america`. This is because the explicit `schema.table` rule is matched first and then executed.

Complex renames involving multiple schemas, tables and columns can be achieved by writing multiple rules into the same CSV file. For example given a schema where all the tables currently reside in a single schema, but must be renamed to specific continents, or to a ’miscellaneous’ schema, while also updating the column names to be more neutral would require a detailed rename definition.

Existing tables are in the schema `sales`:

- Chicago
- Newyork
- London
- Parts
- Munich
- Moscow
- Tokyo
- Shanghai
- Sydney

Need to be renamed to:

- Northamerica.chicago
- Northamerica.newyork
- Europe.london
- Europe.parts
- Europe.munich
- Misc.moscow
Meanwhile, the table definition needs to be updated to support more complex structure:

<table>
<thead>
<tr>
<th>id</th>
<th>area</th>
<th>country</th>
<th>city</th>
<th>value</th>
<th>type</th>
</tr>
</thead>
</table>

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
```

### 10.4.34. ReplicateColumns Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>replicatecolumns</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.replicatecolumns</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td>Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td></td>
</tr>
<tr>
<td>Row events</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>ignore</td>
<td>string</td>
<td>empty</td>
<td>Comma separated list of tables and optional columns names to ignore during replication</td>
</tr>
<tr>
<td>do</td>
<td>string</td>
<td>empty</td>
<td>Comma separated list of tables and optional column names to replicate</td>
</tr>
</tbody>
</table>

### 10.4.35. Replicate Filter

The *replicate* filter enables explicit inclusion or exclusion of tables and schemas. Each specification supports wildcards and multiple entries.
Replication Filters

| Classname                                      | com.continuent.tungsten.replicator.filter.ReplicateFilter |
| Property prefix                               | replicator.filter.replicate                               |
| Stage compatibility                           | Any                                                        |
| **tpm** Option compatibility                  |                                                            |
| Data compatibility                            | Any event                                                  |
| Parameters                                    |                                                            |
| Parameter | Type     | Default | Description                                      |
| ignore    | string    | empty   | Comma separated list of database/tables to ignore during replication |
| do        | string    | empty   | Comma separated list of database/tables to replicate |

Rules using the supplied parameters are evaluated as follows:

- When both `do` and `ignore` are empty, updates are allowed to any table.
- When only `do` is specified, only the schemas (or schemas and tables) mentioned in the list are replicated.
- When only `ignore` is specified, all schemas/tables are replicated except those defined.

For each parameter, a comma-separated list of schema or schema and table definitions are supported, and wildcards using `*` [any number of characters] and `?` [single character] are also honored. For example:

- `do=sales`
  Replicates only the schema `sales`.
- `ignore=sales`
  Replicates everything, ignoring the schema `sales`.
- `ignore=sales.*`
  Replicates everything, ignoring the schema `sales`.
- `ignore=sales.quarter?`
  Replicates everything, ignoring all tables within the `sales` schema starting with `sales.quarter` and a single character. This would ignore `sales.quarter1` but replicate `sales.quarterlytotals`.
- `ignore=sales.quarter*`
  Replicates everything, ignoring all tables in the schema `sales` starting with `quarter`.
- `do=*_quarter`
  Replicates only the table named `quarter` within any schema.
- `do=sales.*totals,invoices`
  Replicates only tables in the `sales` schema that end with `totals`, and the entire `invoices` schema.

10.4.36. **Row Add Database Name Filter**

The `rowadddbname` filter adds a new column to every incoming row of data containing the schema name of the table. This can be used in combination with analytics replication targets where the information is being written into a single schema, concentrating data from multiple source databases into a single database. Optionally, the filter can also add the information as a primary key (required for some heterogeneous targets), and the hash of the source database name.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>rowadddbname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.JavaScriptFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.rowadddbname</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>Any</td>
</tr>
<tr>
<td><strong>tpm</strong> Option compatibility</td>
<td></td>
</tr>
</tbody>
</table>
Replication Filters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addddbhash</td>
<td>boolean</td>
<td>true</td>
<td>Add a hash column, computed using the Java hash value of the string for the incoming source database name</td>
</tr>
<tr>
<td>addkey</td>
<td>boolean</td>
<td>true</td>
<td>Add the information to the primary key data in the THL, in addition to the column data</td>
</tr>
<tr>
<td>fieldname</td>
<td>string</td>
<td>dbname</td>
<td>The name of the database name column that will be added</td>
</tr>
<tr>
<td>fieldhashname</td>
<td>string</td>
<td>dbname</td>
<td>The name of the database has name column that will be added</td>
</tr>
</tbody>
</table>

The `rowadddbname` filter adds a column to every row of THL processed by the filter. This can be used when data is being written into a single target schema within an analytics environment, and where the source database can be used to identify a customer, project or dataset, and therefore queried within the analytics platform either specifically or with other datasets.

The filter is able to perform the following modifications to every row of incoming data:

- Add the source database or schema name.
- Add a numerical hash value of the string of the source database of schema name.
- Add the database name (and hash name) to the primary key data.

For example, the source THL:

```
- ROW# = 0
  - COL(1: id) = 12
  - COL(2: msg) = Hello
  - COL(3: msg2) = World
  - KEY(1: id) = NULL
```

And after the filter has been applied:

```
- ROW# = 0
  - COL(1: id) = 12
  - COL(2: msg) = Hello
  - COL(3: msg2) = World
  - COL(4: dbname) = msg
  - COL(5: dbname_hash) = 108417
  - KEY(1: id) = NULL
  - KEY(4: dbname) = NULL
  - KEY(5: dbname_hash) = NULL
```

This filter is a required component of deployments when replicating into a single schema within Vertica [see Deploying the Vertica Applier (in [Tungsten Replicator 6.1 Manual])] and Amazon Redshift [see Deploying the Amazon Redshift Applier (in [Tungsten Replicator 6.1 Manual])].

10.4.37. Row Add Transaction Info Filter

The `rowaddtxninfo` filter examines an entire row-based event within the THL and then builds a list of transaction information which is embedded into the metadata for the event. This can be combined with functionality in corresponding appliers, such as Deploying the Kafka Applier (in [Tungsten Replicator 6.1 Manual]) which can then us and embedded into messages or document-based databases.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>rowaddtxninfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.JavaScriptFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.rowaddtxninfo</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>Any</td>
</tr>
<tr>
<td>Otpm Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row-events only</td>
</tr>
</tbody>
</table>

The `rowaddtxninfo` filter processes an entire transaction to determine the following information:
Replication Filters

- List of schemas and tables affected by the transaction.
- A count of the rows inserted, updated, or deleted per schema/table combination.
- A total count of all the rows inserted, updated, or deleted across the entire transaction.

Once the information about the transaction has been collected, the data is then formulated into key/value pairs that are incorporated into the metadata for the entire THL event.

For example, when inserting information into the same table across three separate statements but the same overall transaction, the THL with the filter enabled looks like the example below:

```shell
thl list -last
SEQ# = 162 / FRAG# = 0 (last frag)
- TIME = 2018-03-02 08:47:14.0
- EPOCH# = 162
- SOURCEID = trfiltera
- METADATA = [mysql_server_id=366;dbms_type=mysql;tz_aware=true;strings=utf8;service=alpha;shard=msg;tungsten_filter_columnname=true;
  » tungsten_filter_primarykey=true;tungsten_filter_enumtostring=true;
  » txinfo_rowcount_msg.msg=1;txinfo_rowcount_msg.msgsub=2;txinfo_rowcount=3]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1, time_zone = '+00:00']
```

Examining the metadata more closely you can see the transaction information:

```... txinfo_rowcount_msg.msg=1;txinfo_rowcount_msg.msgsub=2;txinfo_rowcount=3 ...
```

This can be further extrapolated as:

- `txinfo_rowcount_msg.msg=1`
  
  One row has been updated in the `msg` schema/table.

- `txinfo_rowcount_msg.msgsub=2`
  
  Two rows has been updated in the `msg.submsg` schema/table.

- `txinfo_rowcount=3`
  
  A total of three rows have been updated within the transaction.

**Note**

If you transactional information is needed within a Kafka deployment, this filter must have been enabled for the transaction information to be included. See Optional Configuration Parameters for Kafka [in [Tungsten Replicator 6.1 Manual]], for more information.

10.4.38. SetToString Filter

The SetToString converts the SET Column type from the internal representation to a string-based representation in the THL. This achieved by accessing the extractor database, obtaining the table definitions, and modifying the THL data before it is written into the THL file.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>Classname</th>
</tr>
</thead>
<tbody>
<tr>
<td>settostring</td>
<td>com.continuent.tungsten.replicator.filter.SetToStringFilter</td>
</tr>
</tbody>
</table>
Replication Filters

Property prefix: replicator.filter.settostring

Stage compatibility: binlog-to-q

Option compatibility: --repl-svc-extractor-filters [473]

Data compatibility: Row events

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>string</td>
<td>${replicator.global.extract.db.user}</td>
<td>The username for the connection to the database for looking up column definitions</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>${replicator.global.extract.db.password}</td>
<td>The password for the connection to the database for looking up column definitions</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>jdbc:mysql:thin://${replicator.global.extract.db.host}:${replicator.global.extract.db.port}/${replicator.schema}?createDatabase=true</td>
<td>JDBC URL of the database connection to use for looking up column definitions</td>
</tr>
</tbody>
</table>

The SetToString filter should be used with heterogeneous replication to ensure that the data is represented as the string value, not the internal numerical representation.

In the THL output below, the table has a SET column, salesman:

```
mysql> describe salesadv;
+----------+--------------------------------------+------+-----+---------+----------------+
| Field    | Type                                 | Null | Key | Default | Extra          |
+----------+--------------------------------------+------+-----+---------+----------------+
| id       | int(11)                              | NO   | PRI | NULL    | auto_increment |
| country  | enum('US','UK','France','Australia') | YES  |     | NULL    |                |
| city     | int(11)                              | YES  |     | NULL    |                |
| salesman | set('Alan','Zachary')                | YES  |     | NULL    |                |
| value    | decimal(10,2)                        | YES  |     | NULL    |                |
+----------+--------------------------------------+------+-----+---------+----------------+
```

When extracted in the THL, the representation uses the internal value (for example, 1 for the first element of the set description). This can be seen in the THL output below.

```
SEQ# = 138 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:09:35.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:0000000000021434;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
  - ACTION = INSERT
  - SCHEMA = test
  - TABLE = salesadv
  - ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 1
  - COL(3: city) = 8374
  - COL(4: salesman) = 1
  - COL(5: value) = 35000.00
```

For the salesman column, the corresponding value in the THL is 1. With the SetToString filter enabled, the value is expanded to the corresponding string value:

```
SEQ# = 121 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:05:14.0
- EPOCH# = 102
- EVENTID = mysql-bin.000012:0000000000018866;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
  - ACTION = INSERT
  - SCHEMA = test
  - TABLE = salesadv
  - ROW# = 0
  - COL(1: id) = 1
  - COL(2: country) = US
  - COL(3: city) = 8374
  - COL(4: salesman) = Alan
  - COL(5: value) = 35000.00
```
The examples here also show the Section 10.4.21, “EnumTostring Filter” and Section 10.4.8, “ColumnName Filter” filters.

10.4.39. 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></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>enabled</td>
<td>boolean</td>
</tr>
<tr>
<td>unknownShardPolicy</td>
<td>string</td>
</tr>
<tr>
<td>unwantedShardPolicy</td>
<td>string</td>
</tr>
<tr>
<td>enforcedHome</td>
<td>boolean</td>
</tr>
<tr>
<td>allowWhitelisted</td>
<td>boolean</td>
</tr>
<tr>
<td>autoCreate</td>
<td>boolean</td>
</tr>
</tbody>
</table>

10.4.40. shardbyseqno.js Filter

Shards within the replicator enable data to be parallelized when they are applied on the slave.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>shardbyseqno</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/shardbyseqno.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.shardbyseqno</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>q-to-dbms</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>--svc-applier-filters [473]</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>shards</td>
<td>numeric</td>
</tr>
</tbody>
</table>

The shardbyseqno filter updates the shard ID, which is embedded into the event metadata, by a configurable number of shards, set by the shards parameter in the configuration:

replicator.filter.shardbyseqno=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.shardbyseqno.script=${replicator.home}/samples/extensions/javascript/shardbyseqno.js
replicator.filter.shardbyseqno.shards=10

The filter works by setting the shard ID in the event using the setShardId() method on the event object:

```javascript
event.setShardId(event.getSeqno() % shards);
```

Note
Care should be taken with this filter, as it assumes that the events can be applied in a completely random order by blindly updating the shard ID to a computed valued. Sharding in this way is best used when provisioning new slaves.

10.4.41. shardbytable.js Filter

An alternative to sharding by sequence number is to create a shard ID based on the individual database and table. The shardbytable filter achieves this at a row level by combining the schema and table information to form the shard ID. For all other events, including statement based events, the shard ID #UNKNOWN is used.
### Replication Filters

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>shardbytable</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/shardbytable.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.shardbytable</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>q-to-dbs</td>
</tr>
<tr>
<td>option compatibility</td>
<td>--svc-applier-filters</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>definitionsFile</td>
<td>string</td>
<td>support/filters-config/shardbytable.js</td>
<td>JSON file containing the definition of which events and which tables to skip</td>
</tr>
</tbody>
</table>

The key part of the filter is the extraction and construction of the ID, which occurs during row processing:

```java
oneRowChange = rowChanges.get(j);
schemaName = oneRowChange.getSchemaName();
tableName = oneRowChange.getTableName();

id = schemaName + "_" + tableName;
if (proposedShardId == null)
{
    proposedShardId = id;
}
```

### 10.4.42. SkipEventByType Filter

The `SkipEventByType` filter enables you to skip individual events based on the event type, schema and table. For example, if you want to skip all `DELETE` events on the schema/table `SALES.INVOICES` (to prevent deletion of invoice data), this filter will skip the event entirely and it will not be applied to the target.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>skipeventbytype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.SkipEventByTypeFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.skipeventbytype</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>any</td>
</tr>
<tr>
<td>option compatibility</td>
<td>--repl-svc-extractor-filters, --repl-svc-applier-filters</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
</tbody>
</table>

### 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>support/filters-config/skipeventbytype.json</td>
<td>JSON file containing the definition of which events and which tables to skip</td>
</tr>
</tbody>
</table>

Configuration of the filter is made using the generic JSON file, which supports both default options to happen for all tables not otherwise explicitly specified. The default JSON file allows all operations:

```json
{
    "__default": {
        "INSERT": "allow",
        "DELETE": "allow",
        "UPDATE": "allow"
    },
    "SCHEMA": {
        "TABLE": {
            "INSERT": "allow",
            "DELETE": "deny",
            "UPDATE": "deny"
        }
    }
}
```

The default section handles the default response when an explicit schema or table name does not appear. Further sections are then organised by schema and then table name. Where the setting is `allow`, the operation will be processed. A `deny` skips the entire event.

To disable all `DELETE` operations, regardless of which table they occur in:

```json
{
    "__default": {
        "INSERT": "allow",
        "DELETE": "deny",
        "UPDATE": "allow"
    }
}
```
Replication Filters

```
"SCHEMA" : {
  "TABLE" : {
    "INSERT" : "allow",
    "DELETE" : "deny",
    "UPDATE" : "deny"
  }
}
```

To normally allow all operations, except on the SALES.INVOICE schema/table:

```
{"__default": {
  "INSERT" : "allow",
  "DELETE" : "allow",
  "UPDATE" : "allow"}
},  
"SALES" : {
  "INVOICE" : {
    "INSERT" : "allow",
    "DELETE" : "deny",
    "UPDATE" : "deny"
  }
}
```

The configuration also supports single wildcard operations, for example:

```
{"__default": {
  "INSERT" : "allow",
  "DELETE" : "allow",
  "UPDATE" : "allow"}
},  
"SALES" : {
  "*" : {
    "INSERT" : "deny",
    "DELETE" : "allow",
    "UPDATE" : "allow"
  }
}
```

Disables INSERT events for all tables within the SALES schema.

10.4.43. TimeDelay (delay) Filter

The TimeDelayFilter delays writing events to the THL and should be used only on slaves in the remote-to-thl stage. This delays writing the transactions into the THL files, but allows the application of the slave data to the database to continue without further intervention.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.TimeDelayFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.delay</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>remote-to-thl</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>repl_svc-thl-filters [474]</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>delay</td>
<td>numeric</td>
<td>300</td>
<td>Number of seconds to delay transaction processing row</td>
</tr>
</tbody>
</table>

The TimeDelay delays the application of transactions recorded in the THL. The delay can be used to allow point-in-time recovery of DML operations before the transaction has been applied to the slave, or where data may need to be audited or checked before transactions are committed.

**Note**

For effective operation, master and slaves should be synchronized using NTP or a similar protocol.

To enable the TimeDelayFilter, use `tpm` command to enable the filter operation and the required delay. For example, to enable the delay for 900 seconds:
Replication Filters

```shell
./tools/tpm update alpha --hosts=host1,host2,host3
   --repl-svc-applier-filters=delay
   --property=replicator.filter.delay.delay=900
```

Time delay of transaction events should be performed with care, since the delay will prevent a slave from being up to date compared to the master. In the event of a node failure, an up to date slave is required to ensure that data is safe.

### 10.4.44. `tosingledb.js` Filter

This filter updates the replicated information so that it goes to an explicit schema, as defined by the user. The filter can be used to combine multiple tables to a single schema.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>tosingledb</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/tosingledb.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.ansiquotes</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>q-to-dbms</td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td><code>--svc-applier-filters</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>
<tbody>
<tr>
<td><code>db</code></td>
<td>string</td>
<td>[none]</td>
<td>Database name into which to replicate all tables</td>
</tr>
<tr>
<td><code>skip</code></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:

```shell
--property=replicator.filter.tosingledb.db=centraldb
--property=replicator.filter.tosingledb.skip=tungsten
```

The above configures all data to be written into `centraldb`, but skips the database `tungsten`.

Similar to other filters, the filter operates by explicitly changing the schema name to the configured schema, unless the skipped schema is in the event data. For example, at a statement level:

```java
if(oldDb!=null && oldDb.compareTo(skip)!=0)
{
    d.setDefaultSchema(db);
}
```

### 10.4.45. `truncatetext.js` Filter

The `truncatetext` filter truncates a MySQL `BLOB` field.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>truncatetext</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/truncatetext.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.truncatetext</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q,q-to-dbms</td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td><code>--svc-extractor-filters</code> <code>--svc-extractor-filters</code></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>length</code></td>
<td>numeric</td>
<td>[none]</td>
<td>Maximum size of truncated field [bytes]</td>
</tr>
</tbody>
</table>

The length is determined by the `length` parameter in the properties:

```java
replicator.filter.truncatetext=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.truncatetext.script=${replicator.home.dir}/samples/extensions/javascript/truncatetext.js
replicator.filter.truncatetext.length=4000
```

Statement-based events are ignored, but row-based events are processed for each volume value, checking the column type, `isBlob()` method and then truncating the contents when they are identified as larger than the configured length. To confirm the type, it is com-
pared against the Java class `com.continuent.tungsten.replicator.extractor.mysql.SerialBlob`, the class for a serialized `BLOB` value. These need to be processed differently as they are not exposed as a single variable.

```java
if (value.getValue() instanceof com.continuent.tungsten.replicator.extractor.mysql.SerialBlob)
{
    blob = value.getValue();
    if (blob != null)
    {
        valueBytes = blob.getBytes(1, blob.length());
        if (blob.length() > truncateTo)
        {
            blob.truncate(truncateTo);
        }
    }
}
```

### 10.4.46. `zerodate2null.js` Filter

The `zerodate2null` filter looks complicated, but is very simple. It processes row data looking for date columns. If the corresponding value is zero within the column, the value is updated to NULL. This is required for MySQL to Oracle replication scenarios.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>zeroDate2null</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/zerodate2null.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.zerodate2null</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>q-to-dbs</td>
</tr>
<tr>
<td>TPM Option compatibility</td>
<td>--svc-applier-filters [473]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The filter works by examining the column specification using the `getColumnSpec()` method. Each column is then checked to see if the column type is a `DATE`, `DATETIME`, or `TIMESTAMP` by looking the type ID using some stored values for the date type.

Because the column index and corresponding value index match, when the value is zero, the column value is explicitly set to NULL using the `setValueNull()` method.

```java
for(j = 0; j < rowChanges.size(); j++)
{
    oneRowChange = rowChanges.get(j);
    columns = oneRowChange.getColumnSpec();
    columnValues = oneRowChange.getColumnValues();
    for (c = 0; c < columns.size(); c++)
    {
        columnSpec = columns.get(c);
        type = columnSpec.getType();
        if (type == TypesDATE || type == TypesTIMESTAMP)
        {
            for (row = 0; row < columnValues.size(); row++)
            {
                values = columnValues.get(row);
                value = values.get(c);
                if (value.getValue() == 0)
                {
                    value.setValueNull()
                }
            }
        }
    }
}
```

### 10.5. Standard JSON Filter Configuration

A number of the filters that are included as part of Tungsten Clustering use a standardised form of configuration file that is designed to be easy to use and familiar, while being flexible enough to support the needs of each filter. For the majority of filter configurations, the core focus of the configuration is based on a 'default' setting, and settings that are specific to a schema or table.

The JSON configuration follows this basic model. The following filters support the use of this JSON configuration file format:

- `convertstringfrommysql`
Replication Filters

- pkey
- skipeventbytype

The basic format of the configuration is a JSON file that is split into two sections:

- A default section, which determines what will happen in the absence of a schema/table specific rule.
- A collection of schema and table specific entries that determine what happens for a specific schema/table combination.

Depending on the filter and use case, the information within both sections can then either be further divided into column-specific information, or the information may be configured as key/value pairs, or objects, to configure individual parts of the filter configuration.

For example, the following configuration file is from the pkey filter:

```json
{
  "__default": {
    "IGNORE": "pkey"
  },
  "test": {
    "msg": {
      "msg": "pkey"
    }
  }
}
```

The above shows the the defaults section, and the schema/table specific section.

Note

Depending on the filter, the default section may merely be a placeholder to indicate the format of the file. The defaults should never be removed.

The sample shows a full schema name, table name, and then column name configuration.

By comparison, the sample below has only schema and table name information, with the configuration within that section being used to define the key/value pairs for specific operations as part of the skipeventbytype filter:

```json
{
  "__default": {
    "INSERT": "allow",
    "DELETE": "allow",
    "UPDATE": "allow"
  },
  "SCHEMA": {
    "TABLE": {
      "INSERT": "allow",
      "DELETE": "deny",
      "UPDATE": "deny"
    }
  }
}
```

The selection and execution of the rules is determined by some specific rules, as detailed in Section 10.5.1, “Rule Handling and Processing” and Section 10.5.2, “Schema, Table, and Column Selection”.

**10.5.1. Rule Handling and Processing**

The processing of the rules and the selection of the tables and appropriate response and operation is configured through the combination of the default and schema/table settings according to explicit rules:

- If the incoming data matches the schema and table (and optionally column) according to the rules, use the configuration information in that section.
- If the schema/table is not specified or does not have explicit configuration, use the configuration within the __defaults section instead.

The default rule is always processed and followed if there is no match for an explicit schema, table, or column definition.

**10.5.2. Schema, Table, and Column Selection**

The format of the JSON configuration and the selection of the schema, table, and column information is in the form of nested structure of JSON objects. The schema first, then the table, then optionally the column within a nested JSON structure. For example:

```json
"test": {
  "msg": {
```

521
In the above example:

- **test** is the schema name
- **msg** is the table name within the **test** schema
- **id** is the column name within the **test.msg** table

For different tables within the same schema, place another entry at the same level:

```json
"test" : {
  "msg" : {
    "id" : "pkey"
  },
  "orders" : {
    "id" : "pkey"
  }
}
```

The above now handles the tables **msg** and **orders** within the **test** schema.

Wildcards are also supported, using the `*` operator. For example:

```json
"orders" : {
  "*" : {  
    "INSERT" : "allow",
    "DELETE" : "deny",
    "UPDATE" : "deny"
  }
}
```

Would match all tables within the **orders** schema. If multiple definitions exist, then the matching operates on the closest match first. For example:

```json
"orders" : {
  "sales" : {
    "INSERT" : "deny",
    "DELETE" : "deny",
    "UPDATE" : "deny"
  },
  "*" : {  
    "INSERT" : "allow",
    "DELETE" : "deny",
    "UPDATE" : "deny"
  }
}
```

In the above, if the schema/table combination **orders.sales** is seen, the rule for that is always used first as it is explicitly stated. Only tables that do not match the wildcards will use the wildcard entry. If neither an explicit schema/table or wildcard exist, the default is used.

### 10.6. JavaScript Filters

In addition to the supplied Java filters, Tungsten Replicator also includes support for custom script-based filters written in JavaScript and supported through the JavaScript filter. This filter provides a JavaScript environment that exposes the transaction information as it is processed internally through an object-based JavaScript API.

The JavaScript implementation is provided through the Rhino open-source implementation. Rhino provides a direct interface between the underlying Java classes used to implement the replicator code and a full JavaScript environment. This enables scripts to be developed that have access to the replicator constructs and data structures, and allows information to be updated, reformatted, combined, extracted and reconstructed.

At the simplest level, this allows for operations such as database renames and filtering. More complex solutions allow for modification of the individual data, such as removing nulls, bad dates, and duplication of information.

**Warning**

If you previously implemented custome filters with older releases of Tungsten Replicator or with the now deprecat-ed, Open Source release, you would have edited the `static-SERVICE.properties` file.

This is no longer a supported method of implementing custom filters, and doing so will break automated upgrades through `tpm`.
To enable custom filters, follow the process here: Section 10.6.2, “Installing Custom JavaScript Filters”

10.6.1. Writing JavaScript Filters

The JavaScript interface to the replicator enables filters to be written using standard JavaScript with a complete object-based interface to the internal Java objects and classes that make up the THL data.

For more information on the Rhino JavaScript implementation, see Rhino.

The basic structure of a JavaScript filter is as follows:

```javascript
// Prepare the filter and setup structures
prepare()
{
}

// Perform the filter process; function is called for each event in the THL
filter(event)
{
 // Get the array of DBMSData objects
 data = event.getData();

 // Iterate over the individual DBMSData objects
 for(i=0;i<data.size();i++)
 {
    // Get a single DBMSData object
    d = data.get(i);

    // Process a Statement Event; event type is identified by comparing the object class type
    if (d = instanceof com.continuent.tungsten.replicator.dbms.StatementData)
    {
        // Do statement processing
    }
    else if (d = instanceof com.continuent.tungsten.replicator.dbms.RowChangeData)
    {
        // Get an array of all the row changes
        rows = data.get(i).getRowChanges();

        // Iterate over row changes
        for(j=0;j<rows.size();j++)
        {
            // Get the single row change
            rowchange = rows.get(j);

            // Identify the row change type
            if (rowchange.getAction() == "INSERT")
            {
                // ....
            }
        }
    }
}
}
```

The following sections will examine the different data structures, functions, and information available when processing these individual events.

10.6.1.1. Implementable Functions

Each JavaScript filter must defined one or more functions that are used to operate the filter process. The `filter()` function must be defined, as it contains the primary operation sequence for the defined filter. The function is supplied the event from the THL as the events are processed by the replicator.

In addition, two other JavaScript functions can optionally be defined that are executed before and after the filter process. Additional, user-specific, functions can be defined within the filter context to support the filter operations.

- `prepare()`

The `prepare()` function is called when the replicator is first started, and initializes the configured filter with any values that may be required during the filter process. These can include loading and identifying configuration values, creating lookup, exception or other reference tables and other internal JavaScript tables based on the configuration information, and reporting the generated configuration or operation for debugging.
• *filter(event)*

The `filter()` function is the main function that is called each time an event is loaded from the THL. The `event` is parsed as the only parameter to the function and is an object containing all the statement or row data for a given event.

• *release()*

The `release()` function is called when the filter is deallocated and removed, typically during shutdown of the replicator, although it may also occur when a processing thread is restarted.

### 10.6.1.2. Getting Configuration Parameters

The JavaScript interface enables you to get two different sets of configuration properties, the filter specific properties, and the general replicator properties. The filter specific properties should be used 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 context of the filter configuration, use the `filterProperties` class with the `getString()` method. For example, the `dbrename` filter uses two properties, `dbsource` and `dbtarget` to identify the database to be renamed and the new name. The definition for the filter within the configuration file might be:

```javascript
replicator.filter.jsdbrename=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.jsdbrename.script=${replicator.home.dir}/support/filters/javascript/dbrename.js
replicator.filter.jsdbrename.dbsource=contacts
replicator.filter.jsdbrename.dbtarget=nyc_contacts
```

Within the JavaScript filter, they are retrieved using:

```javascript
sourceName = filterProperties.getString("dbsource");
targetName = filterProperties.getString("dbtarget");
```

#### Generic Replicator Properties

General properties can be retrieved using the `properties` class and the `getString()` method:

```javascript
master = properties.getString("replicator.thl.remote_uri");
```

### 10.6.1.3. Logging Information and Exceptions

Information about the filtering process can be reported into the standard `trepsvc.log` file by using the `logger` object. This supports different methods according to the configured logging level:

• *logger.info()* — information level entry, used to indicate configuration, loading or progress.

• *logger.debug()* — information will be logged when debugging is enabled, used when showing progress during development.

• *logger.error()* — used to log an error that would cause a problem or replication to stop.

For example, to log an informational entry that includes data from the filter process:

```javascript
logger.info("regexp: Translating string "+ valueString.valueOf());
```

To raise an exception that causes replication to stop, a new `ReplicatorException` object must be created that contains the error message:

```javascript
if(col == null)
{
    throw new com.continuent.tungsten.replicator.ReplicatorException("dropcolumn.js: column name in "+ schema + "." + table + "." + col + " 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 an exception and backtrace within the log.

