



Tungsten Replicator 5.0 Manual

Continuent Ltd

Tungsten Replicator 5.0 Manual

Continuent Ltd

Copyright © 2016 Continuent Ltd

Abstract

This manual documents Tungsten Replicator 5.0. This includes information for:

- Tungsten Replicator
- Tungsten Replicator for Oracle Replication
- Tungsten Replicator for Analytics and Big Data

Build date: 2017-03-29 (d10fe12)

Up to date builds of this document: [Tungsten Replicator 5.0 Manual \(Online\)](#), [Tungsten Replicator 5.0 Manual \(PDF\)](#)

Table of Contents

Preface	xvi
1. Legal Notice	xvi
2. Conventions	xvi
3. Quickstart Guide	xvii
1. Introduction	18
1.1. Tungsten Replicator	18
1.1.1. Extractor	19
1.1.2. Appliers	19
1.1.3. Transaction History Log (THL)	19
1.1.4. Filtering	19
2. Deployment	20
2.1. Deployment Sources	20
2.1.1. Using the TAR/GZipped files	21
2.1.2. Using the RPM and DEB package files	21
2.2. Best Practices	22
2.2.1. Best Practices: Deployment	22
2.2.2. Best Practices: Operations	23
2.2.3. Best Practices: Maintenance	23
2.3. Prepare Hosts	23
2.3.1. Prepare MySQL Hosts	23
2.3.2. Deploy SSH Keys	24
2.4. Deploy License Keys	25
2.5. Common <code>tpm</code> Options During Deployment	25
2.6. Starting and Stopping Tungsten Replicator	25
2.7. Configuring Startup on Boot	26
2.8. Removing Datasources from a Deployment	26
2.8.1. Removing a Datasource from an Existing Deployment	26
3. MySQL-only Deployments	27
3.1. Deploying a Master/Slave Topology	27
3.1.1. Monitoring a Master/Slave Dataservice	28
3.2. Deploying a Multi-master Topology	29
3.2.1. Preparing Hosts for Multimaster	30
3.2.2. Installing Multimaster Deployments	31
3.2.3. Management and Monitoring of Multimaster Deployments	32
3.2.4. Alternative Multimaster Deployments	34
3.3. Deploying a Fan-In Topology	34
3.3.1. Management and Monitoring Fan-in Deployments	36
3.4. Deploying Multiple Replicators on a Single Host	38
3.4.1. Prepare: Multiple Replicators	38
3.4.2. Install: Multiple Replicators	38
3.4.2.1. Deploying Multiple Replicators on a Single Host (Staging Use Case)	38
3.4.2.2. Deploying Multiple Replicators on a Single Host (INI Use Case)	39
3.4.3. Best Practices: Multiple Replicators	41
3.5. Replicating Data Out of a Cluster	41
3.5.1. Prepare: Replicating Data Out of a Cluster	42
3.5.2. Deploy: Replicating Data Out of a Cluster	43
3.5.2.1. Replicating from a Cluster to MySQL (Staging Use Case)	43
3.5.2.2. Replicating from a Cluster to MySQL (INI Use Case)	45
3.5.3. Best Practices: Replicating Data Out of a Cluster	46
3.6. Replicating Data Into an Existing Dataservice	46
4. Heterogeneous Deployments	50
4.1. How Heterogeneous Replication Works	50
5. Heterogeneous Oracle Deployments	53
5.1. Oracle Replication using Redo Reader	53
5.1.1. Oracle Redo Reader Replication Operation	55
5.1.2. Preparing the Oracle Environment for Replication	55
5.1.2.1. Prepare the Oracle Driver	55
5.1.2.2. Prepare the Oracle System User	55
5.1.2.3. Prepare the Oracle DBMS Servers	55
5.1.2.4. Prerequisite Checker Script	56
5.1.3. Installing an Oracle to Oracle Deployment	57
5.1.3.1. Installing a Standalone Oracle Extractor for Heterogeneous Deployments	57
5.1.4. Setting the Replication Start Position	59
5.1.4.1. Setting a Specific Start Position	59