### 10.6.1.4. Exposed Data Structures

Within the `filter()` function that must be defined within the JavaScript filter, a single `event` object is supplied as the only argument. That event object contains all of the information about a single event as recorded within the THL as part of the replication process. Each event contains metadata information that can be used to identify or control the content, and individual statement and row data that contain the database changes.
Replication Filters

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 \texttt{filter()} function as the \texttt{event} argument is \texttt{ReplDBMSEvent}. The \texttt{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 \texttt{DBMSData} objects. Each \texttt{DBMSData} object contains either a \texttt{StatementData} object (in the case of a statement based event), or a \texttt{RowChangeData} object (in the case of row-based events). For row-based events, there will be one or more \texttt{OneRowChange} objects for each individual row that was changed.

When processing the event information, the data that is processed is live and should be updated in place. For example, when examining statement data, the statement needs only be updated in place, not re-submitted. Statements and rows can also be explicitly removed or added by deleting or extending the arrays that make up the objects.

A basic diagram of the structure is shown in the diagram below:

<table>
<thead>
<tr>
<th>ReplDBMSEvent</th>
<th>DBMSData</th>
<th>StatementData</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBMSData</td>
<td>RowChangeData</td>
<td>OneRowChange [527]</td>
</tr>
<tr>
<td>ReplDBMSEvent</td>
<td>DBMSData</td>
<td>RowChangeData</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A single event can contain both statement and row change information within the list of individual \texttt{DBMSData} events. An event or

### 10.6.1.4.1. \texttt{ReplDBMSEvent} Objects

The base object from which all of the data about replication can be obtained is the \texttt{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>\texttt{getAppliedLatency()}</td>
<td>Returns the latency of the embedded event. See Section E.2.6, “Terminology: Fields \texttt{appliedLatency}”</td>
</tr>
<tr>
<td>\texttt{getData()}</td>
<td>Returns an array of the \texttt{DBMSData} objects within the event</td>
</tr>
<tr>
<td>\texttt{getDBMSEvent()}</td>
<td>Returns the original \texttt{DBMSEvent} object</td>
</tr>
<tr>
<td>\texttt{getEpochNumber()}</td>
<td>Get the Epoch number of the stored event. See THL EPOCH# [606]</td>
</tr>
<tr>
<td>\texttt{getEventId()}</td>
<td>Returns the native event ID. See THL EVENTID [606]</td>
</tr>
<tr>
<td>\texttt{getExtractedTstamp()}</td>
<td>Returns the timestamp of the event.</td>
</tr>
<tr>
<td>\texttt{getFragno()}</td>
<td>Returns the fragment ID. See THL SEQNO [605]</td>
</tr>
<tr>
<td>\texttt{getLastFrag()}</td>
<td>Returns true if the fragment is the last fragment in the event.</td>
</tr>
<tr>
<td>\texttt{getSeqno()}</td>
<td>Returns the native sequence number. See THL SEQNO [605]</td>
</tr>
<tr>
<td>\texttt{getShardId()}</td>
<td>Returns the shard ID for the event.</td>
</tr>
<tr>
<td>\texttt{getSourceId()}</td>
<td>Returns the source ID of the event. See THL SOURCEID [606]</td>
</tr>
<tr>
<td>\texttt{setShardId()}</td>
<td>Sets the shard ID for the event, which can be used by the filter to set the shard.</td>
</tr>
</tbody>
</table>

The primary method used is \texttt{getData()}, which returns an array of the individual \texttt{DBMSData} objects contain in the event:

```javascript
function filter(event) {
  data = event.getData();
  if(data != null)
```
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:

```java
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:

```java
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 10.4.41, "shard-by-table.js Filter".

#### 10.6.1.4.3. StatementData Objects

The `StatementData` class contains information about data that has been replicated as an SQL statement, as opposed to information that is replicated as row-based data.

Processing and filtering statement information relies on editing the original SQL query statement, or the metadata recorded with it in the THL, such as the schema name or character set. Care should be taken when modifying SQL statement data to ensure that you are modifying the right part of the original statement. For example, a search and replace on an SQL statement should be made with care to ensure that embedded data is not altered by the process.

The key methods used for interacting with a `StatementData` object are listed below:
Replication Filters

<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 10.4.12, “dbname.js Filter”.

10.6.1.4.4. RowChangeData Objects

RowChangeData is information that has been written into the THL in row format, and therefore consists of rows of individual data divided into the individual columns that make up each row-based change. Processing of these individual changes must be performed one row at a time using the list of OneRowChange 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 [527] object.</td>
</tr>
<tr>
<td>getRowChanges()</td>
<td>Returns an array list of all the changes as OneRowChange [527] objects.</td>
</tr>
<tr>
<td>setRowChanges(rowChanges)</td>
<td>Sets the row changes within the event using the supplied list of OneRowChange objects.</td>
</tr>
</tbody>
</table>

For example, a typical row-based process will operate as follows:

```java
if (d != null && d instanceof com.continuent.tungsten.replicator.dbms.RowChangeData)
{
    rowChanges = d.getRowChanges();
    for(j = 0; j < rowChanges.size(); j++)
    {
        oneRowChange = rowChanges.get(j);
        // Do row filter
    }
}
```

The OneRowChange [527] 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>
</tbody>
</table>
Replication Filters

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>setName()</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() returns the column specification of the row change. The information is returned as an array of the individual columns and their specification:

```java
columns = oneRowChange.getColumnSpec();
```

For each column specification a ColumnSpec object is returned, which supports the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>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>
</tbody>
</table>

528
## Replication Filters

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>isNotNull()</code></td>
<td>Returns true if the column is configured as NOT NULL</td>
</tr>
<tr>
<td><code>isUnsigned()</code></td>
<td>Returns true if the column is unsigned.</td>
</tr>
<tr>
<td><code>setBlob()</code></td>
<td>Set the column blob specification</td>
</tr>
<tr>
<td><code>setIndex()</code></td>
<td>Set the column index order</td>
</tr>
<tr>
<td><code>setLength()</code></td>
<td>Returns the column length</td>
</tr>
<tr>
<td><code>setName()</code></td>
<td>Set the column name</td>
</tr>
<tr>
<td><code>setNotNull()</code></td>
<td>Set whether the column is configured as NOT NULL</td>
</tr>
<tr>
<td><code>setSigned()</code></td>
<td>Set whether the column data is signed</td>
</tr>
<tr>
<td><code>setType()</code></td>
<td>Set the column type</td>
</tr>
<tr>
<td><code>setTypeDescription()</code></td>
<td>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 OR VARCHAR</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>TEXT OR BLOB</td>
<td>2004</td>
<td>Use <code>isBlob()</code> to identify if the column is a blob or not</td>
</tr>
<tr>
<td>TIME</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>DATETIME OF 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:

```java
values = oneRowChange.getColumnValues();
```

This means that index 0 of the array from `getColumnSpec()` refers to the same column as index 0 of the array for a single row from `getColumnValues()`.

<table>
<thead>
<tr>
<th><code>getColumnSpec()</code></th>
<th>msgid</th>
<th>message</th>
<th>msgdate</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0]</td>
<td>1</td>
<td>Hello New York!</td>
<td>Thursday, June 13, 2013</td>
</tr>
</tbody>
</table>

This enables the script to identify the column type by the index, and then the corresponding value update using the same index. In the above example, the `message` field will always be index 1 within the corresponding values.

Each value object supports the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getValue()</code></td>
<td>Get the current column value</td>
</tr>
<tr>
<td><code>setValue()</code></td>
<td>Set the column value to the supplied value</td>
</tr>
<tr>
<td><code>setValueNull()</code></td>
<td>Set the column value to NULL</td>
</tr>
</tbody>
</table>

For example, within the `zerodate2null` sample, dates with a zero value are set to NULL using the following code:

```java
columns = oneRowChange.getColumnSpec();
columnValues = oneRowChange.getColumnValues();
```
In the above example, the column specification is retrieved to determine which columns are date types. Then the list of embedded row values is extracted, and iterates over each row, setting the value for a date that is zero (0) to be NULL using the `setValueNull()` method.

An alternative would be to update to an explicit value using the `setValue()` method.

### 10.6.2. Installing Custom JavaScript Filters

Once you have written your JavaScript filter, and ready to install it you need to follow the steps below. This will allow you to configure and apply the filter to your installation using the standard `tpm` procedure.

For this example, we will assume your new JavaScript file is called `number2binary.js`, and the filter has two additional boolean configuration properties 'roundup' and 'debug'.

#### 10.6.2.1. Step 1: Copy JavaScript Files

By default, the software package will be contained in `/opt/continuent/software/tungsten-clustering-6.1.3-43` Adjust the path in the examples accordingly if your environment differs.

The JavaScript file for your new filter(s) need copying to the following location:

```
/opt/continuent/software/tungsten-clustering-6.1.3-43/tungsten-replicator/samples/extensions/javascript
```

#### 10.6.2.2. Step 2: Create Template Files

You need to create a template file which contains the location of the JavaScript file and the additional configuration properties with the appropriate default values.

Create a file called `number2binary.tpl` that contains the following:

```
replicator.filter.number2binary=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.number2binary.script=${replicator.home.dir}/samples/extensions/javascript/number2binary.js
replicator.filter.number2binary.roundup=true
replicator.filter.number2binary.debug=false
```

This tpl file needs to be copied into the following directory:

```
/opt/continuent/software/tungsten-clustering-6.1.3-43/tungsten-replicator/samples/conf/filters/default
```

#### 10.6.2.3. Step 3: (Optional) Copy json files

If your filter uses json files to load configuration data, this needs to be copied into the `/opt/continuent/share` directory and also included in the tpl file created in Step 2. An example is as follows:

```
replicator.filter.[FILTERNAME].definitionsFile=/opt/continuent/share/[FILTERNAME].json
```

#### 10.6.2.4. Step 4: Update Configuration

Now that all the files are in place you can include the custom filter in your configuration.

Any properties set with a default value in the tpl file, only need including if you wish to overwrite the default value.

The following examples show how you can now include this in your tpm configuration:

For ini installations add the following to your `tungsten.ini`
Replication Filters

For staging installations

```
shell> cd {staging-dir}
shell> tools/tpm configure SERVICENAME
{other-configuration-values-as-requires} \
--svc-extractor-filters={existing filter definitions},number2binary \ 
--property=replicator.filter.number2binary.roundup=false \ 
--property=replicator.filter.number2binary.debug=true
shell> tools/tpm install
```

In the above examples we used the `svc-extractor-filters` property for the extractor replicator. If you are applying your custom filters to your applier, then use `svc-applier-filters` instead.

Your custom filters are now installed in a clean and easy to manage process allowing you to use `tpm` for all future update processes.

If there is a problem with the JavaScript filter during restart, the replicator will be placed into the `OFFLINE` state and the reason for the error will be provided within the replicator `trepsvc.log` log.
Chapter 11. Performance, Tuning and Testing

To help improve the performance of Tungsten Clustering, a number of guides and tuning techniques are provided in this chapter. This may involve parameters entirely within Tungsten Clustering, or changes and modifications to the parameters within the OS.

Tuning related to the Tungsten Replicator functionality

- Section 11.1, “Block Commit” — Increasing performance of replication solutions making use of block commit.

Tuning related to the network performance

- Section 11.2, “Improving Network Performance” — Increasing performance of networking between components by tuning OS parameters.

11.1. Block Commit

Introduced in 2.0.1. 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 --svc-applier-block-commit-size)
- When the commit timer reaches the specified commit interval (set by --svc-applier-block-commit-interval)

The default operation is for block commits to take place based on the transaction count. Commits by the timer are disabled. The default block commit size is 10 transactions from the incoming stream of THL data; the block commit interval is zero (0), which indicates that the interval is disabled.

When both parameters are configured, block commit occurs when either value limit is reached. For example, if the event count is set to 10 and the commit interval to 50s, events will be committed by the applier either when the event count hits 10 or every 50 seconds, whichever is reached first. This means, for example, that even if only one transaction exists, when the 50 seconds is up, that single transaction will be applied.

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=100s
```

Note

The block commit parameters are supported only in applier stages; they have no effect in other stages.

Modification of the block commit interval should be made only when the commit window needs to be altered. The setting can be particularly useful in heterogeneous deployments where the nature and behaviour of the target database is different to that of the source extractor.

For example, when replicating to Oracle, reducing the number of transactions within commits reduces the locks and overheads:

```
shell> ./tools/tpm update alpha
   --repl-svc-applier-block-commit-interval=500
```

This would apply two commits every second, regardless of the block commit size.

When replicating to a data warehouse engine, particularly when using batch loading, such as Redshift, Vertica and Hadoop, larger block commit sizes and intervals may improve performance during the batch loading process:

```
shell> ./tools/tpm update alpha
   --repl-svc-applier-block-commit-size=100000
   --repl-svc-applier-block-commit-interval=60s
```

This sets a large block commit size and interval enabling large batch loading.

11.1.1. Monitoring Block Commit Status

The block commit status can be monitored using the trepctl status -name tasks command. This outputs the lastCommittedBlockSize and lastCommittedBlockTime values which indicate the size and interval (in seconds) of the last block commit.
11.2. Improving Network Performance

The performance of the network can be critical when replicating data. The information transferred over the network contains the full content of the THL in addition to a small protocol overhead. Improving your network performance can have a significant impact on the overall performance of the replication process.

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:

```bash
# Increase size of file handles and inode cache
fs.file-max = 2097152

# tells the kernel how many TCP sockets that are not attached to any user file handle to maintain. In case this number is exceeded, orphaned connections are immediately reset and a warning is printed.
net.ipv4.tcp_max_orphans = 60000

# Do not cache metrics on closing connections
net.ipv4.tcp_no_metrics_save = 1

# Turn on window scaling which can enlarge the transfer window:
net.ipv4.tcp_window_scaling = 1

# Enable timestamps as defined in RFC1323:
net.ipv4.tcp_timestamps = 1

# Enable select acknowledgments:
net.ipv4.tcp_sack = 1

# Maximum number of remembered connection requests, which did not yet receive an acknowledgment from connecting client.
net.ipv4.tcp_max_syn_backlog = 10240

# recommended default congestion control is htcp
net.ipv4.tcp_congestion_control=htcp

# recommended for hosts with jumbo frames enabled
net.ipv4.tcp_mtu_probing=1

# Number of times SYNACKs for passive TCP connection.
net.ipv4.tcp_sack_retries = 2

# Allowed local port range
net.ipv4.ip_local_port_range = 1024 65535

# Protect Against TCP Time-Wait
net.ipv4.tcp_rfc1337 = 1

# Decrease the time default value for tcp_fin_timeout connection
net.ipv4.tcp_fin_timeout = 15
```
11.3. Enable Heterogeneous Replication from a Cluster

The Tungsten Replicator supports replication into many different DBMS platforms. In order for this to work properly the cluster must be configured correctly. During deployment include the `--enable-heterogeneous-service=true` option in the defaults or service definition for the cluster. Use the `tpm update` command if you have already deployed the cluster.

```
shell > ./tools/tpm configure defaults \n   --enable-heterogeneous-service=true
```

During Tungsten Replicator deployment use the `--topology=cluster-slave` as described in Section 3.8, “Replicating Data Out of a Cluster”.

11.4. Tungsten Replicator Block Commit and Memory Usage

Replicators are implemented as Java processes, which use two types of memory: stack space, which is allocated per running thread and holds objects that are allocated within individual execution stack frames, and heap memory, which is where objects that persist across individual method calls live. Stack space is rarely a problem for Tungsten as replicators rarely run more than 200 threads and use limited recursion. The Java defaults are almost always sufficient. Heap memory on the other hand runs out if the replicator has too many transactions in memory at once. This results in the dreaded Java OutOfMemory exception, which causes the replicator to stop operating. When this happens you need to look at tuning the replicator memory size.

To understand replicator memory usage, we need to look into how replicators work internally. Replicators use a “pipeline” model of execution that streams transactions through 1 or more concurrently executing stages. As you can see from the attached diagram, a slave pipeline might have a stage to read transactions to the master and put them in the THL, a stage to read them back out of the THL into an in-memory queue, and a stage to apply those transactions to the slave. This model ensures high performance as the stages work independently. This streaming model is quite efficient and normally permits Tungsten to transfer even exceedingly large transactions, as the replicator breaks them up into smaller pieces called transaction fragments.
The pipeline model has consequences for memory management. First of all, replicators are doing many things at 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 `replicator.global.buffer.size` in the replicator properties file. This controls the size of in-memory queues in the replicator as well as the block commit size. If you still have problems after increasing the heap size, try reducing this value. It reduces the number of objects simultaneously stored on the Java heap. A value of 2 is a good setting to try to get around temporary problems. This can be set at installation time as the `--repl-buffer-size` parameter.
- Property `replicator.stage.q-to-dbms.blockCommitRowCount` in the replicator properties file. This parameter sets the block commit count in the final stage in a slave pipeline. If you reduce the global buffer size, it is a good idea to set this to a fixed size, such as 10, to avoid reducing the block commit effect too much. Very low block commit values in this stage can cut update rates on slaves by 50% or more in some cases. This is available at installation time as the `--repl-svc-applier-block-commit-size` parameter.
- Property `replicator.extractor.dbms.transaction_frag_size` in the replicator properties file. This parameter controls the size of fragments for long transactions. Tungsten automatically breaks up long transactions into fragments. This parameter controls the number of bytes of bin-log per transaction fragment. You can try making this value smaller to reduce overall memory usage if many transactions are simultaneously present. Normally however this value has minimal impact.

Finally, it is worth mentioning that the main cause of out-of-memory conditions in replicators is large transactions. In particular, Tungsten cannot fragment individual statements or row changes, so changes to very large column values can also result in OutOfMemory conditions. For now the best approach is to raise memory, as described above, and change your application to avoid such transactions.

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 commits to take place based on the transaction count. Commits by the timer are disabled. The default block commit size is 10 transactions from the incoming stream of THL data; the block commit interval is zero (0), which indicates that the interval is disabled.

When both parameters are configured, block commit occurs when either value limit is reached. For example, if the event count is set to 10 and the commit interval to 50s, events will be committed by the applier either when the event count hits 10 or every 50 seconds, whichever is reached first. This means, for example, that even if only one transaction exists, when the 50 seconds is up, that single transaction will be applied.

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=100s
```

**Note**

The block commit parameters are supported only in applier stages; they have no effect in other stages.

Modification of the block commit interval should be made only when the commit window needs to be altered. The setting can be particularly useful in heterogeneous deployments where the nature and behavior 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:

```sh
csh> /tools/tpm update alpha
   --repl-svc-applier-block-commit-interval=500
```

This would apply two commits every second, regardless of the block commit size.

When replicating to a data warehouse engine, particularly when using batch loading, such as Redshift, Vertica and Hadoop, larger block commit sizes and intervals may improve performance during the batch loading process:

```sh
csh> /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.

### 11.5. 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 update` and the `--conn-java-mem-size` option:

For example, during installation:

```sh
csh> tpm install ... --conn-java-mem-size=1024
```

Or to update using `tpm update`:

```sh
csh> tpm update ... --conn-java-mem-size=1024
```

### 11.6. Functional Testing

The following chapter includes a number of suggested functional tests that can be performed following installation in a Testing/POC environment.

For a test to be successful, normal operations should be restored after every test. The default master server should regain master role with all replicators and data sources ONLINE.

The tests are split into the following categories:

- **Section 11.6.1, “Manual and Automatic Failover”**
- **Section 11.6.2, “Backup and Restore”**
- **Section 11.6.3, “Connectivity”**
- **Section 11.6.4, “Performance and Other Tests”**

A PDF version of these tests can be downloaded by clicking here.

#### 11.6.1. Manual and Automatic Failover

**Failover Test 1 - Administrator uses Tungsten to promote new master**

<table>
<thead>
<tr>
<th>Scenario</th>
<th><code>cctrl</code> use usa logical</th>
<th><code>cctrl</code> switch</th>
</tr>
</thead>
</table>

**Expectation**

- The master role will move to a different server. Remaining servers will be reconfigured accordingly.
- Examine how the application performs during the process.
- Data should be consistent.
Performance, Tuning and Testing

Failover Test 2 – Manually kill the master database process

| Scenario | shell> cctrl
cctrl> use usa
[LOGICAL] /usa> service master/mysql stop |
|----------|--------------------------------------------------|
| Expectation | • The database server stops and the master role is moved to another server.
• Examine how the application performs during the process
• Data should be consistent |

Failover Test 3 – Remove power from the master database server

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Pull the power plug on the master server or run a restart command if that is not an option.</th>
</tr>
</thead>
</table>
| Expectation | • The database server stops and the master role is moved to another server.
• Examine how the application performs during the process
• Data should be consistent |

Failover Test 4 – Manually kill a slave database process

| Scenario | shell> cctrl
cctrl> use usa
[LOGICAL] /usa> service slave1/mysql stop |
|----------|--------------------------------------------------|
| Expectation | • The slave data source is marked as FAILED.
• Examine how the application performs during the process |

11.6.2. Backup and Restore

Backup Test 1 – Take a backup of a slave and restore it to the same server

| Scenario | shell> cctrl
cctrl> use usa
[LOGICAL] /usa> datasource slave1 backup
[LOGICAL] /usa> datasource slave1 restore |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Expectation</td>
<td>• The commands should complete successfully.</td>
</tr>
</tbody>
</table>

Backup Test 2 – Restore the backup to another server

| Scenario | shell> rsync -avze ssh /opt/continuent/backups/ slave2:/opt/continuent/backups/
shell> cctrl
cctrl> use usa
[LOGICAL] /usa> datasource slave2 restore |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Expectation</td>
<td>• The commands should complete successfully.</td>
</tr>
</tbody>
</table>

Backup Test 3 – Take a backup of the master and restore to a slave

| Scenario | shell> cctrl
cctrl> use usa
[LOGICAL] /usa> datasource master backup
master> rsync -avze ssh /opt/continuent/backups/ slave2:/opt/continuent/backups/
[LOGICAL] /usa> datasource slave2 restore |
|----------|--------------------------------------------------------------------------------------------|
| Expectation | • The restore command should complete successfully, or.
• Run `trepctl online -from-event #######:########` to bring the slave ONLINE |

11.6.3. Connectivity

Connectivity Test 1 – Connect to the connector and verify master host

<table>
<thead>
<tr>
<th>Scenario</th>
<th>shell&gt; mysql -h hostname -P9999 -uapp -p -e&quot;select @hostname for update&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expectation</td>
<td>• The master hostname is returned.</td>
</tr>
</tbody>
</table>

Connectivity Test 2 – Verify access to slaves
This test is only required if read/write splitting has been enabled. It should be run from a connector running on a server other than the master.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Shell&gt; mysql -h <code>hostname</code> -P 9999 -u app -p -e &quot;select @@hostname&quot;</th>
</tr>
</thead>
</table>
| Expectation | • A hostname is returned.  
| | • Repeat the process on other hosts until a slave hostname is returned |

Connectivity Test 3 – Verify access to the master before and after a switch

| Scenario | Shell> cctrl  
| | cctrl> use usa  
| | [LOGICAL] usa> switch |
| Expectation | • Run Test 1 again and verify the new master hostname is returned. |

### 11.6.4. Performance and Other Tests

**Performance Test 1 – Run a load test against the cluster**

| Scenario | Run load tests of some variety against the cluster to ensure the Connector and Replicator properly handle the load. Recommended load test tools are:  
| | • HammerDB.  
| | • sysbench |
| Expectation | Solution can handle 1TB of data with a minimum of 24k reads per minute and 1k writes per minute, over three tests.  
| | • Insert only from multiple locations  
| | • Read/write test  
| | • Heavy read test with few writes |

**Replicator Testing**

| Scenario | Evaluate system responsiveness in conjunction with performance tests (above). This is not a test of network speed |
| Expectation | Quick replication latency between regions (<100ms greater than the current ping latency) |

**Network Partition Testing**

| Scenario | Simulate a partitioned network (for example, by modifying security group rules in AWS), and continue to do reads and writes on multiple clusters for 30 minutes. |
| Expectation | After resolving the partition, clusters should resync. |

| Scenario | Create a similar network partition, and write the same record on both sides of the partition |
| Expectation | After resolving the partition, the replicators on both sides should report errors. Demonstrate possible resolutions:  
| | • Remove record on side, skip record on the other side, and bring replication back online. The record not removed will be replicated  
| | • Remove both records and skip both transactions  
| | • Modify records and bring replicators online |
Appendix A. Release Notes

A.1. Tungsten Clustering 6.1.3 GA [17 February 2020]

Version End of Life: Not Yet Set

Release 6.1.3 contains a small number of critical bug fixes that can affect customers operating geo-distributed clusters across high latency network links, along with a small number of improvements and fixes to common command line tools.

Behavior Changes

The following changes have been made to Tungsten Clustering 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:

- Improved the tungsten_find_position script to add the ability to specify the low and high sequence numbers which limits the amount of TRL the script needs to parse. This allows for better performance and lower system overhead. Also allows the use of the `-file` option.

```bash
(-f|--file) Pass specified file to the thl command as -file {file}
([-l|--from]) Pass specified seqno to the thl command as -low {seqno}
([-h|--to]) Pass specified seqno to the thl command as -high {seqno}
```

Issues: CT-1143

- `tpm diag`: make the output from remote host diagnostic gathering visible in addition to alerting when a host is not reachable.

Issues: CT-1158

Known Issue

The following issues may affect the operation of Tungsten Clustering and should be taken into account when deploying or updating to this release.

- The backup process, when configured to use Xtrabackup, uses the `--stream=tar` option as one of the options passed to the backup process.

This option is no longer available in Xtrabackup 8.0

If you use Xtrabackup 8.0 in combination with MySQL 8, generating backups using the procedures available in Tungsten Clustering will fail. Until a fix is available and to allow backups to continue you will need to make the following edit to the configuration

After installation, open the `static-servicename.properties` file located in `INSTALL_PATH/tungsten/tungsten_replicator/conf` and within the string value, change the value of `replicator.backup.agent.xtrabackup.options` to `tar=false`

If the replicator is already running, then you will need to issue `replicator restart` for the change to take effect

Warning

Changing the properties file directly will cause future `tpm update` commands to fail, therefore you should run this with the `--force` option, and then reedit the file as per the above instructions to reset the tar option.

Issues: CT-1157

Bug Fixes

- Command-line Tools

  - `tpm diag` would fail to collect diagnostics for relay nodes within a Coposite Master-Slave Topology.

Issues: CT-1140

  - Fixes an edge case whereby `tpm mysql` would fail on a node within a Composite Master/Master topology.

Issues: CT-1151

  - `tpm diag` now gathers all hosts in a staging deployment when run from a non-staging node.
**Issues: CT-1155**

- `tpm diag`: fix ensures collection of diagnostics from standalone connector hosts.

**Issues: CT-1159**

- **Tungsten Manager**

- During a local MASTER switch within a Composite Master-Master Topology, where there is a high latency link between clusters, the switch could intermittently fail due to an incorrect rule triggering as the remote cluster sees an incorrect state in the opposing cluster.

**Issues: CT-1141**

Tungsten Clustering 6.1.3 includes the following changes made in Tungsten Replicator 6.1.3

Release 6.1.3 contains a small number of improvements and fixes to common command line tools, and introduces compatibility with MongoDB Atlas.

**Behavior Changes**

The following changes have been made to Continuent 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:

- `tpm diag` has been updated to provide additional feedback detailing the hosts that were gathered during the execution, and also provides examples of how to handle failures when running on a single host configured via the ini method:

  ```shell
  shell> tpm diag
  Collecting localhost diagnostics.
  Note: to gather for all hosts please use the "-a" switch and ensure that you have passwordless ssh access set between the hosts.
  Diagnostic information written to /home/tungsten/tungsten-diag-2020-02-06-19-34-25.zip
  ```

  When running on a staging host, or with the `-a` flag:

  ```shell
  shell> tpm diag [-a|--allhosts]
  Collecting full cluster diagnostics
  Note: if ssh access to any of the cluster hosts is denied, use "--local" or "--hosts=host1,host2,...".
  Collecting diag information on db1....
  Diagnostic information written to /home/tungsten/tungsten-diag-2020-02-06-19-34-25.zip
  ```

**Issues: CT-1137**

**Bug Fixes**

- **Command-line Tools**

- `tpm` would fail to run on some Operating Systems due to missing `realpath`

  `tpm` has been changed to use `readlink`, which is commonly installed by default on most operating systems, however if it is not available, you may be required to install GNU coreutils to satisfy this dependency.

  **Issues: CT-1124**

- Removed dependency on perl module Time::HiRes from `tpm`

  **Issues: CT-1126**

- Added support for handling missing dependency (Data::Dumper) within various `tpm` subcommands

  **Issues: CT-1130**

- `tpm` will now work on MacOS/X systems, provided `greadlink` is installed.

  **Issues: CT-1147**

- `tpm install` will no longer report that the Linux distribution cannot be determined on SUSE platforms.

  **Issues: CT-1148**
• Fixes a condition where tpm diag would fail to set the working path correctly, especially on Debian 8 hosts.
  
  **Issues:** CT-1150

• tpm diag now checks for OS commands in additional paths (/bin, /sbin, /usr/bin and /usr/sbin)
  
  **Issues:** CT-1160

• Fixes an issue introduced in v6.1.2 where the use of the undeployall script would stop services as it removed them from systemctl control
  
  **Issues:** CT-1166

• Core Replicator

  • The MongoDB Applier has been updated to use the latest MongoDB JDBC Driver
    
    **Issues:** CT-1134

  • The MongoDB Applier now supports MongoDB Atlas as a target
    
    **Issues:** CT-1142

  • The replicator would fail with unknown column 'column' in 'where clause' when replicating between MySQL 8 hosts where the client application would write data into the source database host using a different collation to that of the default on the target database.
    
    The replicator would fail due to a mismatch in these collations when querying the information_schema.columns view to gather metadata ahead of applying to the target
    
    **Issues:** CT-1145

A.2. Tungsten Clustering 6.1.2 GA [20 January 2020]

**Version End of Life.** Not Yet Set

Release 6.1.2 contains significant improvements as well as some needed bugfixes.

**Behavior Changes**

The following changes have been made to Tungsten Clustering 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 Passive Witness functionality is now officially DEPRECATED and will be REMOVED in version 6.2
  
  **Issues:** CT-653

• In cluster deployments with witness nodes, if the MySQL servers had been configured to listen on any port other than the standard 3306, the tpm command would default to the wrong port number.

  When witness hosts are in use, the tpm command will no longer "guess" the data source port automatically by running my_print_defaults.

  It is now mandatory to specify the MySQL data server port explicitly using --datasource-port={mysql_listen_port} or one of its aliases
  
  **Issues:** CT-1071

**Improvements, new features and functionality**

• **Command-line Tools**

  • The tpm command was originally written in Ruby. This improvement converts tpm to Perl over time, starting with the tpm shell wrapper and refactoring each sub-command one-by-one.

    For this release, we have redone the diag, mysql and connector sub-commands.

    This also wraps the update sub-command to provide the CT-1093 clustering fix.

    **Issues:** CT-1048
• This version includes an update to the new BETA tool to provision slaves, and masters in a Composite MultiMaster topology. This release fully supports provisioning of nodes in a Composite Master-Master Topology.

It can be invoked by running prov-sl.sh or tps.pl.

This tool will replace tungsten_provision_slave in a future release.

Issues: CT-1070

• Added the tpm policy subcommand to allow easy get and set cluster policy operations instead of using cctrl.

For more information, please see ...

Issues: CT-1106

• The new changes made to the tpm command, require that the zip package be installed on all DB hosts

Issues: CT-1111

• Tungsten Manager

• The new default datasources sort order is alphabetically when using the cctrl ls command. Additionally, the sort order of the datasources list is now configurable.

The behavior is controlled by the tpm configuration property cctrl.sort.datasources.alphabetically which has a default value of true (meaning alpha sort).

If set to false, the sort is ordered by datasource role, so the master or relay will appear first, followed by the slaves. For example, use the following in the INI file for role-based sorting:

```
property=cctrl.sort.datasources.alphabetically=false
```

Issues: CT-1018

Bug Fixes

• Installation and Deployment

• Ensure that all Connector-Manager communications are SSL-encrypted when --disable-security-controls=false

Issues: CT-1060

• Command-line Tools

• When performing an update of a cluster with tpm, the cluster would be switched to MAINTENANCE but would remain in this policy after the update. The original policy is now retained during the update.

Issues: CT-595, CT-1093

• The deployall script was only able to install init.d system startup scripts.

In this release, the script will now detect the initialization system in use [systemd or initd] and prefer systemd when both are available.

For systemd configurations only:

For continuity of service reasons, the deployall script does NOT restart individual components when called, it will only install systemd scripts. This implies that, right after a call to deployall and before host restart, the system will stay in a mixed mode where systemd scripts are in place but components were started without systemd, so won’t be controllable by it.

In order to align the configuration, you will need to run

```
shell> component stop sysd
shell> sudo systemctl start tcomponent
```

For example:

```
shell> connector stop sysd
shell> sudo systemctl start tconnector
```

Issues: CT-853

• When issuing tpm connector --samples, the output displayed clear text passwords. In this release the passwords are obsfucated.
Release Notes

Issues: CT-1021

- Continuent Tungsten Clustering now uses the xtrabackup command instead of the deprecated innobackupex to create and restore backups. A new check was added to TPM for validating different xtrabackup versions along with MySQL version compatibility. The oldest supported version of xtrabackup is v2.3.

Issues: CT-1074

Core Replicator

- When configuring SSL for the Connector only, the Replicator would fail to start due to the Replicator also looking for the SSL configuration.

Issues: CT-956

Tungsten Connector

- Fixed an issue where some applications might fail to connect to the Connector with MariaDB 10+

  Previously, when using MariaDB 10+, the Connector would be confused by the 10 and will think it is a MySQL 8+ server. By default, the Connector will offer to connect with caching_sha2_password. If the application does not know how to switch authentication plugins, it would fail with a message similar to the following:

  ```
  The server requested authentication method unknown to the client [caching_sha2_password]
  ```

  The previous work-around was to specify the authentication plugin using the tpm command:

  ```
  --property=defaultAuthPlugin=mysql_native_password
  ```

  Issues: CT-1033

- Improved Tungsten Connector bridge mode performance when transferring small amounts of data.

  Issues: CT-1081

Tungsten Manager

- When using the cctrl command interactively, the `cluster topology` TAB completion was showing invalid options. Invalid options have been removed.

  Issues: CT-979

- Fixed an issue where long-duration operations like failover and switch would create false positives about network partitioning after completion.

  Issues: CT-1023

- Fixed the cctrl command so that the `[SSL]` indicator in the `ls` output is displayed. This is a Version 5 feature that was lost in v6.0.0, now restored.

  Issues: CT-1061

- Fixed the cctrl command `datasource {hostname} restore` which was failing in Composite Master/Master cluster deployments with: ERROR: MORE THAN ONE PRIMARY DATA SOURCE FOUND.

  Issues: CT-1062

- Continuent Tungsten Clustering now only checks for a running MySQL server when the backup method is `mysqldump` in cctrl.

  Background: Running `datasource {hostname} restore` inside cctrl would fail when the MySQL server was not running. Only the `mysqldump` method requires a running MySQL server. The `xtrabackup-full` and `xtrabackup-incremental` methods will work even if MySQL is stopped.

  Issues: CT-1077

- tungsten_find_orphaned was displaying an incorrect error message if a service name wasn't supplied correctly.

  Issues: CT-1079

- tungsten_find_orphaned would error with `Argument "" isn't numeric in addition`.

  Issues: CT-1080
• Fixed an issue where Composite clusters with only a single site would come up as SHUNNED after install.
  
  **Issues:** CT-1101

Tungsten Clustering 6.1.2 includes the following changes made in Tungsten Replicator 6.1.2

Release 6.1.2 contains both significant improvements as well as some needed bugfixes.

**Behavior Changes**

The following changes have been made to Continuent 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:

• Certified the Tungsten product suite with Java 11.

  A small set of minor issues have been found and fixed (CT-1091, CT-1076) along with this certification.
  
  The code is now compiled with Java compiler v11 while keeping Java 8 compatibility.
  
  Java 9 and 10 have been tested and validated but certification and support will only cover Long Term releases.

**Note**

**Known Issue**

With Java 11, command line tools are slower. There is no impact on the overall clustering or replication performance but this can affect manual operations using CLI tools such as cctrl and trepctl

**Issues:** CT-1052

**Improvements, new features and functionality**

• Core Replicator

  • A new Replicator role, **thl-server**, has been added.

    This new feature allows your slave replicators to still pull generated THL from a Master even when the Master replicator has stopped extracting from the binlogs.

    If used in Tungsten Clusters, this feature must only be enabled when the cluster is in **MAINTENANCE** mode. For more details refer to ???

    **Issues:** CT-58

  • A new JavaScript filter **dropddl.js** has been added to allow selective removal of specific object DDL from THL.

    **Issues:** CT-1092

**Bug Fixes**

• **Behavior Changes**

  • If you need to reposition the extractor, there are a number of ways to do this, including the use of the options **-from-event** or **-base-seqno**

    Both of these options are mutually exclusive, however in some situations, such as when positioning against an Aurora source, you may need to issue both of these options together. Previously this was not possible. In this release both options can now be supplied providing that you include the additional **-force** option, for example

    ```shell```
    trepctl -service servicename online -base-seqno 33 -from-event 000412:762897 -force
    ```shell```

    **Issues:** CT-1065

  • When the Replicator inserts a heartbeat there is an associated timezone. Previously, the heartbeat would be inserted using the GMT timezone, which fails during the DST switch window. The new default uses the Replicator host's timezone instead.
This defaults change corrects an edge case where inserting a heartbeat will fail during the DST
switch window when the MYSQL server is running in a different timezone than the Replicator
(which runs in GMT).

For example, on 31th March 2019, the time switch occurred @ 2AM in the Europe/Paris time-
zone. When inserting a heartbeat in the window between 4 and 5 AM (say at 4:15am), the cor-
responding GMT time would be 2:15am, which is invalid in the Europe/Paris timezone. Replica-
tor would then fail if the MySQL timezone was set to Europe/Paris, as it would try to insert an
invalid timestamp.

A new option, \texttt{-tz} has been added into the \texttt{trepctl heartbeat} command to force the use of a
specific timezone.

For example, use GMT as the timezone when inserting a heartbeat:

\begin{verbatim}
shell> trepctl heartbeat -tz NONE
\end{verbatim}

Use the Replicator host's timezone to insert the heartbeat:

\begin{verbatim}
shell> trepctl heartbeat -tz HOST
\end{verbatim}

Use the given timezone to insert the heartbeat:

\begin{verbatim}
shell> trepctl heartbeat -tz \{valid timezone id\}
\end{verbatim}

If the MySQL server timezone is different from the host timezone (which is strongly not recom-
mended), then \texttt{-tz \{valid timezone id\}} should be used instead where \texttt{\{valid timezone id\}} should
be the same as the MySQL server timezone.

\textbf{Issues: CT-1066}

- Corrected resource leak when loading Java keystores

\textbf{Issues: CT-1091}

- Command-line Tools
  - Fixed error message to indicate the need to specify a service on Composite Master/Master clusters for the tung-
    sten\_find\_position and tungsten\_find\_seqno commands.

\textbf{Issues: CT-1098}

- The tpm command no longer reports warnings about existing system triggers with MySQL 8+

\textbf{Issues: CT-1099}

- Core Replicator
  - When configuring a Kafka Applier, the Kafka Port was set incorrectly

\textbf{Issues: CT-693}

- If a JSON field contained a single quote, the replicator would break during the apply stage whilst running the gener-
ated SQL into MySQL.

Single quotes will now be properly escaped to solve this issue

\textbf{Issues: CT-983}

- Under rare circumstances (network packet loss or MySQL Server hang), the replicator would also hang until restart-
ed.

This issue has been fixed by using specific network timeouts in both the replicator and in the Drizzle jdbc driver
connection logic

\textbf{Issues: CT-1034}

- When configuring MultiMaster, standalone replicators, with the BidiSlave filter enabled, the replicator was incorrect-
ly parsing certain DDL Statements and marking them as unsafe, as a result they were being dropped by the applier
and ignored

The full list of DDL commands fixed in this release are as follows:
• CREATE|DROP TRIGGER
• CREATE|DROP FUNCTION
• CREATE|DROP|ALTER|RENAME USER
• GRANT|REVOKE

Issues: CT-1084, CT-1117

The following warnings would appear in the replicator log due to GTID events not being handled.

WARN extractor.mysql.LogEvent Skipping unrecognized binlog event type 33, 34 or 35)

The WARN message will no longer appear, however GTID Events are still not handled in this release, but will be fully extracted in a future release.

Issues: CT-1114

A.3. Tungsten Clustering 6.1.1 GA [28 October 2019]

Version End of Life.  Not Yet Set

Release 6.1.1 contains both significant improvements as well as some needed bugfixes.

Improvements, new features and functionality

• Tungsten Manager
  • Improved how the Manager and Replicator behave when MySQL dies on the master node.

This improvement will induce a change of behavior in the product during failover by default, possibly causing a delay in failover as a way to protect data integrity.

The new default setting for 6.1.1 is:
replicator.store.thl.stopOnDBError=false

This means that the Manager will wait until the Replicator reads all remaining binlog events on the failing master node.

Failover will only continue once:
• all available events are completely read from the binary logs on the master node
• all events have reached the slaves

WARNING:
The new default means that the failover time could take longer than it used to.

When replicator.store.thl.stopOnDBError=true, then the Replicator will stop extracting once it is unable to update the trep_commit_seqno table in MySQL, and the Manager will perform the failover without waiting, at the risk of possible data loss due to leaving binlog events behind. All such situations are logged.

For use cases where failover speed is more important than data accuracy, those NOT willing to wait for long failover can set replicator.store.thl.stopOnDBError=false and still use tungsten_find_orphaned to manually analyze and perform the data recovery. For more information, please see Section 8.24, “The tungsten_find_orphaned Command”.

Issues: CT-583

• A new feature called “Cluster State Savepoints” has been implemented.

This new functionality was created to support clean, consistent rollbacks during aborted switch and failover operations. This functionality works for both physical clusters as well as for composite clusters.

To support this new feature, a new cluster sub-command has been added to the cctrl command - cluster topology validate, which will check and validate a cluster topology and, in the process, will report any issues that it finds. The purpose of this command is to provide a fast way to see, immediately, if there are any issues with any components of a cluster.

Savepoints are created automatically with every switch and failover command. The savepoint is only used if there is an exception during switch or failover that is actually able to be rolled-back.
WARNING:

Not all exceptions during switch and failover will cause a rollback.

In particular, if an exception happens during switch or failover AFTER a new primary datasource has been put online (relay or master) then the switch or failover operation cannot be rolled back.

The Manager is configured, by default, to hold a maximum of 50 savepoints. When that limit is hit, the Manager resets the current-save-point-id to 0 and starts to overwrite existing savepoints, starting at 0.

Issues: CT-951

For more information, see Section 8.1, “The cctrl Command”.

• Improved the ability of the manager to detect un-extracted, desirable binary log events when recovering the old master via cctrl after a failover.

The cctrl recover command will now fail if:

• any unextracted binlog events exist on the old master that we are trying to recover
• the old master THL contains more events than the slaves

In this case, the cctrl recover command will display text similar to the following:

Recovery failed because the failed master has unextracted events in the binlog. Please run the tungsten_find_orphaned script to inspect this events. Provided you have a recent backup available, you can try to restore the data source by issuing the following command:

datasource (hostname) restore

Please consult the user manual at:
https://docs.continuent.com/tungsten-clustering-6.1/operations-restore.html

The tungsten_find_orphaned script is designed to locate orphaned MySQL binary logs that were not extracted into THL before a failover. For more information, please see Section 8.24, “The tungsten_find_orphaned Command”.

Issues: CT-996

• Improved the ability to configure the manager's behavior upon failover.

During a failover, the manager will now wait until the selected slave has applied all stored THL events before promoting that node to master.

This wait time can be configured via the manager.failover.thl.apply.wait.timeout=0 property.

The default value is 0, which means "wait indefinitely until all stored THL events are applied".

Any value other than zero invites data loss due to the fact that once the slave is promoted to master, any unapplied stored events in the THL will be ignored, and therefore lost.

Whenever a failover occurs, the slave with most events stored in the local THL is selected so that when the events are eventually applied, the data is as close to the original master as possible with the least number of events missed.

That is usually, but not always, the most up-to-date slave, which is the one with the most events applied.

There should be a good balance between the value for manager.failover.thl.apply.wait.timeout and the value for policy.slave.promotion.latency.threshold=900, which is the number of seconds to which a slave must be current with the master in order to qualify as a candidate for failover. The default is 15 minutes (900 seconds).

Issues: CT-1022

Bug Fixes

• Command-line Tools

• Installing with disable-security-controls=false or when updating using: tools/tpm update --replace-jgroups-certificate --replace-tls-certificate would generate self-signed security certs that have a 1-year expiration which will cause installs to break eventually.

This expiration time value is controlled by the tpm command option --java-tls-key-lifetime, which is now set to 10 years or 3,650 days by default.
Release Notes

**Tungsten Manager**

- Fixed an issue where the `ls resources` command run inside of cctrl would fail to list the MANAGER entry on a slave node.
  - Issues: CT-599
- If the pipeline source replicator goes OFFLINE, the relay will reconnect to a different slave.
  - Issues: CT-871
- Fixed an issue where the Manager would show an exception when the MySQL check script did not get expected results.
  - Issues: CT-912
- Fixed use case where xtrabackup would timeout during backup via cctrl
  - Issues: CT-1045
- Improve ability to find needed binaries, both locally and over SSH, for commands: `tungsten_find_orphaned` and `tungsten_is_recoverable`
  - Issues: CT-1053

Tungsten Clustering 6.1.1 includes the following changes made in Tungsten Replicator 6.1.1

**Release 6.1.1 contains both significant improvements as well as some needed bugfixes.**

**Improvements, new features and functionality**

- **Core Replicator**
  - Added Clickhouse applier support.
    - Issues: CT-383
  - If using the dropcolumn filter during extraction, in conjunction with the Batch Applier [eg Replicating to Redshift, Hadoop, Vertica] writes would fail with a CSV mismatch error due to gaps in the THL Index.
    - However, for JDBC appliers, the gaps are required to ensure the correct column mapping
    To handle the two different requirements, a new property has been added to the filter to control whether or not to leave the THL index untouched [the default] or to re-order the Index ID's
    If applying to Batch targets, then the following property should be added to your configuration. The property is not required for JDBC targets:
    ```
    --property=replicator.filter.dropcolumn.fillGaps=true
    ```
    - Issues: CT-1025

**Bug Fixes**

- **Command-line Tools**
  - Fixed an issue that would prevent reading remote binary logs when using SSL.
    - Issues: CT-958
  - Fixed an issue where the command trepctl -all-services status -name watches fails.
    - Issues: CT-977
  - Restored previously-removed log file symbolic links under $CONTINUENT_ROOT/service_logs/
Release Notes

Issues: CT-1026

• Fixed a bug where `tpm diag` would generate an empty zip file if the hostnames contain hyphens `-` or periods `.`.

Issues: CT-1032

• Improve ability to find needed binaries for commands: `tungsten_find_position`, `tungsten_find_seqno` and `tungsten_get_rtt`.

Issues: CT-1054

A.4. Tungsten Clustering 6.1.0 GA [31 July 2019]

Version End of Life. Not Yet Set

Release 6.1.0 contains both significant improvements as well as some needed bugfixes. One of the main features of this release is MySQL 8 support.

The Tungsten Stack now supports the new MySQL 8.0 authentication plugins. Both `sha256_password` and `caching_sha2_password` (the new default) are supported by the Replicator, Manager and Connector.

More info on these authentication plugins can be found here: https://dev.mysql.com/doc/refman/8.0/en/sha256-pluggable-authentication.html

The Drizzle driver has been updated to support these new authentication methods, and the MySQL Connector/J 8 is also supported.

Behavior Changes

The following changes have been made to Tungsten Clustering 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 Connector `passThroughMode` configuration option is now deprecated.

The following `passThroughMode` entry will be removed from `tungsten-connector/conf/connector.properties`. There is currently no `tpm` option for this, and it is undocumented. The default will be kept to `passThroughMode=true`.

```
# The Tungsten Connector offers an extra fast data transfer mode known as
# pass-through. When the following switch enabled (default), the Connector
# will directly transfer data packets between the client and the server.
# When disabled, every native MySQL command will be translated into a JDBC call.
passThroughMode=true
```

Issues: CT-897

Known Issue

The following issues may affect the operation of Tungsten Clustering and should be taken into account when deploying or updating to this release.

• Some applications might fail to connect to the Connector with MariaDB 10+

When using MariaDB 10+, the Connector will be confused by the 10 and will think it is a MySQL 8+ server. By default, the Connector will offer to connect with `caching_sha2_password`. If the application does not know how to switch authentication plugins, it will fail with a message similar to the following:

```
The server requested authentication method unknown to the client [caching_sha2_password]
```

As a work-around, you may specify the authentication plugin using the following `tpm` command option:

```
--property=defaultAuthPlugin=mysql_native_password
```

Issues: CT-1033

Improvements, new features and functionality

• Command-line Tools

• A new utility script has been added to the release, `tungsten_post_process`, which assists with the graceful maintenance of the static cross-site replicator configuration files on disk.

Issues: CT-761
For more information, see Section 8.31, "The tungsten_post_process Command".

- **Tungsten Connector**

  - The Tungsten Stack now supports the new MySQL 8.0 authentication plugins. Both `sha256_password` and `caching_sha2_password` (the new default) are supported by the Replicator, Manager and Connector.


  The Drizzle driver has been updated to support these new authentication methods, and the MySQL Connector/J 8 is also supported.

  In order to be fully transparent with the new defaults, when connected to a MySQL 8+ data source, the Connector will advertise `caching_sha2_password` as the default plugin.

  With earlier versions of MySQL (pre-8.0), the previous default `mysql_native_password` is used by default and advertised to the client applications.

  In order to override the default behavior, a new Connector property option for `tpm`, `property=defaultAuthPlugin=[autodetect|caching_sha2_password|mysql_native_password]`, and is set to `autodetect` by default.

  Note that if `property=defaultAuthPlugin` is set to `caching_sha2_password`, the `sha256_password` authentication is automatically also supported.

  **Warning**

  Please note that the Connector does not support public key retrieval as of yet.

  Also note that, for backwards compatibility, the Connector forces the "CLIENT_DEPRECATE_EOF" to false, disallowing the usage of client session tracking requests [https://dev.mysql.com/doc/refman/5.7/en/session-state-tracking.html](https://dev.mysql.com/doc/refman/5.7/en/session-state-tracking.html)

  **Issues**: CT-771

  - When logging is set to `debug` or `trace`, the Connector will print individual queries. In the past, advanced logging limited the display size of requests to 256 characters to prevent overwhelming the logs in terms of both space and filesystem I/O.

    Some customers need to display more than that, so it is now possible to adjust the size of the statements displayed in `debug` or `trace` logging modes. This is handled by a new Connector property option for `tpm`, `property=statement.display.size.in.kb=NNN`, which is defined as the maximum query length to display in Kbytes, and now defaults to 1KB.

    **Warning**

    Warning: setting this option to a high value while DEBUG or TRACE is enabled will quickly fill logs and disk, in addition to using up disk I/O's!

    For example, if the raw query size is 4KB, then a setting of 1KB would simply display the first 1024 bytes of the query and truncate/discard the rest from a logging perspective.

    For more information about configuring debug and trace logging, please visit Section C.1.8, "Generating Advanced Diagnostic Information".

    **Issues**: CT-990

- **Bug Fixes**

  - **Tungsten Connector**

    - **OLD BEHAVIOR**: If the Master data source was not accessible when the Connector started (i.e. connection refused, etc.), the connector would still fully initialize, leading to a running Connector without an accessible data source. This has the side effects of having default configuration values for both `wait_timeout` and `server_version` instead of properly auto-detected values based on the MySQL server settings.

    **NEW BEHAVIOR**: The Connector will now wait indefinitely for a Master to become available before finishing startup.

    **Issues**: CT-930

    - Introduced a new `tpm` flag allowing for tuning Connector thread stack size, which can be required in particular cases where large requests are sent as text to a connector configured for automated read/write splitting (smartscale and direct reads).

    The setting is commented out by default, leaving the JVM use its own default; generally 1024.
Setting tpm option connector-thread-stack-size={value in kb} will override this value.

**Note**

Please note that since the new size will be allocated for each incoming connection, increasing the thread stack size will affect the total runtime memory used by the connector instance.

**Issues: CT-973**

Tungsten Clustering 6.1.0 includes the following changes made in Tungsten Replicator 6.1.0

Release 6.1.0 contains both significant improvements as well as some needed bugfixes. One of the main features of this release is MySQL 8 support.

**Improvements, new features and functionality**

- **Command-line Tools**
  - Two new utility scripts have been added to the release to help with setting the Replicator position:
    - `tungsten_find_position`, which assists with locating information in the THL based on the provided MySQL binary log event position and outputs a `dsctl set` command as output.
    - `tungsten_find_seqno`, which assists with locating information in the THL based on the provided sequence number and outputs a `dsctl set` command as output.

- **Issues: CT-934**
  - **Core Replicator**
    - A new, beta-quality command has been included called `prov-sl.sh` which is intended to eventually replace the current `tungsten_provision_slave` script.

Currently, `prov-sl.sh` supports provisioning slaves using `mysqldump` and `xtrabackup` tools, and is MySQL 8-compatible.

The `prov-sl.sh` command is written in Bash, has less dependencies compared to the current version and is meant to fix a number of issues with the current version.

Backups are streamed from source to target so that an intermediate write to disk is not performed, resulting in faster provisioning times.

Logs are written to `$CONTINUENT_ROOT/service_logs/prov-sl.log` (i.e. `/opt/continuent/service_logs/prov-sl.log`).

For example, provision a slave from [source db] using `mysqldump` (default):  

```
shell> prov-sl.sh -s {source db}
```

As another example, use `xtrabackup` for the backup method, with 10 parallel threads (default is 4), and ssh is listening on port 2222:

```
shell> prov-sl.sh -s {source db} -n xtrabackup -t 10 -p 2222
```

**Warning**

At the moment, `prov-sl.sh` does not support MultiMaster topologies when used with Tungsten Clustering, however it will be included in a future release.

**Issues: CT-614, CT-723, CT-809, CT-855, CT-963**

- **Upgraded the Drizzle driver to support MySQL 8 authentication protocols (SHA256, caching_sha2).**

  **Issues: CT-914, CT-931, CT-966**

- **The Redshift Applier now allows AWS authentication using IAM Roles. Previously authentication was possible via Access and Secret Key pairs only.**

  **Issues: CT-980**

For more information, see Redshift Preparation for Amazon Redshift Deployments [in Tungsten Replicator 6.1 Manual].
- Command-line Tools
  - When executing `mysqldump`, all Tungsten tools no longer use the `--add-drop-database` flag as it will prevent MySQL 8+ from restoring the dump.
    
    Issues: CT-935
  - Fixed a bug where `tpm diag` would generate an empty zip file if the hostnames contain hyphens `-` or periods `.`
    
    Issues: CT-1032

- Core Replicator
  - Added support for missing charset GB18030 to correct `WARN extractor.mysql.MysqlBinlog Unknown charset errors`.
    
    Issues: CT-915, CT-932
  - Loading data into Redshift would fail with the following error if a row of data contained a specific control character (0x00 [NULL])
    
    Missing newline: Unexpected character 0x30 found at location nnn
    
    Issues: CT-984
  - Now properly extracting the Geometry datatype.
    
    Issues: CT-997
  - The `ddl_map.json` file used by the `apply_schema_changes` filter was missing a rule to handle `ALTER TABLE` statements when replicating between MySQL and Redshift.
    
    Issues: CT-1002
  - The `extract_schema_change` filter wasn't escaping " [double-quotes] and the generated JSON would then cause the applier to error with
    
    pendingExceptionMessage: `SyntaxError: missing } after property list` 
    
    `(../../tungsten-replicator//support/filters-javascript/apply_schema_changes.js#236(eval)#1)`
    
    Issues: CT-1011
Appendix B. Prerequisites

Before you install Tungsten Clustering, there are a number of setup and prerequisite installation and configuration steps that must have taken place before any installation can continue. Section B.1, “Staging Host Configuration” and Section B.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 B.3, “MySQL Database Setup”, and will need to be performed on each appropriate host.

B.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 Tungsten Clustering™ software, and to install, transfer, and initiate the Tungsten Clustering™ service on each of the nodes within the cluster. The staging host can be a separate machine, or a machine that will be part of the cluster.

The recommended way to use Tungsten Clustering™ is to configure SSH on each machine within the cluster and allow the tpm tool to connect and perform the necessary installation and setup operations to create your cluster environment, as shown in Figure B.1, “Tungsten Deployment”.

Figure B.1. Tungsten Deployment

The staging host will be responsible for pushing and configuring each machine. For this to operate correctly, you should configure SSH on the staging server and each host within the cluster with a common SSH key. This will allow both the staging server, and each host within the cluster to communicate with each other.

You can use an existing login as the base for your staging operations. For the purposes of this guide, we will create a unique user, tungsten, from which the staging process will be executed.

1. Create a new Tungsten user that will be used to manage and install Tungsten Clustering™. The recommended choice for MySQL installations is to create a new user, tungsten. You will need to create this user on each host in the cluster. You can create the new user using adduser:

```
shell> sudo adduser tungsten
```

You can add the user to the mysql group adding the command-line option:

```
shell> sudo usermod -G mysql -a tungsten
```
2. Login as the **tungsten** user:

```
shell> su - tungsten
```

3. Create an SSH key file, but do not configure a password:

```
tungsten:shell> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/tungsten/.ssh/id_rsa):
Created directory '/home/tungsten/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/tungsten/.ssh/id_rsa.
Your public key has been saved in /home/tungsten/.ssh/id_rsa.pub.
The key fingerprint is:
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|             .   |
|            . .  |
|        S .. +   |
|       . o .X .  |
|        .oEO + . |
|       .o.=o. o  |
|      o=+..    . |
+-----------------+
```

This creates both a public and private key file; the public key file will be shared with the hosts in the cluster to allow hosts to connect to each other.

4. Within the staging server, profiles for the different cluster configurations are stored within a single directory. You can simplify the management of these different services by configuring a specific directory where these configurations will be stored. To set the directory, specify the directory within the `CONTINUENT_PROFILES` environment variable, adding this variable to your shell startup script (`.bashrc`, for example) within your staging server.