5.1.4.2. Resetting Existing Replication to a specific SCN	60
5.1.4.3. Forcing replication to restart from a new SCN without resetting the THL	60
5.1.5. Provisioning an Oracle Replication Solution	60
5.1.5.1. Provisioning with Oracle Data Pump	60
5.1.6. Oracle Redo Reader Tuning	61
5.2. Deploying Oracle Replication for Disaster Recovery	61
5.2.1. Prepare: Oracle Replication for Disaster Recovery	61
5.2.2. Install: Oracle Replication for Disaster Recovery	61
5.2.2.1. Installing an Oracle for Disaster Recovery Topology (Staging Use Case)	62
5.2.2.2. Installing an Oracle for Disaster Recovery Topology (INI Use Case)	66
5.2.3. Best Practices: Oracle Replication for Disaster Recovery	71
5.2.3.1. Management and Monitoring of Oracle Replication for Disaster Recovery	71
5.2.3.2. Performing a Role Switch	71
5.3. Deploying Oracle Replication for Migration	72
5.3.1. Prepare: Oracle Replication for Migration	72
5.3.2. Install: Oracle Replication for Migration	72
5.3.2.1. Installing an Oracle for Migration Topology (Staging Use Case)	72
5.3.2.2. Installing an Oracle for Migration Topology (INI Use Case)	77
5.3.3. Best Practices: Oracle Replication for Migration	81
5.3.3.1. Management and Monitoring of Oracle Replication for Migration	81
5.4. Deploying Oracle to MySQL Replication	82
5.4.1. Prepare: Oracle to MySQL Replication	82
5.4.2. Install: Oracle to MySQL Replication	82
5.4.2.1. Installing an Oracle to MySQL Topology (Staging Use Case)	82
5.4.2.2. Installing an Oracle to MySQL Topology (INI Use Case)	88
5.4.3. Best Practices: Oracle to MySQL Replication	93
5.4.3.1. Management and Monitoring of Oracle to MySQL Deployments	93
5.5. Deploying Oracle to Hadoop Replication	94
5.5.1. Prepare: Oracle to Hadoop Replication	94
5.5.2. Install: Oracle to Hadoop Replication	94
5.5.2.1. Installing an Oracle to Hadoop Topology (Staging Use Case)	94
5.5.2.2. Installing an Oracle to Hadoop Topology (INI Use Case)	95
5.5.3. Best Practices: Oracle to Hadoop Replication	97
5.5.3.1. Management and Monitoring of Oracle to Hadoop Deployments	97
5.6. Deploying Oracle to Vertica Replication	97
5.6.1. Prepare: Oracle to Vertica Replication	98
5.6.2. Install: Oracle to Vertica Replication	100
5.6.2.1. Installing Oracle to Vertica Replication (Staging Method)	100
5.6.2.2. Installing an Oracle to Vertica (INI Use Case)	108
5.6.3. Best Practices: Oracle to Vertica Replication	114
5.6.3.1. Management and Monitoring of Oracle to Vertica Deployments	114
5.6.3.2. Troubleshooting Vertica Installations	115
5.7. Deploying Oracle Replication using CDC	115
5.7.1. How Oracle Extraction Works	117
5.7.2. Data Type Differences and Limitations	119
5.7.3. Creating an Oracle to MySQL Deployment	121
5.7.3.1. Configuring the Oracle Environment	121
5.7.3.2. Creating the MySQL Environment	122
5.7.3.3. Creating the Destination Schema	123
5.7.3.4. Creating the Master Replicator	123
5.7.3.5. Creating the Slave Replicator	125
5.7.4. Creating an Oracle to Oracle Deployment	127
5.7.4.1. Setting up the Source Oracle Environment	128
5.7.4.2. Setting up the Target Oracle Environment	130
5.7.4.3. Creating the Destination Schema	130
5.7.4.4. Installing the Master Replicator	131
5.7.4.5. Installing the Slave Replicator	133
5.7.5. Deployment with Provisioning	134
5.7.6. Updating CDC after Schema Changes	135
5.7.7. CDC Cleanup and Correction	136
5.7.8. Tuning CDC Extraction	136
5.7.9. Troubleshooting Oracle CDC Deployments	137
5.7.9.1. ORA-00257: ARCHIVER ERROR. CONNECT INTERNAL ONLY, UNTIL FREED	137
6. Heterogeneous MySQL Deployments	138
6.1. Deploying MySQL to Oracle Replication	138
6.1.1. Prepare: MySQL to Oracle Replication	138
6.1.2. Install: MySQL to Oracle Replication	139