```
shell> mkdir -p /opt/continuent/software/conf
shell> mkdir -p /opt/continuent/software/replicator.conf
shell> export CONTINUENT_PROFILES=/opt/continuent/software/conf
shell> export REPLICATOR_PROFILES=/opt/continuent/software/replicator.conf
```

We now have a staging server setup, an SSH keypair for our login information, and are ready to start setting up each host within the cluster.

### B.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 authorization for the user on each host
- Configuring the directories and install locations
- Installing necessary software and tools
- Configuring `sudo` access to enable the configured user to perform administration commands

#### Important

The operations in the following sections must be performed on each host within your cluster. Failure to perform each step may prevent the installation and deployment of Tungsten cluster.

### B.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`:

```
shell> ulimit -a
core file size          (blocks, -c) 0
```

554
### Prerequisites

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>data seg size</td>
<td>(kbytes, -d)</td>
<td>unlimited</td>
</tr>
<tr>
<td>file size</td>
<td>(blocks, -f)</td>
<td>unlimited</td>
</tr>
<tr>
<td>max locked memory</td>
<td>(kbytes, -l)</td>
<td>unlimited</td>
</tr>
<tr>
<td>max memory size</td>
<td>(kbytes, -m)</td>
<td>unlimited</td>
</tr>
<tr>
<td>open files</td>
<td>(-o 256)</td>
<td></td>
</tr>
<tr>
<td>pipe size</td>
<td>(512 bytes, -p)</td>
<td>1</td>
</tr>
<tr>
<td>stack size</td>
<td>(kbytes, -s)</td>
<td>8192</td>
</tr>
<tr>
<td>cpu time</td>
<td>(seconds, -t)</td>
<td>unlimited</td>
</tr>
<tr>
<td>max user processes</td>
<td>(-u 709)</td>
<td></td>
</tr>
<tr>
<td>virtual memory</td>
<td>(kbytes, -v)</td>
<td>unlimited</td>
</tr>
</tbody>
</table>

The system should be configured to allow a minimum of 65535 open files. You should configure both the `tungsten` user and the database user with this limit by editing the `/etc/security/limits.conf` file:

```
tungsten -    nofile    65535
mysql       -    nofile    65535
```

In addition, the number of running processes supported should be increased to ensure that there are no restrictions on the running processes or threads:

```
tungsten -    nproc    8096
mysql       -    nproc    8096
```

You must logout and log back in again for the `ulimit` changes to take effect.

You may also need to set the limit settings on the specific service if your operating system uses the `systemctl` service management framework. To configure your file limits for the specific service:

1. Copy the MySQL service configuration file template to the configuration directory if it does not already exist:

   ```
   shell> sudo cp /lib/systemd/system/mysql.service /etc/systemd/system/
   ```

   **Note**
   Please note that the filename `mysql.service` will vary based on multiple factors. Do check to be sure you are using the correct file. For example, in some cases the filename would be `mysqld.service`

2. Edit the proper file used above, and append to or edit the existing entry to ensure the value of `infinity` for the key `LimitNOFILE`:

   ```
   LimitNOFILE=infinity
   ```

   This configures an unlimited number of open files, you can also specify a number, for example:

   ```
   LimitNOFILE=65535
   ```

3. Reload the `systemctl` daemon configuration:

   ```
   shell> sudo systemctl daemon-reload
   ```

4. Now restart the MySQL service:

   ```
   shell> service mysql restart
   ```

**Warning**

On Debian/Ubuntu hosts, limits are not inherited when using `su/sudo`. This may lead to problems when remotely starting or restarting services. To resolve this issue, uncomment the following line within `/etc/pam.d/su`:

```
session required pam_limits.so
```

### Integration with AppArmor

Make sure that AppArmor, if configured, has been enabled to support access to the `/tmp` directory for the MySQL processes. For example, add the following to the MySQL configuration file (usually `/etc/apparmor.d/local/usr.sbin.mysqld`):

```
/tmp/** rwk
```

### B.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.
Prerequisites

- Do not use Zeroconf [.local] addresses. These may not resolve properly or fully on some systems.
- The server hostname [as returned by the hostname] must match the names you use when configuring your service.
- The IP address that resolves on the hostname for that host must resolve to the IP address [not 127.0.0.1]. The default configuration for many Linux installations is for the hostname to resolve to the same as localhost:

  ```
  127.0.0.1 localhost
  127.0.0.1 host1
  ```

- Each host in the cluster must be able to resolve the address for all the other hosts in the cluster. To prevent errors within the DNS system causing timeouts or bad resolution, all hosts in the cluster, in addition to the witness host, should be added to `/etc/hosts`:

  ```
  127.0.0.1 localhost
  192.168.1.60 host1
  192.168.1.61 host2
  192.168.1.62 host3
  192.168.1.63 host4
  ```

  In addition to explicitly adding hostnames to `/etc/hosts`, the name server switch file, `/etc/nsswitch.conf` should be updated to ensure that hosts are searched first before using DNS services. For example:

  ```
  hosts:       files dns
  ```

  **Important**

  Failure to add explicit hosts and change this resolution order can lead to transient DNS resolving errors triggering timeouts and failsafe switching of hosts within the cluster.

- The IP address of each host within the cluster must resolve to the same IP address on each node. For example, if `host1` resolves to `192.168.0.69` on `host1`, the same IP address must be returned when looking up `host1` on the host `host2`.

  To double check this, you should perform the following tests:
  1. Confirm the hostname:

     ```
     shell> uname -n
     ```

     **Warning**

     The hostname cannot contain underscores.

  2. Confirm the IP address:

     ```
     shell> hostname --ip-address
     ```

  3. Confirm that the hostnames of the other hosts in the cluster resolve correctly to a valid IP address. You should confirm on each host that you can identify and connect to each other host in the planned cluster:

     ```
     shell> nslookup Host1
     shell> ping host1
     ```

     If the host does not resolve, either ensure that the hosts are added to the DNS service, or explicitly add the information to the `/etc/hosts` file.

     **Warning**

     If using `/etc/hosts` then you must ensure that the information is correct and consistent on each host, and double check using the above method that the IP address resolves correctly for every host in the cluster.

  **Witness Hosts**

  Tungsten Clustering 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 Tungsten Clustering 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.

**B.2.2.1. Network Ports**

The following network ports should be open between specific hosts to allow communication between the different components:
Prerequisites

<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>#</td>
<td>#</td>
<td>#</td>
<td>11999-12000</td>
<td>Tungsten manager</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>9997</td>
<td>Manager Remote Method Invocation (RMI)</td>
</tr>
<tr>
<td>#</td>
<td>2112</td>
<td>THL replication</td>
</tr>
<tr>
<td>#</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 Multi-site, Multi-master (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>#</td>
<td>2114</td>
<td>THL replication</td>
</tr>
<tr>
<td>#</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>

**Note**

For 6.0.0 and later, the THL ports configured depend on the first THL port, with one additional THL port used for each further service within the cluster. For example, in a two cluster system, port 2112 is used on the local clusters, and ports 2113, and 2114 would be used on the cross-site replicators.

If a system has a firewall enabled, in addition to enabling communication between hosts as in the table above, the localhost must allow port-to-port traffic on the loopback connection without restrictions. For example, using `iptables` this can be enabled using the following command rule:

```
shell> iptables -A INPUT -i lo -m state --state NEW -j ACCEPT
```

**B.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 `~/.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/
```
Prerequisites

With each host configured, you should try to connecting to each host from the staging server to confirm that the SSH information has been correctly configured. You can do this by connecting to the host using `ssh`:

```
tungsten:shell> ssh tungsten@host
```

You should have logged into the host at the `tungsten` home directory, and that directory should be writable by the `tungsten` user.

B.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. Without these checks, it is possible for the availability of hosts to be falsely determined. These checks are performed using one of two protocols:

- **ping** — the preferred method using the system `ping` (ICMP) command.
- **default** — no longer the default method even though it is labeled that way. Uses 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.

The configuration of which service to use depends on the setting of the `--mgr-ping-method [46]` option during configuration. If not option is given, `tpm` will test `ping` first and then try `default` after. An error will be thrown if neither option works for all members of the dataservice.

B.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 Tungsten Clustering™, as follows:

- **/tmp Directory**

  The `/tmp` directory must be accessible and executable, as it is the location where some software will be extracted and executed during installation and setup. The directory must be writable by the `tungsten` user.

  On some systems, the `/tmp` filesystem is mounted as a separate filesystem and explicitly configured to be non-executable (using the `noexec` filesystem option). Check the output from the `mount` command.

- **Installation Directory**

  Tungsten Clustering™ needs to be installed in a specific directory. The recommended solution is to use `/opt/continuent`. This information will be required when you configure the cluster service.

  The directory should be created, and the owner and permissions set for the configured user:

  ```
  shell> sudo mkdir /opt/continuent
  shell> sudo chown -R tungsten: /opt/continuent
  shell> sudo chmod 700 /opt/continuent
  ```

- **Home Directory**

  The home directory of the `tungsten` user must be writable by that user.

B.2.4. Configure Software

Tungsten Clustering™ relies on the following software. Each host must use the same version of each tool.

<table>
<thead>
<tr>
<th>Software</th>
<th>Versions Supported</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>perl</td>
<td>-</td>
<td>Check using <code>perl -v</code></td>
</tr>
<tr>
<td>rsync</td>
<td>-</td>
<td>Check using <code>rsync --help</code></td>
</tr>
<tr>
<td>Ruby</td>
<td>1.8.7, 1.9.3, or 2.0.0 to 2.4.0 a</td>
<td>JRuby is not supported</td>
</tr>
<tr>
<td>Ruby OpenSSL Module</td>
<td>-</td>
<td>Checking using <code>ruby -ropenssl -e 'p &quot;works&quot;'</code></td>
</tr>
<tr>
<td>Ruby Gems</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Ruby io-console module</td>
<td>-</td>
<td>Install using <code>gem install io-console</code> b</td>
</tr>
<tr>
<td>Ruby net-ssh module</td>
<td>-</td>
<td>Install using <code>gem install net-ssh</code> c</td>
</tr>
<tr>
<td>Ruby net-scp module</td>
<td>-</td>
<td>Install using <code>gem install net-scp</code> d</td>
</tr>
<tr>
<td>GNU tar</td>
<td>-</td>
<td><code>gtar</code> is required for Solaris due to limitations in the native <code>tar</code> command</td>
</tr>
<tr>
<td>Java Runtime Environment</td>
<td>Java SE 8 [or compatible], Java SE 11 [or compatible] is supported in 6.1.2 and higher</td>
<td></td>
</tr>
</tbody>
</table>
Prerequisites

<table>
<thead>
<tr>
<th>Software</th>
<th>Versions Supported</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>zip</td>
<td>-</td>
<td>zip is required by tpm diag in 6.1.2 and higher</td>
</tr>
<tr>
<td>readlink (GNU coreutils)</td>
<td>-</td>
<td>On most platforms, this should be available. readlink, supplied by GNU coreutils, is required by tpm diag</td>
</tr>
<tr>
<td>greadlink</td>
<td>-</td>
<td>Required to run tpm on MacOS/X Systems only</td>
</tr>
</tbody>
</table>

These tools must be installed, running, and available to all users on each host.

To check the current version for any installed tool, login as the configured user [e.g. tungsten], and execute the command to get the latest version. For example:

- **Java**
  
  Run `java -version`:
  ```
  shell>
  java -version
  openjdk version "1.8.0_102"
  OpenJDK Runtime Environment (build 1.8.0_102-b14)
  OpenJDK 64-Bit Server VM (build 25.102-b14, mixed mode)
  ```
  
  Tungsten Clustering is known to work with Java using the following JVMs:
  - Oracle JVM/JDK 8
  - OpenJDK 8
  
  On certain environments, a separate tool such as alternatives [RedHat/CentOS] or update-alternatives [Debian/Ubuntu] may need to be used to switch Java versions globally or for individual users. For example, within CentOS:
  ```
  shell>
  alternatives --display
  ```

  **Important**
  It is recommended to switch off all automated software and operating system update procedures. These can automatically install and restart different services which may be identified as failures by Tungsten Replicator. Software and Operating System updates should be handled by following the appropriate Section 5.15, “Performing Database or OS Maintenance” procedures.

  It is also recommended to install ntp or a similar time synchronization tool so that each host in the cluster has the same physical time.

B.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/xtrabackup, /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:
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
```

### B.2.6 SE Linux Configuration

**Important**

Tungsten best practice is to DISABLE SELinux.

If you are having problems, disable SELinux entirely and test fully. Once the system is fully operational and stable, then re-enable SELinux and see if the problem comes back. If so, then detailed configuration will be required. Such configuration is outside the scope of the Continuent Support Agreement.

To determine the current state of SE Linux enforcement, use the getenforce command. For example:

```
shell> getenforce
Disabled
```

To disable SELinux, use the setenforce command. For example:

```
shell> setenforce 0
```

**Warning**

Disclaimer: Continuent support staff will do our best to help but can’t guarantee results. SELinux context errors are extremely difficult to debug, and difficult to determine the appropriate command to give the correct context. A simple "cp" command could carry an incorrect context to the destination. Since every system is different, the possible variations in configuration are endless.

Any suggestions below are just that - suggestions, and your mileage may vary. You have been warned.

When SELinux is enabled, systemctl may refuse to start mysqld if the listener port or location on disk have been changed. The solution is to inform SELinux about any changed or additional resources.

Tungsten best practice is to change the default MySQL port from 3306 to 13306 so that requesting clients do not accidentally connect directly to the database without being routed by the Connector.

If using a non-standard port for MySQL and SELinux is enabled, you must also change the port context, for example:

```
shell> semanage port -a -t mysqld_port_t -p tcp 13306
```

Ensure the file contexts are set correctly for SELinux. For example, to allow MySQL data to be stored in a non-standard location (i.e. /data):

```
shell> semanage fcontext -a -t etc_runtime_t /data
shell> restorecon -Rv /data/
shell> semanage fcontext -a -t mysql_db_t "/data(/.*)?"
shell> restorecon -Rv /data/*
```

### B.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.

**Important**

For MySQL extraction, Tungsten Clustering must have write access to the database so that status and progress information can be recorded correctly.
Prerequisites

Note
Native MySQL replication should not be running when you install Tungsten Clustering™. The replication service will be completely handled by Tungsten Clustering™, and the normal replication, management and monitoring techniques will not provide you with the information you need.

B.3.1. MySQL Version Support

<table>
<thead>
<tr>
<th>Database</th>
<th>Version</th>
<th>Support Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>5.0, 5.1, 5.5, 5.6, 5.7</td>
<td>Primary platform</td>
<td>Statement and row based replication is supported. MyISAM and InnoDB table types are fully supported; MyISAM tables may introduce replication errors during failover scenarios. JSON datatype support only available in Tungsten Replicator v5.3 onwards</td>
</tr>
<tr>
<td>Percona</td>
<td>5.5, 5.6, 5.7</td>
<td>Primary platform</td>
<td>Statement and row based replication is supported. MyISAM and InnoDB table types are fully supported; MyISAM tables may introduce replication errors during failover scenarios. JSON datatype support only available in Tungsten Replicator v5.3 onwards</td>
</tr>
<tr>
<td>MariaDB</td>
<td>5.5, 10.0, 10.1</td>
<td>Primary platform</td>
<td></td>
</tr>
</tbody>
</table>

B.3.2. MySQL Configuration

Each MySQL Server should be configured identically within the system. Although binary logging must be enabled on each host, replication should not be configured, since Tungsten Replicator will be handling that process.

The configured tungsten user must be able to read the MySQL configuration file (for installation) and the binary logs. Either the tungsten user should be a member of the appropriate group (i.e. mysql), or the permissions altered accordingly.

Important
Parsing of mysqld_multi configuration files is not currently supported. To use a mysqld_multi installation, copy the relevant portion of the configuration file to a separate file to be used during installation.

To setup your MySQL servers, you need to do the following:

- Configure your my.cnf settings. The following changes should be made to the [mysqld] section of your my.cnf file:
  - By default, MySQL is configured only to listen on the localhost address (127.0.0.1). The bind-address parameter should be checked to ensure that it is either set to a valid value, or commented to allow listening on all available network interfaces:
    ```
    #bind-address = 127.0.0.1
    ```
  - Specify the server id
    Each server must have a unique server id:
    ```
    server-id = 1
    ```
    The best practice is for all servers to have a unique ID across all clusters. For example, use a numbering scheme like 0101, 0102, 0201, 0201, where the leading two digits are the cluster number and the last two digits are the node number, which allows for 99 participating clusters with 99 nodes each.
  - (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:
Prerequisites

- Set the `sync_binlog` parameter to 1 (one).

  **Note**

  In MySQL 5.7, the default value is 1.

  The MySQL `sync_binlog` parameter sets the frequency at which the binary log is flushed to disk. A value of zero indicates that the binary log should not be synchronized to disk, which implies that only standard operating system flushing of writes will occur. A value greater than one configures the binary log to be flushed only after `sync_binlog` events have been written. This can introduce a delay into writing information to the binary log, and therefore replication, but also opens the system to potential data loss if the binary log has not been flushed when a fatal system error occurs.

  Setting a value of one (one) will synchronize the binary log on disk after each event has been written.

- 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 increased to support this:

  ```
  max_allowed_packet = 52m
  ```

- Configure InnoDB as the default storage engine

  Tungsten Clustering needs to use a transaction safe storage engine to ensure the validity of the database. The InnoDB storage engine also provides automatic recovery in the event of a failure. Using MyISAM can lead to table corruption, and in the event of a switchover or failure, and inconsistent state of the database, making it difficult to recover or restart replication effectively.

  InnoDB should therefore be the default storage engine for all tables, and any existing tables should be converted to InnoDB before deploying Tungsten Clustering.

  ```
  default-storage-engine = InnoDB
  ```

- Configure InnoDB Settings

  Tungsten Replicator creates tables and must use InnoDB tables to store the status information for replication configuration and application.

  The MySQL option `innodb_flush_log_at_trx_commit` configures how InnoDB writes and confirms writes to disk during a transaction. The available values are:

  - A value of 0 (zero) provides the best performance, but it does so at the potential risk of losing information in the event of a system or hardware failure. For use with Tungsten Clustering™ the value should never be set to 0, otherwise the cluster health may be affected during a failure or failover scenario.
  - A value of 1 (one) provides the best transaction stability by ensuring that all writes to disk are flushed and committed before the transaction is returned as complete. Using this setting implies an increased disk load and so may impact the overall performance.
  - 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.

  When using Tungsten Clustering™ in a multi-master, multi-site, fan-in or data critical cluster, the value of `innodb_flush_log_at_trx_commit` should be set to 1. This not only ensures that the transactional data being stored in the cluster are safely written to disk, this setting also ensures that the metadata written by Tungsten Clustering™ describing the cluster and replication status is also written to disk and therefore available in the event of a failover or recovery situation.

  - A value of 2 (two) ensures that transactions are committed to disk, but data loss may occur if the disk data is not flushed from any OS or hardware-based buffering before a hardware failure, but the disk overhead is much lower and provides higher performance.

  This setting must be used as a minimum for all Tungsten Clustering™ installations, and should be the setting for all configurations that do not require `innodb_flush_log_at_trx_commit` set to 1.

  At a minimum `innodb_flush_log_at_trx_commit` should be set to 2; a warning will be generated if this value is set to zero:

  ```
  innodb_flush_log_at_trx_commit = 2
  ```

  MySQL configuration settings can be modified on a running cluster, providing you switch your host to maintenance mode before reconfiguring and restarting MySQL Server. See Section 5.15, “Performing Database or OS Maintenance”.

Optional configuration changes that can be made to your MySQL configuration:
• InnoDB Flush Method

- **innodb_flush_method=O_DIRECT**

The InnoDB flush method can effect the performance of writes within MySQL and the system as a whole.

O_DIRECT is generally recommended as it eliminates double-buffering of InnoDB writes through the OS page cache. Otherwise, MySQL will be contending with Tungsten and other processes for pages there — MySQL is quite active and has a lot of hot pages for indexes and the like that can result in 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 these 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 4.1, “Deploying Parallel Replication”.

• Increase InnoDB log file size

The default InnoDB Redo Log file size is 48MB. This should be increased to a larger file size for performance and other reasons. Values of 512MB are common.

To change the file size, read the corresponding information in the MySQL manual for configuring the file size information. Please see both “MySQL Redo Log” and “Optimizing MySQL InnoDB Redo Logging”.

• Binary Logging Format

Tungsten Replicator works with both statement and row-based logging, and therefore also mixed-based logging. The chosen format is entirely up to the systems and preferences, and there are no differences or changes required for Tungsten Replicator to operate. For native MySQL to MySQL master/slave replication, either format will work fine.

Depending on the exact use case and deployment, different binary log formats imply different requirements and settings. Certain deployment types and environments require different settings:

- For multi-master deployment, use row-based logging. This will help to avoid data drift where statements make fractional changes to the data in place of explicit updates.
- Use row-based logging for heterogeneous deployments. All deployments to Oracle, MongoDB, Vertica and others rely on row-based logging.
- Use mixed replication if warnings are raised within the MySQL log indicating that statement only is transferring possibly dangerous statements.
- 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:

\[
\text{binlog-format = row}
\]

**Note**

In MySQL 5.7, the default format is **ROW**.

For temporary changes during execution of explicit statements, the binlog format can be changed by executing the following statement:

\[
\text{mysql> SET binlog-format = ROW;}
\]

You must restart MySQL after any changes have been made.

- Ensure the **tungsten** user can access the MySQL binary logs by either opening up the directory permissions, or adding the **tungsten** user to the group owner for the directory.
B.3.3. MySQL Configuration for Multimaster Deployments

If you are inserting to the same table at the same time at two or more different sites, and using bi-directional or multimaster replication, then special care must be taken to avoid primary key conflicts. Either the auto-increment keys on each need to be offset so they do not conflict, or the application needs to be be able to generate unique keys taking multiple sites into account.

Important

The following configuration is required if your application is relying upon the MySQL-native auto-increment primary key feature:

Use the `auto_increment_increment` and `auto_increment_offset` variables to affect the way that MySQL generates the next value in an auto-increment field.

For example, edit `my.cnf` on all servers:

```plaintext
# for all servers at site 1
auto_increment_increment = 10
auto_increment_offset = 1

# for all servers at site 2
auto_increment_increment = 10
auto_increment_offset = 2

# for all servers at site 3
auto_increment_increment = 10
auto_increment_offset = 3

Important

Restart MySQL on all servers.
```

B.3.4. MySQL Configuration for Heterogeneous Deployments

The following are required for replication to heterogeneous targets to ensure that MySQL has been configured and generating row change information correctly:

- MySQL must be using Row-based replication for information to be replicated to heterogenous targets. For the best results, you should change the global binary log format, ideally in the configuration file (`my.cnf`):

  ```plaintext
  binlog-format = row
  ```

  Alternatively, the global binlog format can be changed by executing the following statement:

  ```sql
  mysql> SET GLOBAL binlog-format = ROW;
  ```

  For MySQL 5.6.2 and later, you must enable full row log images:

  ```plaintext
  binlog-row-image = full
  ```

  This information will be forgotten when the MySQL server is restarted; placing the configuration in the `my.cnf` file will ensure this option is permanently enabled.

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

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

  Tables must also be configured as UTF8 tables, and existing tables should be updated to UTF8 support before they are replicated to prevent character set corruption issues.

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

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

B.3.5. MySQL User Configuration

- Tungsten User Login
Prerequisites

The tungsten user connects to the MySQL database and applies the data from the replication stream from other datasources in the dataser-
vie. 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:

```sql
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:

```sql
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:

```sql
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 connec-
tions 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.

```sql
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 Tungsten Clustering™ configuration.

As noted above, the creation of explicit host-specific user entries may be required.

### B.4. Oracle Database Setup

**Important**

For Oracle extraction, Tungsten Clustering must have write access to the database so that status and progress inform-
ration can be recorded correctly.

#### B.4.1. Oracle Version Support

<table>
<thead>
<tr>
<th>Database</th>
<th>Version</th>
<th>Support Status</th>
<th>Notes</th>
</tr>
</thead>
</table>
| Oracle   | 10g Release 2 (10.2.0.5), 11g | Primary Platform                                         | Synchronous CDC is supported on Standard Edition only; Synchronous and Asyn-
chronous are supported on Enenterprise Editions |

#### B.4.2. Oracle Environment Variables

Ensure the `tungsten` User being used for the master Tungsten Replicator service has the same environment setup as an Oracle database user.
The user must have the following environment variables set:

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Sample Directory</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORACLE_HOME</td>
<td>/home/oracle/app/oracle/product/11.2.0/dbhome_2</td>
<td>The home directory of the Oracle installation.</td>
</tr>
<tr>
<td>LD_LIBRARY_PATH</td>
<td>$ORACLE_HOME/llib</td>
<td>The library directory of the Oracle installation.</td>
</tr>
<tr>
<td>ORACLE_SID</td>
<td>orcl</td>
<td>Oracle System ID for this installation.</td>
</tr>
</tbody>
</table>
Prerequisites

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Sample Directory</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAVA_HOME</td>
<td></td>
<td>The home of the Java installation.</td>
</tr>
<tr>
<td>PATH</td>
<td>$ORACLE_HOME/bin:$JAVA_HOME/bin</td>
<td>Must include the Oracle and Java binary directories.</td>
</tr>
<tr>
<td>CLASSPATH</td>
<td>$ORACLE_HOME/ucp/lib/ucp.jar:$ORACLE_HOME/jdbc/lib/ojdbc6.jar:$CLASSPATH</td>
<td>Must include the key Oracle libraries the Oracle JDBC driver.</td>
</tr>
</tbody>
</table>

These should be set within the `.bashrc` or `.profile` to ensure these values are set correctly for all logins.

B.5. Prerequisite Checklist

To simplify the process of preparing your hosts, the checklist below is designed to provide a quick summary of the main prerequisites required.

A PDF version of this checklist can also be downloaded [here](#).

**Host Specific**

<table>
<thead>
<tr>
<th>Pre-Req</th>
<th>Complete?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create OS User – Typically called <code>tungsten</code></td>
<td></td>
</tr>
<tr>
<td>Set <code>ulimit</code> for OS User</td>
<td></td>
</tr>
<tr>
<td>Configure sudoers</td>
<td></td>
</tr>
<tr>
<td>Disable SELinux</td>
<td></td>
</tr>
<tr>
<td>Compile <code>/etc/hosts</code></td>
<td></td>
</tr>
<tr>
<td>Setup SSH between hosts</td>
<td></td>
</tr>
<tr>
<td>Create directory for installation (Typically, <code>/opt/continuent</code>)</td>
<td></td>
</tr>
<tr>
<td>Create directory for software package if using tar bundle (Typically, <code>/opt/continuent/software</code>)</td>
<td></td>
</tr>
<tr>
<td>Create directory for configuration file if INI Install (<code>/etc/tungsten</code>)</td>
<td></td>
</tr>
<tr>
<td>Check ownership of new directories set to new OS user</td>
<td></td>
</tr>
<tr>
<td>Install Perl (if not already present)</td>
<td></td>
</tr>
<tr>
<td>Install Ruby</td>
<td></td>
</tr>
<tr>
<td>Install Ruby gems : <code>net-ssh</code></td>
<td></td>
</tr>
<tr>
<td>Install Ruby gems : <code>net-scp</code></td>
<td></td>
</tr>
<tr>
<td>Install Ruby gems : <code>io-console</code></td>
<td></td>
</tr>
<tr>
<td>install Java 8</td>
<td></td>
</tr>
<tr>
<td>install rsync</td>
<td></td>
</tr>
</tbody>
</table>

**Network Specific**

<table>
<thead>
<tr>
<th>Pre-Req</th>
<th>Complete?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensure Network ports open</td>
<td></td>
</tr>
</tbody>
</table>

**Database Specific [All Topologies]**

<table>
<thead>
<tr>
<th>Pre-Req</th>
<th>Complete?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensure <code>server-id</code> unique amongst all nodes</td>
<td></td>
</tr>
<tr>
<td>increase Open Files limits</td>
<td></td>
</tr>
<tr>
<td>Ensure <code>bin-logging</code> enabled for cluster nodes, or source replicator nodes</td>
<td></td>
</tr>
<tr>
<td>Review <code>sync_binlog</code> parameter</td>
<td></td>
</tr>
<tr>
<td>increase, if required, <code>max_allowed_packet</code></td>
<td></td>
</tr>
<tr>
<td>Review InnoDB settings</td>
<td></td>
</tr>
<tr>
<td>Set <code>binlog_format</code> to <code>ROW</code> (Essential for Multimaster or heterogeneous deployments)</td>
<td></td>
</tr>
<tr>
<td>Ensure <code>auto_increment</code> offsets adjusted for Multimaster deployments</td>
<td></td>
</tr>
</tbody>
</table>
### Prerequisites

<table>
<thead>
<tr>
<th>Pre-Req</th>
<th>Complete?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create DB user with FULL privileges and <code>GRANT OPTION</code> — typically called <code>tungsten</code> (Used by managers and repli-</td>
<td></td>
</tr>
<tr>
<td>cators)</td>
<td></td>
</tr>
</tbody>
</table>

**Database Specific for Clustering Deployments**

<table>
<thead>
<tr>
<th>Pre-Req</th>
<th>Complete?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change DB port from default (eg 13306)</td>
<td></td>
</tr>
<tr>
<td>Create user with FULL privileges except <code>SUPER</code>, for use by connectors — typically called <code>app_user</code></td>
<td></td>
</tr>
<tr>
<td>Ensure additional application DB user accounts have <code>REPLICATION CLIENT</code> privilege (Only if connectors configured</td>
<td></td>
</tr>
<tr>
<td>to use <code>SMARTSCALE</code>)</td>
<td></td>
</tr>
</tbody>
</table>
Appendix C. Troubleshooting

The following sections contain both general and specific help for identifying, troubleshooting and resolving problems. Key sections include:

- General notes on contacting and working with support and supplying information, see Section C.1, “Contacting Support”.
- Error/Cause/Solution guidance on specific issues and error messages, and how the reason can be identified and resolved, see Section C.2, “Error/Cause/Solution”.
- Additional troubleshooting for general systems and operational issues.

C.1. Contacting Support

The support portal may be accessed at https://continuent.zendesk.com.

Continuent offers paid support contracts for Continuent Tungsten and Tungsten Replicator. If you are interested in purchasing support, contact our sales team at sales@continuent.com.

C.1.1. Support Request Procedure

Please use the following procedure when requesting support so we can provide prompt service. If we are unable to understand the issue due to lack of required information, it will prevent us from providing a timely response.

1. Please provide a clear description of the problem
2. Which environment is having the issue? [Prod, QA, Dev, etc.]
3. What is the impact upon the affected environment?
4. Identify the problem host or hosts and the role [master, slave, etc]
5. Provide the steps you took to see the problem in your environment
6. Upload the resulting zip file from tpm diag, potentially run more than once on different hosts as needed. Alternatively, use the tungsten_send_diag command.
7. Provide steps already taken and commands already run to resolve the issue
9. Have you checked the Continuent documentation? https://docs.continuent.com
10. Have you checked our general knowledge base? For our Error/Cause/Solution guidance on specific issues and error messages, and how the reason can be identified and resolved, see Section C.2, “Error/Cause/Solution”.

C.1.2. Creating a Support Account

You can create a support account by logging into the support portal at https://continuent.zendesk.com. Please use your work email address so that we can recognize it and provide prompt service. If we are unable to recognize your company name it may delay our ability to provide a response.

Be sure to allow email from helpdesk@continuent.com and notifications-helpdesk@continuent.com. These addresses will be used for sending messages from Zendesk.

C.1.3. Generating Diagnostic Information

To aid in the diagnosis of issues, a copy of the logs and diagnostic information will help the support team to identify and trace the problem. There are two methods of providing this information:

- Using tpm diag

The tpm diag command will collect the logs and configuration information from the active installation and generate a Zip file with the diagnostic information for all hosts within it. The command should be executed from the staging directory. Use tpm query staging to determine this directory:

```shell
tpm query staging
```

```shell
tungsten@host1:/home/tungsten/tungsten-clustering-6.1.3-43
```

```shell:
toolspm diag
```

568
The process will create a file called `tungsten-diag-2014-03-20-10-21-29.zip`, with the corresponding date and time information replaced. This file should be included in the reported support issue as an attachment.

For a staging directory installation, `tpm diag` will collect together all of the information from each of the configured hosts in the cluster. For an INI file based installation, `tpm diag` will connect to all configured hosts if `ssh` is available. If a warning that `ssh` is not available is generated, `tpm diag` must be run individually on each host in the cluster.

- **Manually Collecting Logs**

  If `tpm diag` cannot be used, or fails to return all the information, the information can be collected manually:

  1. Run `tpm reverse` on all the hosts in the cluster:

     ```
     shell> tpm reverse
     ```

  2. Collect the logs from each host. Logs are available within the `service_logs` directory. This contains symbolic links to the actual log files. The original files can be included within a `tar` archive by using the `-h` option. For example:

     ```
     shell> cd /opt/continuent
     shell> tar zcfh host1-logs.tar.gz ./service_logs
     ```

     The `tpm reverse` and log archives can then be submitted as attachments with the support query.

C.1.4. Open a Support Ticket

Login to the support portal and click on ‘Submit a Request’ at the top of the screen. You can access this page directly at https://continuent.zendesk.com/requests/new.

C.1.5. Open a Support Ticket via Email

Send an email to helpdesk@continuent.com from the email address that you used to create your support account. You can include a description and attachments to help us diagnose the problem.

C.1.6. Getting Updates for all Company Support Tickets

If multiple people in your organization have created support tickets, it is possible to get updates on any support tickets they open. You should see your organization name along the top of the support portal. It will be listed after the Check Your Existing Requests tab.

To see all updates for your organization, click on the organization name and then click the Subscribe link.

If you do not see your organization name listed in the headers, open a support ticket asking us to create the organization and list the people that should be included.

C.1.7. Support Severity Level Definitions

Summary of the support severity levels with initial response targets:

- **Urgent**: initial response within an hour

  Represents a reproducible emergency condition (i.e. a condition that involves either data loss, data corruption, or lack of data availability) that makes the use or continued use of any one or more functions impossible. The condition requires an immediate solution. Continuent guarantees a maximum one (1) hour initial response time. Continuent will continue to work with Customer until Customer’s database is back in production. The full resolution and the full root cause analysis will be provided when available.

- **High**: initial response within four (4) hours

  Represents a reproducible, non-emergency condition (i.e. a condition that does not involve either data loss, data corruption or lack of database availability) that makes the use or continued use of any one or more functions difficult, and cannot be circumvented or avoided on a temporary basis by Customer. Continuent guarantees a maximum four (4) hours initial response time.

- **Normal**: initial response within one (1) business day

  Represents a reproducible, limited problem condition that may be circumvented or avoided on a temporary basis by Customer. Continuent guarantees a maximum one (1) business day initial response time.

- **Low**: no guaranteed initial response interval

  Represents minor problem conditions or documentation errors that are easily avoided or circumvented by Customer. Additional request for new feature suggestions, which are defined as new functionality in existing product, are also classified as low severity level. Continuent
Troubleshooting

does not guarantee any particular initial response time, or a commitment to fix in any particular time frame unless Customer engages Continuent for professional services work to create a fix or a new feature.

C.1.8. Generating Advanced Diagnostic Information
To aid in the diagnosis of difficult issues, below are tools and procedures to assist in the data collection.

Warning
ONLY excute the below commands and procedures when requested by Continuent support staff.

Manager •Memory Usage Script
We have provided a script to easily tell us how much memory a given manager is consuming.
Place the script on all of your manager hosts (i.e. into the tungsten OS user home directory).

Note
The script assumes that 'cctrl' is in the path. If not, then change the script to provide a full path for cctrl.
shell> su - tungsten
shell> vi tungsten_manager_memory
#!/bin/bash
memval=`echo gc | cctrl | grep used | tail -1 | awk -F: '{print $2}' | tr -d ' |'`
megabytes=`expr $memval / 1000000`
timestamp=`date +"%F %T" | tr '-' '/'`
echo "$timestamp | `hostname` | $megabytes MB"
shell> chmod 750 tungsten_manager_memory
shell> ./tungsten_manager_memory

This script is ideally run from cron and the output redirected to time-stamped log files for later correlation with manager issues.

Manager •Thread Dump Procedure
This procedure creates a Manager memory thread dump for detailed analysis.
Run this command on manager hosts when requested by Continuent support.
This will append the detailed thread dump information to the log file named tmsvc.log in the /opt/continuent/tungsten/tungsten-manager/log directory.
shell> su - tungsten
shell> manager dump
shell> tungsten_send_diag -f /opt/continuent/tungsten/tungsten-manager/log/tmsvc.log -c {case_number}

Manager •Heap Dump Procedure
This procedure creates a Manager memory heap dump for detailed analysis.
Run this command on manager hosts when requested by Continuent support.
This will create a file named {hostname}.hprof in the directory where you run it.
shell> su - tungsten
shell> jmap -dump:format=b,file=`hostname`.hprof `ps aux | grep JANINO | grep -v grep | awk '{print $2}'`
shell> tungsten_send_diag -f `hostname`.hprof -c {case_number}

•
Configuring
Connector Debug Logging
This procedure allows the Connector to be configured for debug logging.
Perform this procedure on Connector hosts when requested by Continuent support.

Warning
Enabling Connector debug logging will decrease performance dramatically. Disk writes will increase as will disk
space consumption. Do not use in production environments unless instructed to do so by Continuent support. In
any case, run in this mode for as short a period of time as possible - just long enough to gather the needed debug
information. After that is done, disable debug logging.
570


Troubleshooting

To enable debug logging, edit the Connector configuration file `tungsten-connector/conf/log4j.properties` and uncomment two lines. For example:

```
shell> su - tungsten
shell> vi /opt/continuent/tungsten/tungsten-connector/conf/log4j.properties
 Uncomment these two lines:
#log4j.logger.org.continuent.myosotis=debug, stdout
#log4j.additivity.org.continuent.myosotis=false
so they look like this:
log4j.logger.org.continuent.myosotis=debug, stdout
log4j.additivity.org.continuent.myosotis=false
shell> connector reconfigure
```

**Warning**

When disabling debug logging, DO NOT comment the lines out! Instead replace `debug` with `info`.

To disable debug logging, edit the Connector configuration file `tungsten-connector/conf/log4j.properties` and change the keyword `debug` to `info` on for the single key `log4j.logger.org.continuent.myosotis`.

For example, this is how it should look when the edit to disable is completed:

```
shell> vi /opt/continuent/tungsten/tungsten-connector/conf/log4j.properties
... log4j.logger.org.continuent.myosotis=info, stdout ...
shell> connector reconfigure
```

### Configuring Connector Verbose Debug Logging

This procedure allows the Connector to be configured for verbose debug logging (and we mean VERBOSE).

Perform this procedure on Connector hosts when requested by Continuent support.

**Warning**

Enabling Connector verbose debug logging will decrease performance dramatically. Disk writes will increase as will disk space consumption. Do not use in production environments unless instructed to do so by Continuent support. In any case, run in this mode for as short a period of time as possible - just long enough to gather the needed debug information. After that is done, disable debug logging.

To enable verbose debug logging (and we mean VERBOSE), edit the Connector log configuration file `tungsten-connector/conf/log4j.properties` and ADD two lines. For example:

```
shell> su - tungsten
shell> vi /opt/continuent/tungsten/tungsten-connector/conf/log4j.properties
log4j.logger.org.continuent.myosotis.protocol.mysql.MySQLProtocolHandler=trace, stdout
log4j.additivity.org.continuent.myosotis.protocol.mysql.MySQLProtocolHandler=false
shell> connector reconfigure
```

**Warning**

When disabling verbose debug logging, DO NOT comment the lines out! Instead replace `trace` with `info`.

To disable verbose debug logging, edit the Connector log configuration file `tungsten-connector/conf/log4j.properties` and change the keyword `trace` to `info` for the single key `log4j.logger.org.continuent.myosotis.protocol.mysql.MySQLProtocolHandler`.

For example, this is how it should look when the edit to disable is completed:

```
shell> vi /opt/continuent/tungsten/tungsten-connector/conf/log4j.properties
... log4j.logger.org.continuent.myosotis.protocol.mysql.MySQLProtocolHandler=info, stdout ...
shell> connector reconfigure
```

### Configuring Connector Debug Logging via the Manager

This procedure allows the Manager to be configured to enable Connector-specific debug logging in the manager logs.
Troubleshooting

Perform this procedure on Manager hosts when requested by Continuent support.

Warning

Enabling Manager-based Connector debug logging will decrease performance. Disk writes will increase as will disk space consumption. Do not use in production environments unless instructed to do so by Continuent support. In any case, run in this mode for as short a period of time as possible - just long enough to gather the needed debug information. After that is done, disable debug logging.

To enable Manager-based Connector debug logging, edit the Manager log configuration file `tungsten-manager/conf/log4j.properties` and ADD two lines. For example:

```shell
su - tungsten
vi /opt/continuent/tungsten/tungsten-manager/conf/log4j.properties
log4j.logger.com.continuent.tungsten.manager.router.gateway.RouterGatewayProxy=DEBUG, stdout
log4j.additivity.com.continuent.tungsten.manager.router.gateway.RouterGatewayProxy=false
manager restart
```

Warning

When disabling Manager-based Connector debug logging, DO NOT comment the lines out! Instead replace DEBUG with info.

To disable Manager-based Connector debug logging, edit the Manager log configuration file `tungsten-manager/conf/log4j.properties` and change the keyword DEBUG to info for the single key `log4j.logger.com.continuent.tungsten.manager.router.gateway.RouterGatewayProxy`.

For example, this is how it should look when the edit to disable is completed:

```shell
vi /opt/continuent/tungsten/tungsten-manager/conf/log4j.properties
...
log4j.logger.com.continuent.tungsten.manager.router.gateway.RouterGatewayProxy=info, stdout
...
manager restart
```

C.2. Error/Cause/Solution

C.2.1. Lots of entries added to replicator log

Last Updated: 2015-06-01

Condition or Error

The logging level used by Tungsten Clustering creates a lot of entries, including WARN, this generates a lot of information that is difficult to find the real errors and problems. How do i change the logging level?

Causes

- By default, Tungsten Clustering reports a lot of information and detail, including INFO and other levels of detail that may generate a lot of information. For example:

```
```

Rectifications
• The logging level used to report status and other information, and that is written into the log, can be changed to reduce or lower the reporting level. To do this:

1. Edit the ~tungsten_home/tungsten/tungsten-replicator/conf/log4j.properties
2. Find the following line:
   
   ```
   log4j.logger.com.continuent.tungsten.replicator.filter.ReplicateFilter=DEBUG, stdout
   ```
   
   This will change the logging level so that only entries at DEBUG and higher will be output.

### C.2.2. Error: could not settle on encryption_client algorithm

**Condition or Error**

The following error is reported when trying to connect:

```plaintext
Error: could not settle on encryption_client algorithm
```

**Causes**

• Can be due to missing an acceptable cipher on any one of the hosts.

**Rectifications**

• This is a list of acceptable ciphers:

```plaintext
aes128-cbc
des-cbc
blowfish-cbc
cast128-cbc
aes192-cbc
aes256-cbc
rijndael-cbc@lysator.liu.se
idea-cbc
none
arcfour128
arcfour256
```

These can be configured in `/etc/ssh/sshd_config` under Ciphers.

Try adding a supported cipher (e.g., `aes256-cbc`) to the end of the ciphers in your ssh server config file. Note that SSH and OpenSSL ciphers are mapped, for example like the following:

```java
// Maps the SSH name of a cipher to it's corresponding OpenSSL name
SSH_TO_OSSL = {
    "3des-cbc" => "des-ede3-cbc",
    "blowfish-cbc" => "bf-cbc",
    "aes128-cbc" => "aes-128-cbc",
    "aes192-cbc" => "aes-192-cbc",
    "aes256-cbc" => "aes-256-cbc",
    "rijndael-cbc@lysator.liu.se" => "rijndael-cbc",
    "idea-cbc" => "idea-cbc",
    "arcfour128" => "rc4",
    "arcfour256" => "rc4",
    "arcfour512" => "rc4",
    "none" => "none"
}
```

### C.2.3. The master replicator stopped with a JDBC error.

**Condition or Error**

The master replicator stopped with a JDBC error.

**Causes**

• The error log may show a more detailed failure with the JDBC error message:

```plaintext
INFO | jvm 1 | 2016-02-08 17:16:24 | 2016-02-08 17:16:24,627 [qktdb - pool-2-thread-1] ERROR management.tungsten.TungstenPlugin Unable to start replication service due to underlying error
INFO | jvm 1 | 2016/02/08 17:16:24 | java.lang.NumberFormatException: For input string: "0000002417562130"
INFO | jvm 1 | 2016/02/08 17:16:24 | at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
```
The underlying reason for the error is that MySQL has created a binlog over 2GB and the replicator could not process the event due to the limit of a Java integer.

**Rectifications**

- The solution for this error, if the log is a rotate event (use `mysqlbinlog`) is to reposition the replicator using `tungsten_set_position`.

### C.2.4. ERROR backup.BackupTask Backup operation failed: null

**Last Updated: 2019-01-11**

**Condition or Error**

A full Xtrabackup backup has failed, and left the datasource's replicator offline.

**Causes**

- A common cause of this failure is the existence of zero-length `store-*.properties` files in the `backups/{serviceName}/` directory.

**Rectifications**

- Simply remove any zero-byte `store-*.properties` files from the `backups/{serviceName}/` directory and retry the backup.

### C.2.5. 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 or Error**

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 ) CHARACTER_SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL;
  ```

  This will update the tables from the following structure:

  ```
  mysql> SHOW CREATE TABLE tungsten.heartbeat;
  ```
C.2.6. Unable to update the configuration of an installed directory

Last Updated: 2013-08-07

Condition or Error

Running an update or configuration with `tpm` returns the error 'Unable to update the configuration of an installed directory'

Causes

- Updates to the configuration of a running cluster must be performed from the staging directory where Tungsten Clustering was originally installed.

Rectifications

- Change to the staging directory and perform the necessary commands with `tpm`. To determine the staging directory, use:

  `shell> tpm query staging`

- Then change to the staging directory and perform the updates:

  `shell> ./tools/tpm configure ....`

More Information

Chapter 2, Deployment

C.2.7. [S1000] [unixODBC][MySQL][ODBC 5.3[w] Driver]SSL connection error: unknown error number [ISQL]ERROR: Could not SQLConnect

Last Updated: 2015-06-01

Condition or Error

We have a new server dedicated to Zabbix monitoring. Zabbix uses an ODBC connection for MySQL. When we try to connect to a Tungsten connector from the new server using ODBC we receive an error:

[S1000] [unixODBC][MySQL][ODBC 5.3[w] Driver]SSL connection error: unknown error number [ISQL]ERROR: Could not SQLConnect

Causes

- The underlying cause is related to an SSL or encryption error, either the certificate is wrong, or the ciphers being used are not supported.

Examine the `connector.log` on the Tungsten server we are connecting to returns an error with each attempt:

```
INFO | jvm 1 | 2016/05/20 13:07:17 | WARN [MySQLProtocolHandler] - [172.16.0.120:43571] Error during transfer of authentication packet: no cipher suites
```
Troubleshooting

Connecting from to the new server using the mysql client may work:

```
[root@zabbix etc#]# mysql -uzabbix -pZ@bbix487sql -hnas-db-ct01-a.safemls.net
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 40019
Server version: 5.6.20-68.0-log tungsten Percona Server (GPL), Release 68.0, Revision 656
```

Connecting directly MySQL database on port 13306 using the ODBC connection may also work:

```
[root@zabbix etc#]# isql -v ct01
+---------------------------------------+
| Connected! |
|           |
| sql-statement |
| help [tablename] |
| quit |
|      |
+---------------------------------------+
```

Rectifications

- **zabbix** is trying to connect to the connector with SSL encryption, but the SSL is not operating. The easiest way to bypass this is disable SSL connections for ODBC. Add the following entry in `odbc.ini` (under the section for the host you're testing):

```
useSSL = No
```

C.2.8. Connector shows errors with "java.net.SocketException: Broken pipe"

**Last Updated: 2013-11-01**

**Condition or Error**

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.

C.2.9. cctrl reports MANAGER(state=STOPPED)

**Last Updated: 2013-11-01**

**Condition or Error**

`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 if it is not running. You can start it by running:

```
shell> manager start
```

Or:

```
shell> /opt/continuent/tungsten/tungsten-manager/bin/manager start
```

**More Information**

Section 4.2.3, “Restarting the Manager Service”

C.2.10. Backup agent name not found: xtrabackup-full

**Last Updated: 2015-06-01**
Troubleshooting

Condition or Error
A backup was taken with xtrabackup-full from the master. Slave appears to not be configured for xtrabackup-full, which results in there being issues with the restore. How can we configure the slave to use xtrabackup-full for restore?

Causes

- The underlying cause and indication is that the xtrabackup has not been installed properly on the slave, or not installed at all, at the point when Tungsten Clustering was being installed. The following will be seen in the status output after a failed restore:

```plaintext
minimumStoredSeqNo : -1
offlineRequests : NONE
pendingError : Unable to spawn restore request
pendingErrorCode : NONE
pendingErrorEventId : NONE
pendingErrorSeqno : -1
pendingExceptionMessage: Backup agent name not found: xtrabackup-full
```

Probably the other hosts didn’t require this setting specifically because xtrabackup was installed and detected when Tungsten Clustering was installed on them.

Rectifications

- The steps you need are:

  1. Install xtrabackup if not already installed on the slave in question
  2. If using the INI configuration method, add the line below to /etc/tungsten/tungsten.ini:

        ```plain
        backup-method=xtrabackup-full
        ```

        And then run tpm update on that slave host to update the configuration.

        Or if using staging method, update the configuration using:

        ```plain
        shell> tpm update --backup-method=xtrabackup-full
        ```

C.2.11. Replicator runs out of memory

Last Updated: 2016-05-18

Condition or Error
The replicator runs out of memory, triggers a stack trace indicator a memory condition, or the replicator fails to extract the transaction information from the MySQL binary log.

Causes

- The replicator operates by extracting (or applying) an entire transaction. This means that when extracting data from the binary log, and writing that to THL, or extracting from the THL in preparation for applying to the target, the entire transaction, or an entire statement within a multi-statement transaction, must be held in memory.

  In the event of a very large transaction having to be extracted, this can cause a problem with the memory configuration. The actual configuration of how much memory is used is determined through a combination of the number of fragments, the size of the internal buffer used to store those fragments, and the overall fragment size.

Rectifications

- Although you can increase the overall memory allocated to the replicator, changing the internal sizes used can also improve the performance and ability to extract data.

  First, try reducing the size of the buffer [replicator.global.buffer.size] used to hold the transaction fragments. The default for this value is 10, but reducing this to 5 or less will ease the required memory:

  ```plain
  replicator.global.buffer.size=10
  ```

  Altering the size of each fragment can also help, as it reduces the memory required to hold the data before it is written to disk and sent out over the network to slave replicators. Reducing the fragment size will reduce the memory footprint. The size is controlled by the replicator.extractor.dbms.transaction_frag_size parameter:

  ```plain
  replicator.extractor.dbms.transaction_frag_size=1000000
  ```

  Note that if you change the fragment size, you may need to reset the service on the extractor so that the binary log is parsed again. You can reset the service by using the trepcmd reset command.
C.2.12. trepctl status hangs

Last Updated: 2013-11-01

Condition or Error

The `trepctl status` command hangs at the end of the output after a "cannot fork" error.

Causes

- This can be caused by THL corruption on the slave.
- This can be also be caused by an Out-Of-Memory condition in either the replicator or in the OS itself.

Rectifications

- You can recreate the THL files on the slave(s) ONLY. 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:

```bash
shell> replicator stop
cd /opt/continuent
mv thl thl.old;
mkdir thl
shell> replicator start
shell> trepctl status
```

To increase the Replicator memory, add or edit the following `tpm` configuration option, then run `tpm update`:

```bash
repl-java-mem-size=2048
```

C.2.13. Replicator fails to connect after updating password

Last Updated: 2013-11-01

Condition or Error

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 Tungsten Clustering 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 offline
```
C.2.14. Attempt to write new log record with equal or lower fragno: seqno=3 previous stored fragno=32767 attempted new fragno=-32768

Last Updated: 2016-05-18

Condition or Error

The number of fragments in a single transaction has been exceeded.

Causes

- The maximum number of fragments within a single transaction within the network protocol is limited to 32768. If there is a very large transaction that exceeds this number of fragments, the replicator can stop and be unable to continue. The total transaction size is a combination of the fragment size [default is 1,000,000 bytes, or 1MB], and this maximum number (approximately 32GB).

Rectifications

- It is not possible to change the number of fragments in a single transaction, but the size of each fragment can be increased to handle much larger single transactions. To change the fragment size, configure the `replicator.extractor.dbms.transaction_frag_size` parameter. For example, by doubling the size, a transaction of 64GB could be handled:

```
replicator.extractor.dbms.transaction_frag_size=2000000
```

If you change the fragment size in this way, the service on the extractor must be reset so that the transaction can be reprocessed and the binary log is parsed again. You can reset the service by using the `trepctl reset` command.

C.2.15. ORA-00257: ARCHIVER ERROR. CONNECT INTERNAL ONLY, UNTIL FREED

Last Updated: 2016-04-20

Condition or Error

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

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

Causes

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

Rectifications

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

```
shell> sqlplus sys/oracle as sysdba
SQL> ALTER SYSTEM SET db_recovery_file_dest_size = 80G;
```

C.2.16. ERROR 1010 (HY000) at line 5094506: Error dropping database [can't rmdir './mysql-bin/', errno: 17]

Last Updated: 2013-11-01

Condition or Error

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:
Troubleshooting

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

C.2.17. ERROR >> host1 >> can't alloc thread

Last Updated: 2019-03-04

Condition or Error
tpm installation fails with this error

Causes
- Most common occurrence of this error is often attributed to OS permissions.

Rectifications
- Review Section B.2.3, “Directory Locations and Configuration”
- Ensure all installation paths are owned by the correct OS user
- Ensure OS user configured with correct sudo rights

C.2.18. ERROR 1580 (HY000) at line 5093787: You cannot 'DROP' a log table if logging is enabled

Last Updated: 2013-11-01

Condition or Error
Loading a mysql dump 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
  ```

C.2.19. WARNING: An illegal reflective access operation has occurred

Last Updated: 2020-01-21

Condition or Error
The following Warning may be seen in the Manager Logs when running v6.1.2 and above in conjunction with Java 9 and above

```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.drools.core.rule.builder.dialect.asm.ClassGenerator
(file:/opt/continuent/releases/tungsten-clustering-6.1.2-75_pid46553/tungsten-manager/lib/drools-core-6.3.0.Final.jar)
to method java.lang.ClassLoader.defineClass(java.lang.String,byte[],int,int)
WARNING: Please consider reporting this to the maintainers of org.drools.core.rule.builder.dialect.asm.ClassGenerator
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
```

Causes
- This is a known issue due to new encapsulation controls in Java 9+

Rectifications
- This warning does not affect operations of the Cluster and can be safely ignored

C.2.20. ERROR 2013 (HY000) at line 583: Lost connection to MySQL server during query

Last Updated: 2013-11-01
Troubleshooting

**Condition or Error**

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.

**C.2.21. pendingExceptionMessage**: "Unable to update last commit seqno: Incorrect datetime value: '2016-03-13 02:02:26' for column 'update_timestamp' at row 1

**Last Updated: 2015-06-01**

**Condition or Error**

The following error is reported when applying an event:

**pendingExceptionMessage**: "Unable to update last commit seqno: Incorrect datetime value: '2016-03-13 02:02:26' for column 'update_timestamp' at row 1

**Causes**

- The underlying reason for this error is the format and value of the datetime value that is being represented are either incompatible with the current SQL mode within MySQL, or the datetime combination is one that occurs during a DST switch, which may be incompatible with the SQL mode.

**Rectifications**

- The solution is to update the SQL mode so that explicit changes are ignored when applying the data, rather than using the information defined during the session. To update the settings. Because the problem will be short lived and specific to the data being applied it can be done temporarily:
  - Edit file `/opt/continuent/tungsten/tungsten-replicator/conf/static-endtest.properties`
  - Find this line:
    ```
    replicator.applier.dbms.ignoreSessionVars=autocommit
    ```
  - Change the line to:
    ```
    replicator.applier.dbms.ignoreSessionVars=autocommit|sql_mode
    ```
  - Restart the replicator using:
    ```shell>
    replicator restart
    ```
  - Wait for the replicator to come online, and process the change that originally caused the problem. Once the data has been replicated, revert the settings in the file back to the old value and restart the replicator again.

**C.2.22. Too many open processes or files**

**Last Updated: 2013-10-09**

**Condition or Error**

The operating system or environment reports that the `tungsten` or designated Tungsten Clustering user has too many open files, processes, or both.

**Causes**

- User limits for processes or files have either been exhausted, or recommended limits for user configuration have not been set.
Troubleshooting

Rectifications

- Check the output of `ulimit` and check the configure file and process limits:

```shell
shell> ulimit -a
core file size (blocks, -c) 0
data seg size (kbytes, -d) unlimited
file size (blocks, -f) unlimited
max locked memory (kbytes, -l) unlimited
max memory size (kbytes, -m) unlimited
open files (-n) 256
pipe size (512 bytes, -p) 1
stack size (kbytes, -s) 8192
cpu time (seconds, -t) unlimited
max user processes (-u) 709
virtual memory (kbytes, -v) unlimited
```

If the figures reported are less than the recommended settings, see Section B.2.1, “Creating the User Environment” for guidance on how these values should be changed.

More Information

Section B.2.1, “Creating the User Environment”

C.2.23. WARN [KeepAliveTimerTask] - Error while sending a KEEP_ALIVE query to connection.

Last Updated: 2017-02-15

Condition or Error

Connections to MySQL through the connector report KeepAliveTimerTask errors in the `connector.log` file

Causes

- Possible causes include local scripts that kill stale connections after some fixed period of time, or the wait_timeout was changed without restarting the connector, or a bug that was fixed in v4.0.3

Rectifications

- Upgrade to the latest version if you are running a version below 4.0.3

  Check for local scripts that are killing connections

  Restart the Connector

```
shell> connector restart
```

C.2.24. Latency is high: master:ONLINE, progress=41331580333, THL latency=78849.733

Last Updated: 2015-06-01

Condition or Error

Latency is high: master:ONLINE, progress=41331580333, THL latency=78849.733

Causes

- There are many possible causes for this error, however, if you see the following within the log on the master it may indicate a specific issue:

```java
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitSeqnoTable.updateLastCommitSeqno(CommitSeqnoTable.java:548)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitManager.updateCommitSeqnoTable(CatalogManager.java:223)
```

The stack trace shows that the replicator is updating the `trep_commit_seqno` table which is normally a very fast operation.

The underlying reason may either be:
Troubleshooting

- It is possible that updates to MySQL are somehow getting delayed, which would slow down the operation of the replicator as it updates each status update.
- Check the block commit size, as low values will increase the number of updates to the table, and if the MySQL server updates are slow, this in turn slows down the operation of the replicator.

Rectifications

- Focus on making sure the IO system and MySQL commits are not being blocked.

You can try increasing `replicator.stage.q-to-thl.blockCommitRowCount` so the replicator has less commits to MySQL on the master.

You can continue to track progress through `trepctl status -name tasks` as you may see the `appliedLastSeqno` value updating less often if you increase this by a lot. Beware that increasing this value too much increases possible data loss since it creates less sync points with the slaves.

C.2.25. Triggers not firing correctly on slave

Last Updated: 2013-11-01

Condition or Error

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.pipeline.SingleThreadStageTask [q-to-dbms] 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 C.3.1, “Triggers”

C.2.26. cctrl hangs

Last Updated: 2013-11-01

Condition or Error

cctrl hangs

Causes
• Within Tungsten Clustering 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 4.2.3, “Restarting the Manager Service”

**C.2.27. Missing events, or events not extracted correctly**

*Last Updated: 2016-04-20*

**Condition or Error**

You have missing events, or events that have been correctly extraction from a Tungsten Clustering host.

**Causes**

• There are multiple potential causes for this issue, including, but not limited to:

**Rectifications**

• **Note**

  Before proceeding, ensure you have access to the failed master server and the new master server, if one was promoted. Ensure that your environment is properly initialized. For more information, see Initializing shell environment for VMware Continuent [2105429].

  The example operations below are for a service called `east`.

  **Note**

  To identify transactions that have not been extracted from a MySQL master running VMware Continuent replication:

  1. Identify the last transaction replicated to the slave servers.

     • If a new master has already been promoted or created:

        a. Run the following command:

        ```shell
        shell> trepctl -service east status
        ```

        Note the value of `latestEpochNumber`. Assume 5385 is the value of `latestEpochNumber`.

        b. Confirm this is the point where replication switched from the failed master to the new master.

        Now determine the THL and associated data from that THL sequence number using:

        ```shell
        shell> thl -service east list -headers -low 5385 -high 5385
        ```

        Now compare the last value on each line to ensure they are different. This value indicates the master host for the sequence number.

        If there are no differences, you need to refer to the trepsvc.log file of the new master to find the point where it became the master. Search for `[binlog-to-q-0] Setting extractor position from the end of the log`. This shows where the master started the extraction process from a new sequence number. Run the above check for each sequence number until you find the point where extraction switched from one master to another.

     • If no new master has been promoted but the master did fail:

        Run the following command to determine the current sequence number on the slaves:

        ```shell
        shell> dsctl -service east get
        ```

        If these values are different between multiple slaves, use the lowest reported sequence to reset the position. If the `dsctl` utility is not available, see the `trep_commit_seqno` table for information on the last replicated transaction for the slave. The location of this table is different depending on your slave DBMS.
Troubleshooting

Assume 5384 is the value of seqno returned by the command.

2. Check the output of the following command on the master and slaves:

```shell
thl -service east list -headers -seqno 5384
```

The output should be used to confirm that each host reports the same information. If there are differences, it is indicative of larger problems and you should reprovision from a known good datasource.

3. Run the following command to read the events from the master:

```shell
tungsten_read_master_events --service=east --after=5384
```

This will display the `mysqlbinlog` output for any transactions that were applied after the listed sequence number.

Use this information to determine the next course of action for your application. You may choose to ignore the transactions if they are not important or add them to the remaining servers manually and then repair replication.

More Information

Section 5.9.1, “Identifying a Transaction Mismatch”

C.2.28. Replication latency very high

_Last Updated: 2013-11-01_

**Condition or Error**

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
```

Slow slaves can be the cause, but it may require some configuration changes.

C.2.29. There were issues configuring the sandbox MySQL server

_Last Updated: 2016-04-20_

**Condition or Error**

- The command `tungsten_provision_thl` fails when using Percona Server.
- When running the command `tungsten_provision_thl`, you see the error:

```shell
There were issues configure the sandbox MySQL server
```

- MySQL Sandbox fails when using Percona Server.
- In the `$CONTINUENT_ROOT/service_logs/provision_thl.log` file, you see entries similar to:

```shell
mysqld. error while loading shared libraries: libssl.so.6: cannot open shared object file: No such file or directory
```

- In the `$CONTINUENT_ROOT/provision_thl.log` file, you see entries similar to:

```shell
mysql_install_db Error in my_thread_global_end(): 1 threads didn't exit
```

**Causes**

- This issue occurs because of a problem in Percona Server tarball distributions.

There are two issues with Percona Server tarball distributions, which depends on the version you have downloaded.
Troubleshooting

Look in the log file $CONTINUENT_ROOT/service_logs/provision_thl.log for:

- **mysqld**: error while loading shared libraries: libssl.so.6
- **mysql_install_db**: Error in my_thread_global_end()

Rectifications

- To resolve this issue in CentOS, install openssl by running the command:

  ```shell
sudo yum install openssl098e
  ````

  Alternatively, use Oracle MySQL or MariaDB which do not experience these issues.

  **Note**

  VMware does not endorse or recommend any particular third party utility.

More Information

Section 8.33, "The tungsten_provision_thl Command"

**C.2.30. Services requires a reset**

*Last Updated: 2016-05-18*

**Condition or Error**

The replicator service needs to be reset, for example if your MySQL service has been reconfigured, or when resetting a data warehouse or batch loading service after a significant change to the configuration.

**Causes**

- If the replicator stops replicating effectively, or the configuration and/or schema of a source or target in a data warehouse loading solution has changed significantly. This will reset the service, starting extraction from the current point, and the target/slave from the new master position. It will also reset all the positions for reading and writing.

**Rectifications**

- To reset a service entirely, without having to perform a re-installation, you should follow these steps. This will reset both the THL, source database binary log reading position and the target THL and starting point.

  1. Take the slave offline:
     ```shell
     slave-shell> trepctl offline
     ````

  2. Take the master offline:
     ```shell
     slave-shell> trepctl offline
     ````

  3. Use `trepctl` to reset the service on the master and slave. You must use the service name explicitly on the command-line:
     ```shell
     master-shell> trepctl -service alpha reset -y
     slave-shell> trepctl -service alpha reset -y
     ````

  4. Put the slave online:
     ```shell
     slave-shell> trepctl offline
     ````

  5. Put the master online:
     ```shell
     slave-shell> trepctl offline
     ````

**C.2.31. element 'mysql_readonly' not found in path**

*Last Updated: 2015-06-01*

**Condition or Error**

We are getting the following INFO message in the `tmsvc.log` every few seconds (host 1 is the Master), logs attached:

```
INFO | jvm 1 | 2016/03/04 15:49:38 | |host1 |
INFO | jvm 1 | 2016/03/04 15:49:38 | +----------------------------------------------------------------------------+
```


Troubleshooting

Causes

- This is caused by the manager

Rectifications

- To prevent INFO messages being reported in the /opt/continuent/tungsten/tungsten-manager/log/tmsvc.log file:
  1. Put the cluster into MAINTENANCE mode
  2. Stop all managers
  3. Start all managers starting with the master
  4. Put the cluster into AUTOMATIC mode

C.2.32. Starting replication after performing a restore because of an invalid restart sequence number

Last Updated: 2013-11-01

Condition or Error

Starting replication fails because of an invalid restart sequence number. Checking the sequence number, trep_commit_seqno shows an empty or invalid table contents:

```
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>
<tr>
<td>-------</td>
<td>--------</td>
<td>-----------</td>
<td>-----------</td>
<td>--------------</td>
<td>---------</td>
<td>-----------------</td>
<td>---------------------</td>
</tr>
</tbody>
</table>
0 rows 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. Ensure no replicator processes are running:

```
shell> replicator stop
```

Ensure the /opt/continuent/thl directory is empty:

```
shell> tpm reset-thl
```

2. Restore the backup using whatever backup/restore tool you used.

3. Check that trep_commit_seqno has a valid restart position in it

4. `shell> replicator start`

C.2.33. MySQL is incorrectly configured

Last Updated: 2013-11-01

Condition or Error

The configuration of MySQL was wrong; it included autocommit=0 and the wrong server-id

Causes

- Pre-requisites were not followed correctly.

Rectifications
Troubleshooting

- Edit `my.cnf` and clean up. Restart MySQL if possible. Alternatively, set manually:

```sql
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)
```

### C.2.34. Replicator reports an Out of Memory error

**Last Updated: 2013-11-01**

**Condition or Error**

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 `wrapper.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”

### C.2.35. Backup/Restore is not bringing my host back to normal

**Last Updated: 2013-11-01**

**Condition or Error**

A backup/restore was 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 5.11.3, “Restoring from Another Slave” for instructions on that.

### C.3. Known Issues

#### C.3.1. Triggers

Tungsten Replicator does not automatically shut off triggers on slaves. This can create problems on slaves as the trigger will run twice. Typical symptoms are duplicate key errors, though other problems may appear.

There is no simple one-answer-fits-all solution as the behaviour of MySQL and Triggers will differ based on various conditions.

- When using `ROW` Based Binary Logging, MySQL will log all data changes in the binary log, including any data changes performed as a result of a trigger firing

- When using `MIXED` Based Binary Logging...
  - if the Trigger is deemed to be non-deterministic then MySQL will behave based on the `ROW` Based Logging rules and log all data changes, including any data changes performed as a result of a trigger firing.
  - if the Trigger is deemed to be deterministic, then MySQL will behave based on `STATEMENT` Based Logging rules and **ONLY** log the statement issued by the client and **NOT** log any changes as a result of the trigger firing

The mixed behaviour outlined above presents challenges for Tungsten Replicator because MySQL does not flag transactions as being the result of a trigger firing or a client application. Therefore, it is not possible for the replicator to make a decision either.
This means, that if you are running with **MIXED** Based Binary Logging enabled, then there may be times when you would want the triggers on the target to fire, and times when you don't. Therefore the recommendations are as follows:

Tungsten Clustering Deployments

- Switch to **ROW** Based Binary Logging, and either
  - Implement the `is_master()` function outlined below, or
  - Use the `replicate.ignore` filter to ignore data changes to tables altered by Triggers [ONLY suitable if the filtered tables are solely managed by the Trigger]

Tungsten Replicator Deployments

- If source instance is running in **ROW** Based Binary Logging mode
  - Drop triggers on target. This is practical in fan-in topologies for reporting or other cases where you do not need to failover to the slave at a later time. Optionally also implement the `dropddl.js` JavaScript filter [Available in Tungsten Replicator v6.1.2 onwards] to prevent CREATE/DROP TRIGGER DDL being replicated, or
  - Implement the `is_master()` function outlined below, or
  - Use the `replicate.ignore` filter to ignore data changes to tables altered by Triggers [ONLY suitable if the filtered tables are solely managed by the Trigger], or
  - Switch to **ROW** Based Binary Logging and follow recommendations above

The `is_master()` approach is simple to implement. First, create a function like the following that returns false if we are using the Tungsten user, as would be the case on a slave.

```sql
create function is_master()
returns boolean
deterministic
return if(substring_index(user(),'@',1) != 'tungsten',true, false);
```

Next add this to triggers that should not run on the slave, as shown in the next example. This suppresses trigger action to insert into table bar except on the master.

```sql
delimiter //
create trigger foo_insert after insert on foo
for each row begin
  if is_master() then
    insert into bar set id=NEW.id;
  end if;
end;
//
```

As long as applications do not use the Tungsten account on the master, the preceding approach will be sufficient to suppress trigger operation.

Alternatively, if you are implementing the `is_master()` within a clustering deployment, you could check the database `read_only` parameter. In a clustered deployment, the slave databases will be in `read_only` mode and therefore the trigger could be coded to only fire when the database `read_only` mode is OFF

### C.4. Troubleshooting Timeouts

### C.5. Troubleshooting Backups

- Operating system command failed
  - Backup directory does not exist.
C.6. Running Out of Diskspace

The above indicates that the THL information could not be stored on disk. To recover from this error, make space available on the disk, or move the THL files to a different device with more space, then set the replicator service online again.

For more information on moving THL files to a different disk, see Section D.1.5.3, "Moving the THL File Location"; for information on moving the backup file location, see Section D.1.1.4, "Relocating Backup Storage".

C.7. Troubleshooting SSH and tpm

When executing `tpm`, ssh is used to connect and install the software on other hosts in the cluster. If this fails, and the public key information is correct, there are a number of operations and settings that can be checked. Ensure that you have followed the Section B.2.2.2, "SSH Configuration" instructions.

- The most likely representation of this error will be when executing `tpm` during a deployment:

```
Error:
******************************************************************************
Validation failed
******************************************************************************
Errors for host1
******************************************************************************
ERROR>>host1>>Unable to SSH to host1 as root. (SSHLoginCheck)
Ensure that the host is running and that you can login as root via SSH using key authentication
```

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 [ "-t" ]; 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.

C.8. Troubleshooting Data Differences

It can sometimes become necessary to identify table and data differences due to unexpected behaviour or failures. There are a number of third party tools that can help identify and fix however a lot of them assume native replication is in place, the following explains the recommended methods for troubleshooting a Tungsten Environment based on MySQL as the source and target technologies.

C.8.1. Identify Structural Differences

If you suspect that there are differences to a table structure, a simple method to resolve this will be to compare schema DDL.
Troubleshooting

Extract DDL on the Master node, specifying the schema in place of {DB}:

```
shell> mysqldump -u root -p --no-data --host=localhost --databases {DB} >master.sql
```

Repeat the same on the Slave node:

```
shell> mysqldump -u root -p --no-data --host=localhost --databases {DB} >slave.sql
```

Now, using diff, you can compare the results:

```
shell> diff master.sql slave.sql
```

Using the output of diff, you can then craft the necessary SQL statements to re-align your structure

### C.8.2. Identify Data Differences

It is possible to use pt-table-checksum from the Percona Toolkit to identify data differences, providing you use the syntax described below for bypassing the native replication checks. First of all, it is advisable to familiarise yourself with the product by reading through the providers own documentation here:


Once you are ready, ensure you install the latest version to the persona toolkit on all nodes, next execute the following on the Master node:

```
shell> pt-table-checksum --set-vars innodb_lock_wait_timeout=500 --recursion-method=none --ignore-databases=mysql --ignore-databases-regex=tungsten* --host=localhost,u=tungsten,p=secret
```

On first run, this will create a database called percona, and within that database a table called checksums. The process will gather checksum information on every table in every database excluding the mysql and tungsten related schemas. You can now execute the following SQL Statement on the slave to identify tables with data differences:

```
SELECT db, tbl, SUM(this_cnt) AS total_rows, COUNT(*) AS chunks
FROM percona.checksums
WHERE (master_cnt <> this_cnt
OR master_crc <> this_crc
OR ISNULL(master_crc) <> ISNULL(this_crc))
GROUP BY db, tbl;
```

This `SELECT` will return any tables that it detects are different, it won’t show you the differences, or indeed how many, this is just a basic check. To identify and fix the changes, you could use `pt-table-sync`, however this product would by default assume native replication and also try and fix the problems for you. In a tungsten environment this would not be recommended, however by using the `--print` switch you can gather the SQL needed to be executed to fix the mistakes. You should run this, and review the output to determine whether you want to manually patch the data together or consider using `tungsten_provision_slave` to retrovision a node in the case of large quantities of differences.

To use `pt-table-sync`, first identify the tables with differences on each slave, in this example, the `SELECT` statement above identified that there was a data difference on the departments table within the employees database on db2. Execute the `pt-table-sync` script on the master, passing in the database name, table name and the slave host that the difference exists on:

```
shell> pt-table-sync --databases employees --tables departments --print h=db1,u=tungsten,p=secret,P=13306 h=db2
```

The first `h=` option should be the Master, also the node you run the script from, the second `h=` option relates to the slave that the difference exists on. Executing the script will output SQL statements that can be used to patch the data, for example the above statement produces the following output:

```
UPDATE `employees`.`departments`
SET `dept_name`='Sales'
WHERE `dept_no`='d007'
LIMIT 1;
```

The `UPDATE` statements could now be issued directly on the slave to correct the problem.

**Warning**

Generally, changing data directly on a slave is not recommended, but every environment is different. Before making any changes like this always ensure you have a FULL backup, and it would be recommended to shun the slave node (if in a clustered environment) before making any changes so as not to cause any potential interruption to connected clients.
C.9. Comparing Table Data

The Percona Toolkit includes a tool called pt-table-checksum that enables you to compare databases on different databases using a checksum comparison. This can be executed by running the checksum generation process on the master:

```
shell> pt-table-checksum --set-vars innodb_lock_wait_timeout=500 \
    --recursion-method=none \
    --ignore-databases=mysql \
    --ignore-databases-regex=tungsten* \
    h=localhost,u=tungsten,p=secret
```

Using MySQL, the following statement must then be executed to check the checksums generated on the master:

```
mysql> SELECT db, tbl, SUM(this_cnt) AS total_rows, COUNT(*) AS chunks 
    -> FROM percona.checksums WHERE ( master_cnt <> this_cnt OR master_crc <> this_crc OR ISNULL(master_crc) <> ISNULL(this_crc)) GROUP BY db, tbl;
```

Any differences will be reported and will need to manually corrected.

C.10. Troubleshooting Memory Usage
Appendix D. Files, Directories, and Environment

D.1. The Tungsten Clustering Install Directory

Any Tungsten Clustering™ installation creates an installation directory that contains the software and the additional directories where active information, such as the transaction history log and backup data is stored. A sample of the directory is shown below, and a description of the individual directories is provided in Table D.1, “Continuent Tungsten Directory Structure”.

```
ls -al /opt/continuent
```

The directories shown in the table are relative to the installation directory, the recommended location is `/opt/continuent`. For example, the THL files would be located in `/opt/continuent/thl`.

Table D.1. Continuent Tungsten Directory Structure

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backups</td>
<td>Default directory for backup file storage</td>
</tr>
<tr>
<td>conf</td>
<td>Configuration directory with a copy of the current and past configurations</td>
</tr>
<tr>
<td>relay</td>
<td>Location for relay logs if relay logs have been enabled.</td>
</tr>
<tr>
<td>releases</td>
<td>Contains one or more active installations of the Continuent Tungsten software, referenced according to the version number and active process ID.</td>
</tr>
<tr>
<td>service-logs</td>
<td>Logging information for the active installation</td>
</tr>
<tr>
<td>share</td>
<td>Active installation information, including the active JAR for the MySQL connection</td>
</tr>
<tr>
<td>thl</td>
<td>The Transaction History Log files, stored in a directory named after each active service.</td>
</tr>
<tr>
<td>tungsten</td>
<td>Symbolic link to the currently active release in <code>releases</code>.</td>
</tr>
</tbody>
</table>

Some advice for the contents of specific directories within the main installation directory are described in the following sections.

D.1.1. The backups Directory

The `backups` directory is the default location for the data and metadata from any backup performed manually or automatically by Tungsten Clustering™. The backup data and metadata for each backup will be stored in this directory.

An example of the directory content is shown below:

```
ls -al /opt/continuent/backups/
```

The `storage.index` contains the backup file index information. The actual backup data is stored in the GZipped file. The properties of the backup file, including the tool used to create the backup, and the checksum information, are location in the corresponding `.properties` file. Note that each backup and property file is uniquely numbered so that you can identify and restore a specific backup.

Different backups scripts and methods may place their backup information in a separate subdirectory. For example, `xtrabackup` stores backup data into `/opt/continuent/backups/xtrabackup`.

D.1.1.1. Automatically Deleting Backup Files

The Tungsten Replicator will automatically remove old backup files. This is controlled by the `--repl-backup-retention` 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.

D.1.1.2. Manually Deleting Backup Files

If you no longer need one or more backup files, you can delete the files from the filesystem. You must delete both the SQL data, and the corresponding properties file. For example, from the following directory:

```
ls -al /opt/continuent/backups
```

To delete the backup files for index 4:

```
rm /opt/continuent/backups/alpha/store-0000000004*
```

See the information in Section D.1.1.3, "Copying Backup Files" about additional files related to a single backup. There may be additional files associated with the backup that you will need to manually remove.

**Warning**

Removing a backup should only be performed if you know that the backup is safe to be removed and will not be required. If the backup data is required, copy the backup files from the backup directory before deleting the files in the backup directory to make space.

D.1.1.3. Copying Backup Files

The files created during any backup can copied to another directory or system using any suitable means. Once the backup has been completed, the files will not be modified or updated and are therefore safe to be moved or actively copied to another location without fear of corruption of the backup information.

There are multiple files associated with each backup. The number of files will depend on the backup method that was used. All backups will use at least two files in the /opt/continuent/backups directory.

```
cd /opt/continuent/backups
'''

To copy full backups:

```
scp store-[0-9]* host3:$PWD
```

If the xtrabackup-full method was used, you must transfer the corresponding directory from /opt/continuent/backups/xtrabackup. In this example that would be:

```
cd /opt/continuent/backups/xtrabackup
rsync -aze ssh full_xtrabackup_2014-08-16_15-44_86 host3:$PWD
```

If the xtrabackup-incremental method was used, you must transfer multiple directories. In addition to the corresponding directory from /opt/continuent/backups/xtrabackup you must transfer all xtrabackup-incremental directories since the most recent xtrabackup-full backup and then transfer that xtrabackup-full directory. See the example below for further explanation:

```
ls -alt /opt/continuent/backups/xtrabackup/
```

---

594
In this example there are two instances of \textit{xtrabackup-full} backups and four \textit{xtrabackup-incremental} backups.

- To restore either of the \textit{xtrabackup-full} backups then they would be copied to the target host on their own.
- To restore \texttt{incr\_xtrabackup\_2014-10-21\_20-58\_97}, it must be copied along with \texttt{full\_xtrabackup\_2014-10-20\_20-57\_41}.
- To restore \texttt{incr\_xtrabackup\_2014-10-19\_20-57\_76}, it must be copied along with \texttt{incr\_xtrabackup\_2014-10-18\_20-55\_38} and \texttt{full\_xtrabackup\_2014-10-17\_20-55\_1}.

D.1.1.4. Relocating Backup Storage

If the filesystem on which the main installation directory is running out of space and you need to increase the space available for backup files without interrupting the service, you can use symbolic links to relocate the backup information.

\textbf{Note}

When using an NFS mount point when backing up with \textit{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 \textit{xtrabackup} command to fail. The following settings should be made on the directory:

- Ensure the \texttt{no\_root\_squash} option on the NFS export is not set.
- Change the group and owner of the mount point to the \texttt{tungsten} user and \texttt{mysql} group:

  \begin{verbatim}
  shell> chown tungsten /mnt/backups
  shell> chgrp mysql /mnt/backups
  \end{verbatim}

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:

  \begin{verbatim}
  shell> chmod 770 /mnt/backups
  \end{verbatim}

- Mount the directory:

  \begin{verbatim}
  shell> mount host1:/exports/backups /mnt/backups
  \end{verbatim}

The backup directory can be changed using two different methods:

- Section D.1.1.4.1, “Relocating Backup Storage using Symbolic Links”
- Section D.1.1.4.2, “Relocating Backup Storage using Configuration Changes”

D.1.1.4.1. Relocating Backup Storage using Symbolic Links

To relocate the backup directory using symbolic links:

1. Ensure that no active backup is taking place of the current host. Your service does not need to be offline to complete this operation.

2. Create a new directory, or attach a new filesystem and location on which the backups will be located. You can use a directory on another filesystem or connect to a SAN, NFS or other filesystem where the new directory will be located. For example:

   \begin{verbatim}
   shell> mkdir /mnt/backupdata/continuent
   \end{verbatim}

3. Optional

   Copy the existing backup directory to the new directory location. For example:

   \begin{verbatim}
   shell> rsync -r /opt/continuent/backups/* /mnt/backupdata/continuent/
   \end{verbatim}

4. Move the existing directory to a temporary location:

   \begin{verbatim}
   shell> mv /opt/continuent/backups /opt/continuent/old-backups
   \end{verbatim}

5. Create a symbolic link from the new directory to the original directory location:

   \begin{verbatim}
   shell> ln -s /mnt/backupdata/continuent /mnt/backupdata/continuent
   \end{verbatim}
The backup directory has now been moved. If you want to verify that the new backup directory is working, you can optionally run a backup and ensure that the backup process completes correctly.

### D.1.4.2. Relocating Backup Storage using Configuration Changes

To relocate the backup directory by reconfiguration:

1. Ensure that no active backup is taking place of the current host. Your service does not need to be offline to complete this operation.

2. Create a new directory, or attach a new filesystem and location on which the backups will be located. You can use a directory on another filesystem or connect to a SAN, NFS or other filesystem where the new directory will be located. For example:

   ```bash
   mkdir /mnt/backupdata/continuent
   ```

3. Optional

   Copy the existing backup directory to the new directory location. For example:

   ```bash
   rsync -r /opt/continuent/backups/* /mnt/backupdata/continuent/
   ```

4. Following the directions for `tpm update` to apply the `--backup-directory=/mnt/backupdata/continuent` setting.

The backup directory has now been moved. If you want to verify that the new backup directory is working, you can optionally run a backup and ensure that the backup process completes correctly.

### D.1.2. The `releases` Directory

The `releases` directory contains a copy of each installed release. As new versions are installed and updated [through `tpm update`], a new directory is created with the corresponding version of the software.

For example, a number of releases are listed below:

```bash
ll /opt/continuent/releases/
total 20
drwxr-xr-x  5 tungsten mysql 4896 May 23 16:19 ./
drwxr-xr-x  9 tungsten mysql 4896 May 23 16:19 ../
drwxr-xr-x 10 tungsten mysql 4896 May 23 16:19 tungsten-clustering-6.1.3-43_pid16184/
drwxr-xr-x 10 tungsten mysql 4896 May 23 16:19 tungsten-clustering-6.1.3-43_pid14577/
drwxr-xr-x 10 tungsten mysql 4896 May 23 16:19 tungsten-clustering-6.1.3-43_pid23747/
drwxr-xr-x 10 tungsten mysql 4896 May 23 16:19 tungsten-clustering-6.1.3-43_pid24978/
```

The latest release currently in use can be determined by checking the symbolic link, `tungsten` within the installation directory. For example:

```bash
ll /opt/continuent
total 40
drwxr-xr-x 9 tungsten mysql 4896 May 23 16:19 ./
drwxr-xr-x 3 root     root  4096 Apr 29 16:09 ../
drwxr-xr-x 2 tungsten mysql 4896 May 23 16:19 backups/
drwxr-xr-x 2 tungsten mysql 4896 May 23 16:19 conf/
drwxr-xr-x 3 tungsten mysql 4896 May 23 16:19 relay/
drwxr-xr-x 5 tungsten mysql 4896 May 23 16:19 releases/
drwxr-xr-x 2 tungsten mysql 4896 May 23 16:19 service_logs/
drwxr-xr-x 2 tungsten mysql 4896 May 23 16:18 share/
drwxr-xr-x 2 tungsten mysql 4896 May 23 16:19 thl/
lrwxrwxrwx 1 tungsten mysql   63 May 23 16:19 tungsten -> /opt/continuent/releases/tungsten-clustering-6.1.3-43_pid24978/
```

If multiple services are running on the host, search for `.pid` files within the installation directory to determine which release directories are currently in use by an active service:

```bash
find /opt/continuent -name "*.pid"
```

Directories within the `releases` directory that are no longer being used can be safely removed.

### D.1.3. The `service_logs` Directory

The `service_logs` directory contains links to the log files for the currently active release. The directory contains the following links:

- `connector.log` — a link to the Tungsten Connector log.
- `tmsvc.log` — a link to the Tungsten Clustering manager log.
- `trepsvc.log` — a link to the Tungsten Replicator log.
D.1.4. The \textit{share} Directory

The \textit{share} directory contains information that is shared among all installed releases and instances of Tungsten Clustering. Unlike other directories, the \textit{share} directory is not overwritten or replaced during installation or update using \texttt{tpm}. This means that the directory can be used to hold information, such as filter configurations, without the contents being removed when the installation is updated.

D.1.5. The \textit{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 \textit{thl} directory:

```
shell> ls -al /opt/continuent/thl/alpha/
total 2291984
```

```
drwxrwxr-x 2 tungsten tungsten 4096 Apr 16 13:44 .
drwxrwxr-x 3 tungsten tungsten 4096 Apr 15 15:53 ..
-rw-r--r-- 1 tungsten tungsten 0 Apr 15 15:53 disklog.lck
-rw-r--r-- 1 tungsten tungsten 100137585 Apr 15 18:13 thl.data.0000000001
-rw-r--r-- 1 tungsten tungsten 100134069 Apr 15 18:18 thl.data.0000000002
-rw-r--r-- 1 tungsten tungsten 100180770 Apr 15 18:31 thl.data.0000000003
-rw-r--r-- 1 tungsten tungsten 10043894 Apr 16 18:34 thl.data.0000000004
-rw-r--r-- 1 tungsten tungsten 100379260 Apr 16 18:35 thl.data.0000000005
-rw-r--r-- 1 tungsten tungsten 100859685 Apr 16 18:26 thl.data.0000000006
-rw-r--r-- 1 tungsten tungsten 100515215 Apr 16 18:28 thl.data.0000000007
-rw-r--r-- 1 tungsten tungsten 100180770 Apr 16 18:31 thl.data.0000000008
-rw-r--r-- 1 tungsten tungsten 100134069 Apr 16 18:18 thl.data.0000000009
-rw-r--r-- 1 tungsten tungsten 100137585 Apr 16 18:13 thl.data.0000000010
-rw-r--r-- 1 tungsten tungsten 100133258 Apr 16 12:24 thl.data.0000000011
-rw-r--r-- 1 tungsten tungsten 100293278 Apr 16 12:32 thl.data.0000000012
-rw-r--r-- 1 tungsten tungsten 100819317 Apr 16 12:34 thl.data.0000000013
-rw-r--r-- 1 tungsten tungsten 100250972 Apr 16 12:35 thl.data.0000000014
-rw-r--r-- 1 tungsten tungsten 100337285 Apr 16 12:37 thl.data.0000000015
-rw-r--r-- 1 tungsten tungsten 10055338 Apr 16 12:38 thl.data.0000000016
-rw-r--r-- 1 tungsten tungsten 1007833 Apr 16 12:40 thl.data.0000000017
-rw-r--r-- 1 tungsten tungsten 100198421 Apr 16 13:32 thl.data.0000000018
-rw-r--r-- 1 tungsten tungsten 100136955 Apr 16 13:34 thl.data.0000000019
-rw-r--r-- 1 tungsten tungsten 100490927 Apr 16 13:41 thl.data.0000000020
-rw-r--r-- 1 tungsten tungsten 10064146 Apr 16 13:44 thl.data.0000000021
-rw-r--r-- 1 tungsten tungsten 100225119 Apr 16 13:42 thl.data.0000000022
-rw-r--r-- 1 tungsten tungsten 100390819 Apr 16 13:43 thl.data.0000000023
-rw-r--r-- 1 tungsten tungsten 100418115 Apr 16 13:43 thl.data.0000000024
```

THL files are created on both the master and slaves within the cluster. THL data can be examined using the \texttt{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 \texttt{thl} command:

```
shell> thl index
```

```
LogIndexEntry thl.data.0000000001(0:106)
LogIndexEntry thl.data.0000000002(107:203)
LogIndexEntry thl.data.0000000003(204:367)
LogIndexEntry thl.data.0000000004(368:464)
LogIndexEntry thl.data.0000000005(465:561)
LogIndexEntry thl.data.0000000006(562:658)
LogIndexEntry thl.data.0000000007(659:755)
LogIndexEntry thl.data.0000000008(756:1251)
LogIndexEntry thl.data.0000000009(1252:1348)
LogIndexEntry thl.data.0000000010(1349:1997)
LogIndexEntry thl.data.0000000011(1998:2493)
LogIndexEntry thl.data.0000000012(2494:2590)
LogIndexEntry thl.data.0000000013(2591:2754)
LogIndexEntry thl.data.0000000014(2755:2851)
LogIndexEntry thl.data.0000000015(2852:2948)
LogIndexEntry thl.data.0000000016(2949:3045)
LogIndexEntry thl.data.0000000017(3046:3142)
LogIndexEntry thl.data.0000000018(3143:3239)
LogIndexEntry thl.data.0000000019(3240:3672)
```

The THL files are retained for seven days by default, although this parameter is configurable. Due to the nature and potential size required to store the information for the THL, you should monitor the disk space and usage.

The purge is continuous and is based on the date the log file was written. Each time the replicator finishes the current THL log file, it checks for files that have exceeded the defined retention configuration and spawns a job within the replicator to delete files older than the retention policy. Old files are only removed when the current THL log file rotates.
D.1.5.1. Purging THL Log Information on a Slave

**Warning**

Purging the THL on a slave node can potentially remove information that has not yet been applied to the database. Please check and ensure that the THL data that you are purging has been applied to the database before continuing.

The THL files can be explicitly purged to recover disk space, but you should ensure that the currently applied sequence no to the database is not purged, and that additional hosts are not reading the THL information.

To purge the logs on a SLAVE node:

1. Determine the highest sequence number from the THL that you want to delete. To purge the logs up until the latest sequence number, you can use `trepctl` to determine the highest applied sequence number:

   ```shell
   trepctl services
   Processing services command...
   NAME           VALUE
   ----           -----  
   appliedLastSeqno: 3672
   appliedLatency  : 331.0
   role           : slave
   serviceName    : alpha
   serviceType    : local
   started        : true
   state          : ONLINE
   Finished services command...
   ```

2. Shun the slave datasource and put the replicator into the offline state using `cctrl`:

   ```shell
cctrl
   [LOGICAL] /alpha > datasource host1 shun
   [LOGICAL] /alpha > replicator host1 offline
   ```

   **Important**

   NEVER Shun the Master datasource!

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 3672
   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
   [LOGICAL] /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 8.21.4, “thl purge Command”.

D.1.5.2. Purging THL Log Information on a Master

**Warning**

Purging the THL on a Master node can potentially remove information that has not yet been applied to the slave databases. Please check and ensure that the THL data that you are purging has been applied to the database on all slaves before continuing.

**Important**

If the situation allows, it may be better to switch the Master role to a current, up-to-date slave, then perform the steps to purge THL from a slave on the old master host using Section D.1.5.1, “Purging THL Log Information on a Slave”.

598
Warning

Follow the below steps with great caution! Failure to follow best practices will result in slaves unable to apply transactions, forcing a full re-provisioning. For those steps, please see Section 5.6.1.1, “Provision or Reprovision a Slave”.

The THL files can be explicitly purged to recover disk space, but you should ensure that the currently applied sequence no to the database is not purged, and that additional hosts are not reading the THL information.

To purge the logs on a MASTER node:

1. Determine the highest sequence number from the THL that you want to delete. To purge the logs up until the latest sequence number, you can use `trepctl` to determine the highest applied sequence number:

   ```shell
   shell> trepctl services
   Processing services command...
   NAME   VALUE  
   appliedLastSeqno: 3675
   appliedLatency  : 0.835
   role          : master
   serviceName   : alpha
   serviceType   : local
   started       : true
   state         : ONLINE
   Finished services command...
   ```

2. Set the cluster to Maintenance mode and put the replicator into the offline state using `cctrl`:

   ```shell
   shell> cctrl
   [LOGICAL] /alpha > set policy maintenance
   [LOGICAL] /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
   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. Set the cluster to Automatic mode and put the replicator online using `cctrl`:

   ```shell
   shell> cctrl
   [LOGICAL] /alpha > replicator host1 online
   [LOGICAL] /alpha > set policy automatic
   ```

You can now check the current THL file information:

   ```shell
   shell> thl index
   logindexentry thl.data.0000000024(3240:3672)
   ```

For more information on purging events using `thl`, see Section 8.21.4, “thl purge Command”.

D.1.5.3. Moving the THL File Location

The location of the THL directory where THL files are stored can be changed, either by using a symbolic link or by changing the configuration to point to the new directory:

- Changing the directory location using symbolic links can be used in an emergency if the space on a filesystem has been exhausted. See Section D.1.5.3.1, “Relocating THL Storage using Symbolic Links”
- Changing the directory location through reconfiguration can be used when a permanent change to the THL location is required. See Section D.1.5.3.2, “Relocating THL Storage using Configuration Changes”

D.1.5.3.1. Relocating THL Storage using Symbolic Links

In an emergency, the directory currently holding the THL information, can be moved using symbolic links to relocate the files to a location with more space.

Moving the THL location requires updating the location for a slave by temporarily setting the slave offline, updating the THL location, and re-enabling back into the cluster:

1. Shun the datasource and switch your node into the offline state using `cctrl`

599
2. Create a new directory, or attach a new filesystem and location on which the THL content will be located. You can use a directory on another filesystem or connect to a SAN, NFS or other filesystem where the new directory will be located. For example:

```
shell> mkdir /mnt/data/thl
```

3. Copy the existing THL directory to the new directory location. For example:

```
shell> rsync -r /opt/continuent/thl/* /mnt/data/thl/
```

4. Move the existing directory to a temporary location:

```
shell> mv /opt/continuent/thl /opt/continuent/old-thl
```

5. Create a symbolic link from the new directory to the original directory location:

```
shell> ln -s /mnt/data/thl /opt/continuent/thl
```

6. 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 'host1@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 'host2@alpha' TO POINT TO NEW MASTER 'host1@alpha'
SWITCH TO 'host1@alpha' WAS SUCCESSFUL
```

D.1.5.3.2. Relocating THL Storage using Configuration Changes

To permanently change the directory currently holding the THL information can be reconfigured to a new directory location.

To update the location for a slave by temporarily setting the slave offline, updating the THL location, and re-enabling back into the cluster:

1. 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:
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
   SELECTED SLAVE: host2@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 'host2@alpha' ONLINE
   PUT THE PRIOR MASTER 'host1@alpha' ONLINE AS A SLAVE
   RECONFIGURING SLAVE 'host2@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

D.1.5.4. Changing the THL Retention Times

THL files are by default retained for seven days, but the retention period can be adjusted according to requirements of the service. Longer times retain the logs for longer, increasing disk space usage while allowing access to the THL information for longer. Shorter logs reduce disk space usage while reducing the amount of log data available.

Note

The files are automatically managed by Tungsten Clustering. Old THL files are deleted only when new data is written to the current files. If there has been no THL activity, the log files remain until new THL information is written.

Use the `tpm update` command to apply the `--repl-thl-log-retention` setting. The replication service will be restarted on each host with updated retention configuration.

D.1.6. The `tungsten` Directory

```
shell> ls -l /opt/continuent/tungsten/
```

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bristlecone</td>
<td>Contains the bristlecone load-testing tools.</td>
</tr>
<tr>
<td>cluster-home</td>
<td>Home directory for the main tools, configuration and libraries of the Tungsten Clustering installation.</td>
</tr>
<tr>
<td>cookbook</td>
<td>Cookbook installation and testing tools.</td>
</tr>
<tr>
<td>INSTALL</td>
<td>Text file describing the basic installation process for Tungsten Clustering</td>
</tr>
<tr>
<td>README.LICENSES</td>
<td>Software license information.</td>
</tr>
</tbody>
</table>

Table D.2. Continuent Tungsten `tungsten` Sub-Directory Structure
Files, Directories, and Environment

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tools</td>
<td>Directory containing the tools for installing and configuring Tungsten Clustering.</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>

D.1.6.1. The **tungsten-connector** Directory

This directory holds all of the files, libraries, configuration and other information used to support the installation of Tungsten Connector.

D.1.6.1.1. The **tungsten-connector** Directory

This directory holds library files specific to Tungsten Connector. When perform patches or extending functionality specifically for Tungsten Connector, JAR files can be placed into this directory.

D.1.6.2. The **tungsten-manager** Directory

This directory holds all of the files, libraries, configuration and other information used to support the installation of Tungsten Manager.

D.1.6.2.1. The **tungsten-manager/lib** Directory

This directory holds library files specific to Tungsten Manager. When perform patches or extending functionality specifically for Tungsten Manager, JAR files can be placed into this directory.

D.1.6.3. The **tungsten-replicator** Directory

This directory holds all of the files, libraries, configuration and other information used to support the installation of Tungsten Manager.

D.1.6.3.1. The **tungsten-replicator/lib** Directory

This directory holds library files specific to Tungsten Replicator. When perform patches or extending functionality specifically for Tungsten Replicator, for example when adding JDBC libraries for other databases, the JAR files can be placed into this directory.

D.1.6.3.2. The **tungsten-replicator/scripts** Directory

This directory contains scripts used to support Tungsten Replicator operation.

D.2. Log Files

Each component within a cluster generates its own log files. These log files are all written to the directory structure within the installation directory itself. Additionally, symbolic links are generated for easier access to light weight logs more suited for general user use.

For example, this is the listing of the default log directory, `/opt/continuent/service_logs`:

```
connector-user.log -> /opt/continuent/tungsten/tungsten-connector/log/connector-user.log
manager-user.log -> /opt/continuent/tungsten/tungsten-manager/log/manager-user.log
mysqldump.log -> /opt/continuent/tungsten/tungsten-replicator/log/mysqldump.log
replicator-user.log -> /opt/continuent/tungsten/tungsten-replicator/log/replicator-user.log
xtrabackup.log -> /opt/continuent/tungsten/tungsten-replicator/log/xtrabackup.log
```

As you can see, each log file is a symlink to the user-level log for each of the layers, along with logs for backups, if they exist.

The more detailed log files can be found in each of the “real” log directories, for example:

```
/opt/continuent/tungsten/tungsten-manager/log/tmsvc.log
/opt/continuent/tungsten/tungsten-connector/log/connector.log
/opt/continuent/tungsten/tungsten-replicator/log/trepsvc.log
```

D.3. Environment Variables

- **$CONTINUENT_PROFILES**

  This environment variable is used by `tpm` as the location for storing the `deploy.cfg` file that is created by `tpm` during a `tpm configure` or `tpm install` operation. For more information, see Section 9.3, “tpm Staging Configuration”.

- **$REPLICATOR_PROFILES**
When using `tpm` with Tungsten Replicator, `$REPLICATOR_PROFILES` is used for storing the `deploy.cfg` file during configuration and installation. If `$REPLICATOR_PROFILES` does not exist, then `$CONTINUENT_PROFILES` if it exists. For more information, see Section 9.3, "tpm Staging Configuration".

- `$CONTINUENT_ROOT`

The `$CONTINUENT_ROOT` variable is created by the `env.sh` file that is created when installing Tungsten Clustering. When defined, the variable will contain the installation directory of the corresponding Tungsten Clustering installation.

On hosts where multiple installations have been created, the variable can be used to point to different installations.
Appendix E. Terminology Reference

Tungsten Clustering involves a number of different terminology that helps define different parts of the product, and specific areas of the output information from different commands. Some of this information is shared across different tools and systems.

This appendix includes a reference to the most common terms and terminology used across Tungsten Clustering.

E.1. Transaction History Log (THL)

The Transaction History Log (THL) stores transactional data from different data servers in a universal format that is then used to exchange and transfer the information between replicator instances. Because the THL is stored and independently managed from the data servers that it reads and writes, the data can be moved, exchanged, and transmuted during processing.

The THL is created by any replicator service acting as a master, where the information is read from the database using the native format, such as the MySQL binary log, or Oracle Change Data Capture (CDC), writing the information to the THL. Once in the THL, the THL data can be exchanged with other processes, including transmission over the network, and then applied to a destination database. Within Tungsten Replicator, this process is handled through the pipeline stages that read and write information between the THL and internal queues.

Information stored in THL is recorded in a series of event records in sequential format. The THL therefore acts as a queue of the transactions. On a replicator reading data from a database, the THL represents the queue of transactions applied on the source database. On a replicator applying that information to a database, the THL represents the list of the transactions to be written. The THL has the following properties:

- THL is a sequential list of events
- THL events are written to a THL file through a single thread (to enforce the sequential nature)
- THL events can be read from individually or sequentially, and multiple threads can read the same THL at the same time
- THL events are immutable; once stored, the contents of the THL are never modified or individually deleted (although entire files may be deleted)
- THL is written to disk without any buffering to prevent software failure causing a problem; the operating system buffers are used.

THL data is stored on disk within the `thl` directory of your Tungsten Replicator installation. The exact location can configured using `logDir` parameter of the THL component. A sample directory is shown below:

```
total 710504
-rw-r--r-- 1 tungsten tungsten         0 May  2 10:48 disklog.lck
-rw-r--r-- 1 tungsten tungsten 100842900 Jun  4 10:10 thl.data.0000000000000013
-rw-rw-r-- 1 tungsten tungsten 100825311 Jun  4 11:41 thl.data.0000000000000014
-rw-rw-r-- 1 tungsten tungsten 100441599 Jun  4 11:43 thl.data.0000000000000015
-rw-rw-r-- 1 tungsten tungsten 100898492 Jun  4 11:44 thl.data.0000000000000016
-rw-rw-r-- 1 tungsten tungsten 100355613 Jun  4 11:44 thl.data.0000000000000017
-rw-rw-r-- 1 tungsten tungsten 100835516 Jun  4 11:44 thl.data.0000000000000018
-rw-rw-r-- 1 tungsten tungsten 101698649 Jun  4 11:45 thl.data.0000000000000019
-rw-rw-r-- 1 tungsten tungsten 23886641 Jun  5 21:55 thl.data.0000000000000020
```

The THL files have the format `thl.data.#########`, and the sequence number increases for each new log file. The size of each log file is controlled by the `--thl-log-file-size` configuration parameter. The log files are automatically managed by Tungsten Replicator, with old files automatically removed according to the retention policy set by the `--thl-log-retention` configuration parameter. The files can be manually purged or moved. See Section D.1.5.1, “Purging THL Log Information on a Slave”.

The THL can be viewed and managed by using the `thl` command. For more information, see Section 8.21, “The thl Command”.

E.1.1. THL Format

The THL is stored on disk in a specific format that combines the information about the SQL and row data, metadata about the environment in which the row changes and SQL changes were made [metadata], and the log specific information, including the source, database, and timestamp of the information.

A sample of the output is shown below, the information is taken from the output of the `thl` command:

```
SEQ# = 0 / FRAG# = 0 (last frag)
-TIME = 2013-03-21 18:47:39.8
-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
```

In versions up to 5.3.2:
The sample above shows the information for the SQL executed on a MySQL server. The `EVENT_ID` [606] shows the MySQL binary log from which the statement has been read. The MySQL server has stored the information in the binary log using `STATEMENT` or `MIXED` mode; log events written in `ROW` mode store the individual row differences. A summary of the THL stored format information, including both hidden values and the information included in the `thl` command output is provided in Table E.1, "THL Event Format".

**Table E.1. THL Event Format**

<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>-</td>
<td>record_length</td>
<td>Integer</td>
<td>4 bytes</td>
<td>Length of the full record information, including this field</td>
</tr>
<tr>
<td>-</td>
<td>record_type</td>
<td>Byte</td>
<td>1 byte</td>
<td>Event record type identifier</td>
</tr>
<tr>
<td>-</td>
<td>header_length</td>
<td>Unsigned int</td>
<td>4 bytes</td>
<td>Length of the header information</td>
</tr>
<tr>
<td>SEQ# [605]</td>
<td>seqno</td>
<td>Unsigned long</td>
<td>8 bytes</td>
<td>Log sequence number, a sequential value given to each log entry</td>
</tr>
<tr>
<td>FRAG# [605]</td>
<td>fragno</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>
</tr>
<tr>
<td>-</td>
<td>last_frag</td>
<td>Byte</td>
<td>1 byte</td>
<td>Indicates whether the fragment is the last fragment in the sequence</td>
</tr>
<tr>
<td>EPOCH# [606]</td>
<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>
</tr>
<tr>
<td>SOURCEID [606]</td>
<td>source_id</td>
<td>UTF-8 String</td>
<td>Variable</td>
<td>Event source ID, the hostname or identity of the dataserver that generated the event</td>
</tr>
<tr>
<td>EVENTID [606]</td>
<td>event_id</td>
<td>UTF-8 String</td>
<td>Variable</td>
<td>Event ID; in MySQL, for example, the binlog filename and position that contained the original event</td>
</tr>
<tr>
<td>SHARDID [607]</td>
<td>shard_id</td>
<td>UTF-8 String</td>
<td>Variable</td>
<td>Shard ID to which the event belongs</td>
</tr>
<tr>
<td>TIME [606]</td>
<td>tstamp</td>
<td>Unsigned long</td>
<td>8 bytes</td>
<td>Time of the commit that triggered the event</td>
</tr>
<tr>
<td>FILE</td>
<td>-</td>
<td>String</td>
<td>-</td>
<td>Filename of the THL file containing the event</td>
</tr>
<tr>
<td>-</td>
<td>data_length</td>
<td>Unsigned int</td>
<td>4 bytes</td>
<td>Length of the included event data</td>
</tr>
<tr>
<td>-</td>
<td>event</td>
<td>Binary</td>
<td>Variable</td>
<td>Serialized Java object containing the SQL or ROW data</td>
</tr>
<tr>
<td>METADATA [606]</td>
<td>Part of event</td>
<td>-</td>
<td>-</td>
<td>Metadata about the event</td>
</tr>
<tr>
<td>TYPE [606]</td>
<td>Part of event</td>
<td>-</td>
<td>-</td>
<td>Internal storage type of the event</td>
</tr>
<tr>
<td>OPTIONS [606]</td>
<td>Part of event</td>
<td>-</td>
<td>-</td>
<td>Options about the event operation</td>
</tr>
<tr>
<td>SCHEMA [607]</td>
<td>Part of event</td>
<td>-</td>
<td>-</td>
<td>Schema used in the event</td>
</tr>
<tr>
<td>SQL [607]</td>
<td>Part of event</td>
<td>-</td>
<td>-</td>
<td>SQL statement or row data</td>
</tr>
<tr>
<td>-</td>
<td>crc_method</td>
<td>Byte</td>
<td>1 byte</td>
<td>Method used to compute the CRC for the event</td>
</tr>
<tr>
<td>-</td>
<td>crc</td>
<td>Unsigned int</td>
<td>4 bytes</td>
<td>CRC of the event record (not including the CRC value)</td>
</tr>
</tbody>
</table>

- **SEQ# [605] and FRAG# [605]**

Individual events within the log are identified by a sequential `SEQUENCE` [605] 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** [606]

The `EPOCH` [606] value is used as a check to ensure that the logs on the slave and the master match. The `EPOCH` [606] is stored in the THL, and a new `EPOCH` [606] is generated each time a master goes online. The `EPOCH` [606] value is then written and stored in the THL alongside each individual event. The `EPOCH` [606] acts as an additional check, beyond the sequence number, to validate the information between the slave and the master. The `EPOCH` [606] 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` [606] acts as a check so that when the master rejoins as the slave, the `EPOCH` [606] 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` [606] prevents the slave from accepting events that happen to match only the sequence number, but not the corresponding `EPOCH` [606].

Each time a Tungsten Replicator master goes online, the `EPOCH` [606] number is incremented. When the slave connects, it requests the `SEQUENCE` [605] and `EPOCH` [606], and the master confirms that the requested `SEQUENCE` [605] has the requested `EPOCH` [606]. 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 5.9, “Managing Transaction Failures”.

• **SOURCEID** [606]

The `SOURCEID` [606] is a string identifying the source of the event stored in the THL. Typically it is the hostname or host identifier.

• **EVENTID** [606]

The `EVENTID` [606] is a string identifying the source of the event information in the log. Within a MySQL installation, the `EVENTID` [606] 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 `EVENTID` 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** [606]

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` [606] value as stored in the THL is used to compute latency information when reading and applying data on a slave.

• **METADATA** [606]

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** [606]

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** [606]
Part of the **EVENT** binary payload, the **OPTIONS**[606] 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**[606] block for each SQL[607] statement stored in the event.

- **SCHEMA**[607]**

  Part of the **EVENT** structure, the **SCHEMA**[607] provides the database or schema name in which the statement or row data was applied.

- **SHARDID**[607]**

  When using parallel apply, provides the generated shard ID for the event when it is applied by the parallel applier thread.

- **SQL**[607]**

  For statement based events, the SQL of the statement that was recorded. Multiple individual SQL statements as part of a transaction can be contained within a single event fragment.

  For example, the MySQL statement:

  ```
  mysql> INSERT INTO user VALUES (null, 'Charles', now());
  Query OK, 1 row affected (0.01 sec)
  ```

  Stores the following into the THL:

  ```
  SEQ# = 3583 / FRAG# = 0 (last frag)
  - TIME = 2013-05-27 11:49:45.0
  - EPOCH = 2080
  - EVENTID = mysql-bin.000007:0000000625753960:0
  - SOURCEID = host1
  - METADATA = [mysql_server_id=1687011;dbms_type=mysql;service=firstrep;shard=test]
  - TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
  - SQL(0) = SET INSERT_ID = 3
  - OPTIONS =
    - charset = ISO8859_1
    - autocommit = 1
    - sql_auto_is_null = 0
    - foreign_key_checks = 1
    - unique_checks = 1
    - sql_mode = ''
    - character_set_client = 8
    - collation_connection = 8
    - collation_server = 8
  - SCHEMA = test
  - SQL(1) = INSERT INTO user VALUES (null, 'Charles', now()) /* ___SERVICE___ = [firstrep] */
  ```

  For row based events, the information is further defined by the individual row data, including the action type (UPDATE, INSERT or DELETE), **SCHEMA**[607], **TABLE**[607] and individual ROW data. For each ROW, there may be one or more **COL**[607](column) and identifying **KEY**[607] event to identify the row on which the action is to be performed.

  The same statement when recorded in **ROW**[607] format:

  ```
  SEQ# = 3582 / FRAG# = 0 (last frag)
  - TIME = 2013-05-27 11:45:19.0
  - EPOCH = 2080
  - EVENTID = mysql-bin.000007:0000000625753710:0
  - SOURCEID = host1
  - METADATA = [mysql_server_id=1687011;dbms_type=mysql;service=firstrep;shard=test]
  - TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
  - SQL(0) =
    - ACTION = INSERT
    - SCHEMA = test
    - TABLE = user
    - ROW# = 0
    - COL(1: ) = 2
    - COL(2: ) = Charles
    - COL(3: ) = 2013-05-27 11:45:19.0
  ```

**E.2. Generated Field Reference**

When using any of the tools within Tungsten Clustering status information is output using a common set of fields that describe different status information. These field names and terms are constant throughout all of the different tools. A description of each of these different fields is provided below.

**E.2.1. Terminology: Fields** **accessFailures**

**E.2.2. Terminology: Fields** **active**
E.2.3. Terminology: Fields  

activeSeqno

E.2.4. Terminology: Fields  

appliedLastEventId

The event ID from the source database of the last corresponding event from the stage that has been applied to the database.

MySQL

When extracting from MySQL, the output from `trepctl` shows the MySQL binary log file and the byte position within the log where the transaction was extracted:

```
shell> trepctl status
Processing status command...
NAME VALUE
    appliedLastEventId : mysql-bin.000064:0000000002757461;0
```

Oracle CDC

When extracting from Oracle using the CDC method, the event ID is composed of the Oracle SCN number:

```
NAME VALUE
    appliedLastEventId : ora:16626156
```

Oracle Redo Reader

When extracting from Oracle using the Redo Reader method, the event ID is composed of a combination of Oracle SCN, transaction, and PLOG file numbers, separated by a hash symbol:

```
NAME VALUE
    appliedLastEventId : 8931871791244#0018.002.000196e1#LAST#8931871791237#100644
```

The format is:

```
COMMITSCN#XID#LCR#MINSCN#PLOGSEQ
```

- COMMITSCN
  Last committed Oracle System Change Number (SCN).
- XID
  Transaction ID.
- LCR
  Last committed record number.
- MINSCN
  Minimum stored Oracle SCN.
- PLOGSEQ
  PLOG file sequence number.

E.2.5. Terminology: Fields  

appliedLastSeqno

The last sequence number for the transaction from the Tungsten stage that has been applied to the database. This indicates the last actual transaction information written into the slave database.

```
appliedLastSeqno : 212
```

When using parallel replication, this parameter returns the minimum applied sequence number among all the channels applying data.

E.2.6. Terminology: Fields  

appliedLatency

The `appliedLatency` is the latency between the commit time of the source event and the time the last committed transaction reached the end of the corresponding pipeline within the replicator.
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.

The latency is measured in seconds. Increasing latency may indicate that the destination database is unable to keep up with the transactions from the master.

In replicators that are operating with parallel apply, appliedLatency indicates the latency of the trailing channel. Because the parallel apply mechanism does not update all channels simultaneously, the figure shown may trail significantly from the actual latency.

E.2.7. Terminology: Fields **applier.class**

classname of the current applier engine

E.2.8. Terminology: Fields **applier.name**

Name of the current applier engine

E.2.9. Terminology: Fields **applyTime**

E.2.10. Terminology: Fields **autoRecoveryEnabled**

Indicates whether autorecovery has been enabled by setting the `--auto-recovery-max-attempts` field. The field indicates the value as either `true` or `false` accordingly.

E.2.11. Terminology: Fields **autoRecoveryTotal**

A count of the number of times the replicator has used autorecovery to go back online since the replicator was started. This can be used to determine if autorecovery has been used. More details on autorecovery can be found in the `trepsvc.log` file.

The counter is reset when the replicator determines that the replicator has successfully gone online after an autorecovery.

E.2.12. Terminology: Fields **averageBlockSize**

E.2.13. Terminology: Fields **blockCommitRowCount**


E.2.15. Terminology: Fields **channel**

E.2.16. Terminology: Fields **channels**

The number of channels being used to apply transactions to the target database. 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.

E.2.17. Terminology: Fields **clusterName**

The name of the cluster. This information is different to the service name and is used to identify the cluster, rather than the individual service information being output.

E.2.18. Terminology: Fields **commits**

E.2.19. Terminology: Fields **committedMinSeqno**
E.2.20. Terminology: Fields *criticalPartition*

E.2.21. Terminology: Fields *currentBlockSize*

E.2.22. Terminology: Fields *currentEventId*

   Event ID of the transaction currently being processed

E.2.23. Terminology: Fields *currentLastEventId*

E.2.24. Terminology: Fields *currentLastFragno*

E.2.25. Terminology: Fields *currentLastSeqno*

E.2.26. Terminology: Fields *currentTimeMillis*

   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, the information is used to record the time when transactions are applied, and may therefore be the *appliedAtency* figure.

E.2.27. Terminology: Fields *dataServerHost*

E.2.28. Terminology: Fields *discardCount*

E.2.29. Terminology: Fields *doChecksum*

E.2.30. Terminology: Fields *estimatedOfflineInterval*

E.2.31. Terminology: Fields *eventCount*

E.2.32. Terminology: Fields *extensions*

E.2.33. Terminology: Fields *extractTime*

E.2.34. Terminology: Fields *extractor.class*

E.2.35. Terminology: Fields *extractor.name*

E.2.36. Terminology: Fields *filter.#.class*

E.2.37. Terminology: Fields *filter.#.name*

E.2.38. Terminology: Fields *filterTime*
E.2.39. Terminology: Fields \textit{flushIntervalMillis}

E.2.40. Terminology: Fields \textit{fsyncOnFlush}

E.2.41. Terminology: Fields \textit{headSeqno}

E.2.42. Terminology: Fields \textit{intervalGuard}

E.2.43. Terminology: Fields \textit{lastCommittedBlockSize}

The \textit{lastCommittedBlockSize} contains the size of the last block that was committed as part of the block commit procedure. The value is only displayed on appliers and defines the number of events in the last block. By comparing this value to the configured block commit size, the commit type can be determined.

For more information, see Section 11.1, "Block Commit".

E.2.44. Terminology: Fields \textit{lastCommittedBlockTime}

The \textit{lastCommittedBlockSize} contains the duration since the last committed block. The value is only displayed on appliers and defines the number of seconds since the last block was committed. By comparing this value to the configured block interval, the commit type can be determined.

For more information, see Section 11.1, "Block Commit".

E.2.45. Terminology: Fields \textit{latestEpochNumber}

E.2.46. Terminology: Fields \textit{logConnectionTimeout}

E.2.47. Terminology: Fields \textit{logDir}

E.2.48. Terminology: Fields \textit{logFileRetainMillis}

E.2.49. Terminology: Fields \textit{logFileSize}

E.2.50. Terminology: Fields \textit{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:

\begin{verbatim}
masterConnectUri : thl://host1:2112/
\end{verbatim}

In a secure installation where SSL is being used to exchange data, the URI protocol will be \textit{thls}:

\begin{verbatim}
masterConnectUri : thls://host1:2112/
\end{verbatim}

E.2.51. Terminology: Fields \textit{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.

\begin{verbatim}
masterListenUri : thl://host1:2112/
\end{verbatim}

E.2.52. Terminology: Fields \textit{maxChannel}
E.2.53. Terminology: Fields maxDelayInterval

E.2.54. Terminology: Fields maxOfflineInterval

E.2.55. Terminology: Fields maxSize

E.2.56. Terminology: Fields maximumStoredSeqNo

The maximum transaction ID that has been stored locally on the machine in the THL. Because Tungsten Replicator operates in stages, it is sometimes important to compare the sequence and latency between information being ready from the source into the THL, and then from the THL into the database. You can compare this value to the appliedLastSeqno, which indicates the last sequence committed to the database. The information is provided at a resolution of milliseconds.

| maximumStoredSeqNo : 25 |

E.2.57. Terminology: Fields minimumStoredSeqNo

The minimum transaction ID stored locally in the THL on the host:

| minimumStoredSeqNo : 0 |

The figure should match the lowest transaction ID as output by the thl index command. On a busy host, or one where the THL information has been purged, the figure will show the corresponding transaction ID as stored in the THL.

E.2.58. Terminology: Fields name

E.2.59. Terminology: Fields offlineRequests

Contains the specifications of one or more future offline events that have been configured for the replicator. Multiple events are separated by a semicolon:

```
shell> trepctl status
...
minimumStoredSeqNo : 0
offlineRequests : Offline at sequence number: 5262;Offline at time: 2014-01-01 00:00:00 EST
pendingError : NONE
```

E.2.60. Terminology: Fields otherTime

E.2.61. Terminology: Fields pendingError


E.2.63. Terminology: Fields pendingErrorEventId

E.2.64. Terminology: Fields pendingErrorSeqno

The sequence number where the current error was identified

E.2.65. Terminology: Fields pendingExceptionMessage

The current error message that caused the current replicator offline

E.2.66. Terminology: Fields pipelineSource

The source for data for the current pipeline. On a master, the pipeline source is the database that the master is connected to and extracting data from. Within a slave, the pipeline source is the master replicator that is providing THL data.
E.2.67. Terminology: Fields \textit{processedMinSeqno}

E.2.68. Terminology: Fields \textit{queues}

E.2.69. Terminology: Fields \textit{readOnly}

E.2.70. Terminology: Fields \textit{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 \textit{relativeLatency} on the slave may indicate that the replicator may have stalled and stopped applying changes to the dataserver.

E.2.71. Terminology: Fields \textit{resourcePrecedence}

E.2.72. Terminology: Fields \textit{rmiPort}

E.2.73. Terminology: Fields \textit{role}

The current role of the host in the corresponding service specification. Primary roles are \textit{master} and \textit{slave}.

E.2.74. Terminology: Fields \textit{seqnoType}

The internal class used to store the transaction ID. In MySQL replication, the sequence number is typically stored internally as a Java Long [\texttt{java.lang.Long}]. In heterogeneous replication environments, the type used may be different to match the required information from the source database.

E.2.75. Terminology: Fields \textit{serializationCount}

E.2.76. Terminology: Fields \textit{serialized}

E.2.77. Terminology: Fields \textit{serviceName}

The name of the configured service, as defined when the deployment was first created through \texttt{tpm}.

\begin{verbatim}
serviceName : alpha
\end{verbatim}

A replicator may support multiple services. The information is output to confirm the service information being displayed.

E.2.78. Terminology: Fields \textit{serviceType}

The configured service type. Where the replicator is on the same host as the database, the service is considered to be \textit{local}. When reading or write to a remote dataserver, the service is \textit{remote}.

E.2.79. Terminology: Fields \textit{shard_id}

E.2.80. Terminology: Fields \textit{simpleServiceName}

A simplified version of the \textit{serviceName}. 

613
E.2.81. Terminology: Fields siteName

E.2.82. Terminology: Fields sourceId

E.2.83. Terminology: Fields stage

E.2.84. Terminology: Fields started

E.2.85. Terminology: Fields state

E.2.86. Terminology: Fields stopRequested

E.2.87. Terminology: Fields store.

E.2.88. Terminology: Fields storeClass

E.2.89. Terminology: Fields syncInterval

E.2.90. Terminology: Fields taskCount

E.2.91. Terminology: Fields taskId

E.2.92. Terminology: Fields timeInCurrentEvent

   Shows the time that the replicator has been processing the current event. When processing very large transactions this can be used to determine whether the replicator has stalled or is still actively extracting or applying the information.

E.2.93. Terminology: Fields timeInStateSeconds

E.2.94. Terminology: Fields timeoutMillis

E.2.95. Terminology: Fields totalAssignments

E.2.96. Terminology: Fields transitioningTo

E.2.97. Terminology: Fields uptimeSeconds

E.2.98. Terminology: Fields version
Appendix F. Internals

Tungsten Clustering includes a number of different systems and elements to provide the core services and functionality. Some of these are designed only to be customer-configured. Others should be changed only on the advice of Continuent or Continuent support. This chapter covers a range of different systems that are designated as internal features and functionality.

This chapter contains information on the following sections of Tungsten Clustering:

- Section F.1, “Extending Backup and Restore Behavior” — details on how the backup scripts operate and how to write custom backup scripts.
- Section F.2, “Character Sets in Database and Tungsten Clustering” — covers how character sets affect replication and command-line tool output.
- Section F.4, “Memory Tuning and Performance” — information on how the memory is used and allocated within Tungsten Clustering.

F.1. Extending Backup and Restore Behavior

The backup and restore system within Tungsten Clustering is handled entirely by the replicator. When a backup is initiated, the replicator on the specified datasource is asked to start the backup process.

The backup and restore system both use a modular mechanism that is used to perform the actual backup or restore operation. This can be configured to use specific backup tools or a custom script.

F.1.1. Backup Behavior

When a backup is requested, the Tungsten Replicator performs a number of separate, discrete, operations designed to perform the backup operation.

The backup operation performs the following steps:

1. Tungsten Replicator identifies the filename where properties about the backup will be stored. The file is used as the primary interface between the underlying backup script and Tungsten Replicator.
2. Tungsten Replicator executes the configured backup/restore script, supplying any configured arguments, and the location of a properties file, which the script updates with the location of the backup file created during the process.
3. If the backup completes successfully, the file generated by the backup process is copied into the configured Tungsten Clustering directory (for example /opt/continuent/backups).
4. Tungsten Replicator updates the property information with a CRC value for the backup file and the standard metadata for backups, including the tool used to create the backup.

A log is created of the backup process into a file according to the configured backup configuration. For example, when backing up using mysqldump the log is written to the log directory as mysqldump.log. When using a custom script, the log is written to script.log.

As standard, Tungsten Replicator supports two primary backup types, mysqldump and xtrabackup. A third option is based on the incremental version of the xtrabackup tool. The use of external backup script enables additional backup tools and methods to be supported.

To create a custom backup script, see Section F.1.3, “Writing a Custom Backup/Restore Script” for a list of requirements and samples.

F.1.2. Restore Behavior

The restore operation operates in a similar manner to the backup operation. The same script is called (but supplied with the -restore command-line option).

The restore operation performs the following steps:

1. Tungsten Replicator creates a temporary properties file, which contains the location of the backup file to be restored.
2. Tungsten Replicator executes the configured backup/restore script in restore mode, supplying any configured arguments, and the location of the properties file.
3. The script used during the restore process should read the supplied properties file to determine the location of the backup file.
4. The script performs all the necessary steps to achieve the restore process, including stopping the dataserver, restoring the data, and restarting the dataserver.
5. The replicator will remain in the **OFFLINE** state once the restore process has finished.

## F.1.3. Writing a Custom Backup/Restore Script

The synopsis of the custom script is as follows:

```
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
```
An alternative script using Perl is provided below:

```perl
#!/usr/bin/perl
use strict;
use warnings;
use Getopt::Long;
use IO::File;

my $argstring = join(' ',@ARGV);

my ($backup,$restore,$properties,$options) = (0,0,'','');
my $result = GetOptions("backup" => $backup,
  "restore" => $restore,
  "properties=s" => $properties,
  "options=s" => $options,
);

if ($backup)
{
  my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time);
  my $backupfile = sprintf('mcbackup.%04d%02d%02d-%02d%02d%02d-%02d.dump',
    ($year+1900),$mon,$mday,$hour,$min,$sec,$$);
  my $out = IO::File->new($backupfile,'w') or die "Couldn't open the backup file: $backupfile";
  # Fake backup data
  print $out "Backup data!
";
  $out->close();
  # Update the properties file
  my $profile = IO::File->new($properties,'w') or die "Couldn't write to the properties file";
  print $profile "file=$backupfile\n";
  $profile->close();
}

if ($restore)
{
  warn "Would be restoring information using $argstring\n";
}
exit 0;
```

F.1.4. Enabling a Custom Backup Script

To enable a custom backup script, the installation must be updated through `tpm` to use the script backup method. To update the configuration:

1. Create or copy the backup script into a suitable location, for example `/opt/continuent/share`.
2. Copy the script to each of the datasources within your dataservice.
3. Update the configuration using `tpm`. The `--repl-backup-method [439]` should be set to `script`, and the directory location set using the `--repl-backup-script [439]` 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 [439]` 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:
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 script, not the actual name of the script being used.

### F.2. Character Sets in Database and Tungsten Clustering

Character sets within the databases and within the configuration for Java and the wrappers for Tungsten Clustering must match to enable the information to be extracted and viewed.

For example, if you are extracting with the UTF-8 character set, the data must be applied to the target database using the same character set. In addition, the Tungsten Replicator should be configured with a corresponding matching character set. For installations where replication is between identical database flavours (for example, MySQL and MySQL) no explicit setting should be made. For heterogeneous deployments, the character set should be set explicitly.

When installing and using Tungsten Clustering, be aware of the following aspects when using character sets:

- When installing Tungsten Clustering, use the --java-file-encoding to configure the character set.
- When using the thl command, the character set may need to be explicitly stated to view the content correctly:

  ```none
  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. Understanding Replication of Date/Time Values

- Replicator processes default to UTC internally by setting the Java VM default time zone to UTC. This default can be changed by setting the replicator.time_zone property in the replicator services.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 Continuent support.
F.4. Memory Tuning and Performance

Different areas of Tungsten Clustering use memory in different ways, according to the operation and requirements of the component. Specific information on how memory is used by different components and how it is used is available below:

- **Tungsten Replicator** — Memory performance and tuning options.
- **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 once, hence need enough memory to hold all current objects. Second, the replicator works fastest if the in-memory queues between stages are large enough that they do not ever become empty. This keeps delays in upstream processing from delaying things at the end of the pipeline. Also, it allows 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 replicator.stage.q-to-dbms.blockCommitRowCount 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-svc-applicer-buffer-size` parameter.

- **Property replicator.global.buffer.size.** This sets the size of fragments for 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-applicer-buffer-size` parameter.

- **Property replicator.extractor.dbms.transaction_frag_size in the replicator.properties file.** This parameter controls the size of fragments for long transactions. Tungsten automatically breaks up long transactions into fragments. This parameter controls the number of bytes of bin-log per transaction fragment. You can try making this value smaller to reduce overall memory usage if many transactions are simultaneously present. Normally however this value has minimal impact.

Finally, it is worth mentioning that the main cause of out-of-memory conditions in replicators is large transactions. In particular, Tungsten cannot fragment individual statements or row changes, so changes to very large column values can also result in OutOfMemory conditions. For now the best approach is to raise memory, as described above, and change your application to avoid such transactions.
F.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` 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
```

F.5. Tungsten Replicator Pipelines and Stages

A pipeline (or service) acts upon data.

Pipelines consist of a variable number of stages.

Every stage’s workflow consists of three (3) actions, which are:

- Extract: the source for extraction could be the mysql server binary logs on a master, and the local THL on disk for a slave
- Filter: any configured filters are applied here
- Apply: the apply target can be THL on disk on a master, and the database server on a slave

Stages can be customized with filters, and filters are invoked on a per-stage basis.

By default, there are two pipeline services defined:

- Master replication service, which contains two (2) stages:
  - binlog-to-q: reads information from the MySQL binary log and stores the information within an in-memory queue.
  - q-to-thl: in-memory queue is written out to the THL file on disk.

- Slave replication service, which contains three (3) stages:
  - remote-to-thl: remote THL information is read from a master datasource and written to a local file on disk.
  - thl-to-q: THL information is read from the file on disk and stored in an in-memory queue.
  - q-to-dbms: data from the in-memory queue is written to the target database.

F.6. Tungsten Clustering Schemas
Appendix G. Frequently Asked Questions (FAQ)

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 fix the mysql-connectorj to drizzle MySQL driver bug which prevents my application from connecting through the Connector?

When upgrading from version 2 to v4+, or simply just moving away from the mysql-connectorj driver to the Drizzle driver, the update process doesn’t correctly remove all the connectorJ properties, causing a mismatch when connectors that did get the update try to make a connection to the cluster.

This is a known issue logged as CT-7

As yet, a fix has not been found, but the following workaround will correct the issue by hand:

To properly identify this issue, check the extended output of `cctrl` for the active driver. There will be one line of output for each node in the local cluster. Repeat once per cluster, on which node does not matter.

```shell
echo ls -l | cctrl -expert| grep driver: | awk '{print $3}'
```

For example, for a three-node cluster, you may see something like this:

```
com.mysql.jdbc.Driver
com.mysql.jdbc.Driver
com.mysql.jdbc.Driver
```

If any line on any node in any cluster shows the `com.mysql.jdbc.Driver`, please use the workaround below:

```

```

**Warning**
If you have multiple clusters, either MSMM, CMM or Composite HA/DR, always ensure you check ALL clusters. Especially in Composite clusters, the Master cluster, and especially the Master node, must be checked and corrected if necessary.

Ensure the `tpm update` was done with the `--replace-release` option.
Review the `tpm reverse` output and analyze based on the following:

- `--mysql-driver=drizzle` should exist in the defaults section
- You may (or may not) see the old `--mysql-connectorj-path` entry within each service definition or in the defaults
- If none of the above appear in the output, then the default drizzle driver will be active by default as of v4.0.0.

Repeat the following steps for all clusters, one by one:

1. Place the cluster into Maintenance Mode using the `cctrl` command:
   ```
cctrl> set policy maintenance
   ```

2. Stop all managers on all nodes within the single cluster:
   ```
shell> manager stop
Stopping Tungsten Manager Service...
Waiting for Tungsten Manager Service to exit...
Stopped Tungsten Manager Service.
   ```

3. On all nodes within the single cluster, remove all files from the `/opt/continuent/tungsten/cluster-home/conf/cluster/{local_service-name}/datasource/` directory.
   Only delete the files from the local cluster service name directory, do not touch the composite service directory if there is one.

4. Start all managers on all nodes within the single cluster, starting with the master:
   ```
shell> manager start
Starting Tungsten Manager Service...
Waiting for Tungsten Manager Service............
running: PID:24819
   ```

5. Place the cluster back into Automatic Mode
   ```
shell> echo set policy automatic | cctrl -expert
Tungsten Clustering 6.0.3 build 608
alpha: session established, encryption=false, authentication=false
[LOGICAL:EXPERT] /alpha > set policy automatic
policy mode is now AUTOMATIC
[LOGICAL:EXPERT] /alpha > Exiting...
   ```

Once the above has been completed, confirm that the procedure has worked as follows:

```
shell> echo ls -l | cctrl -expert| grep driver: | awk '{print $3}'
org.drizzle.jdbc.DrizzleDriver
org.drizzle.jdbc.DrizzleDriver
org.drizzle.jdbc.DrizzleDriver
   ```

G.1.3. How do I update the password for the replication user in the cluster?

If you need to change the password used by Tungsten Clustering 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.1.4. 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:
Frequently Asked Questions (FAQ)

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.1.5. Does the replicate filter (i.e. replicate.do and replicate.ignore) address both DML and DDL?

Both filters replicate.do and replicate.ignore will either do or ignore both DML and DDL.

DDL is currently ONLY replicated for MySQL to MySQL or Oracle to Oracle topologies, or within MySQL Clusters, although it would be advisable not to use ignore/do filters in a clustered environment where data/structural integrity is key.

With replicate.do, all DML and DDL will be replicated ONLY for any database or table listed as part of the do filter.

With replicate.ignore, all DML and DDL will be replicated except for any database or table listed as part of the ignore filter.

G.1.6. 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.1.7. On a Tungsten Replicator slave, how do I set both the local slave THL listener port and the upstream master’s THL listener port?

You need to specify two options: `thl-port` to set the slave THL listener port and `master-thl-port` to define the upstream master THL listener port. Otherwise `thl-port` alone sets BOTH.

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 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 3.6.1, “Adding Datasources to an Existing Deployment”.

G.2.4. Do you handle bandwidth/traffic management to the DB servers?

This is not something currently supported.
Appendix H. Ecosystem Support

In addition to the core utilities provided by Tungsten Clustering, additional tools and scripts are available that augment the core code with additional functionality, such as integrating with third-party monitoring systems, or providing additional functionality that is designed to be used and adapted for specific needs and requirements.

Different documentation and information exists for the following tools:

- Github — a selection of tools and utilities are provided in Github to further support and expand the functionality of Tungsten Clustering during deployment, monitoring, and management.

- logrotate — provides configuration information for users making use of the logrotate to manage Tungsten Clustering logs.

- Cacti — templates and scripts to enable monitoring through the Cacti environment.

- Nagios — templates and scripts to enable monitoring through the Nagios environment.
Appendix I. Configuration Property Reference