6.1.2.1. Configure the MySQL database	140
6.1.2.2. Configure the Oracle database	140
6.1.2.3. Create the Destination Schema	141
6.1.2.4. Install the Master Replicator Service	141
6.1.2.5. Install Slave Replicator	142
6.2. Deploying MySQL to Hadoop Replication	143
6.2.1. Hadoop Replication Operation	144
6.2.2. Preparing Hosts for Hadoop Replication	145
6.2.2.1. MySQL Host	145
6.2.2.2. Hadoop Host	146
6.2.2.3. Schema Generation	147
6.2.3. Installing Hadoop Replication	147
6.2.3.1. MySQL to Hadoop Master Replicator Service	148
6.2.3.2. Oracle to Hadoop Master Replicator Service	149
6.2.3.3. Slave Replicator Service	149
6.2.4. Generating Materialized Views	162
6.2.5. Accessing Generated Tables in Hive	162
6.2.6. Management and Monitoring of Hadoop Deployments	162
6.2.6.1. Troubleshooting Hadoop Replication	163
6.3. Deploying MySQL to Amazon Redshift Replication	164
6.3.1. Redshift Replication Operation	165
6.3.2. Preparing Hosts for Amazon Redshift Deployments	167
6.3.2.1. MySQL Preparation for Amazon Redshift Deployments	167
6.3.2.2. Redshift Preparation for Amazon Redshift Deployments	167
6.3.2.3. Amazon Redshift DDL Generation for Amazon Redshift Deployments	168
6.3.3. Installing Amazon Redshift Replication	169
6.3.4. Verifying your Redshift Installation	171
6.3.5. Keeping CDC Information	172
6.3.6. Management and Monitoring of Amazon Redshift Deployments	172
6.3.7. Troubleshooting Amazon Redshift Installations	173
6.4. Deploying MySQL to Vertica Replication	174
6.4.1. Preparing Hosts for Vertica Deployments	174
6.4.2. Installing Vertica Replication	176
6.4.3. Management and Monitoring of Vertica Deployments	178
6.4.4. Troubleshooting Vertica Installations	180
7. Advanced Deployments	181
7.1. Deploying Parallel Replication	181
7.1.1. Application Prerequisites for Parallel Replication	181
7.1.2. Enabling Parallel Apply	181
7.1.3. Channels	181
7.1.4. Disk vs. Memory Parallel Queues	182
7.1.5. Parallel Replication and Offline Operation	183
7.1.5.1. Clean Offline Operation	183
7.1.5.2. Tuning the Time to Go Offline Cleanly	183
7.1.5.3. Unclean Offline	183
7.1.6. Adjusting Parallel Replication After Installation	183
7.1.6.1. How to Change Channels Safely	183
7.1.6.2. How to Switch Parallel Queue Types Safely	184
7.1.7. Monitoring Parallel Replication	184
7.1.7.1. Useful Commands for Parallel Monitoring Replication	184
7.1.7.2. Parallel Replication and Applied Latency On Slaves	184
7.1.7.3. Relative Latency	185
7.1.7.4. Serialization Count	185
7.1.7.5. Maximum Offline Interval	185
7.1.7.6. Workload Distribution	186
7.1.8. Controlling Assignment of Shards to Channels	187
7.2. Batch Loading for Data Warehouses	188
7.2.1. How It Works	188
7.2.2. Important Limitations	188
7.2.3. Batch Applier Setup	189
7.2.4. JavaScript Batchloader Scripts	189
7.2.4.1. JavaScript Batchloader Scripts <code>apply()</code> Function	190
7.2.4.2. JavaScript Batchloader Scripts <code>begin()</code> Function	190
7.2.4.3. JavaScript Batchloader Scripts <code>commit()</code> Function	190
7.2.4.4. JavaScript Batchloader Scripts <code>prepare()</code> Function	190
7.2.4.5. JavaScript Batchloader Scripts <code>release()</code> Function	191
7.2.4.6. JavaScript Batchloader with Parallel Apply	191

7.2.4.7. Batchloading Javascript Scripts	191
7.2.5. Staging Tables	191
7.2.5.1. Staging Table Names	191
7.2.5.2. Whole Record Staging	191
7.2.5.3. Delete Key Staging	192
7.2.5.4. Staging Table Generation	192
7.2.6. Character Sets	192
7.2.7. CSV Formats	192
7.2.8. Time Zones	192
7.3. Additional Configuration and Deployment Options	192
7.3.1. Deploying Multiple Replicators on a Single Host	193
7.4. Deploying SSL Secured Replication and Administration	194
7.4.1. Creating the Truststore and Keystore	194
7.4.1.1. Creating Your Own Client and Server Certificates	194
7.4.1.2. Creating a Custom Certificate and Getting it Signed	196
7.4.1.3. Using an existing Certificate	197
7.4.1.4. Converting SSL Certificates for keytool	198
7.4.2. SSL and Administration Authentication	198
7.4.3. Configuring the Secure Service through tpm	199
7.5. Deployment Security	201
7.5.1. Disabling Security	202
7.5.2. Creating Suitable Certificates	202
7.5.3. Installing from a Staging Host with Manually Generated Certificates	202
7.5.4. Installing via INI File with Manually Generated Certificates	202
7.5.5. Replacing the TLS Certificate from a Staging Directory	202
7.5.6. Removing TLS Encryption from a Staging Directory	203
8. Operations Guide	204
8.1. The Tungsten Replication Home Directory	204
8.2. Establishing the Shell Environment	204
8.3. Checking Replication Status	204
8.3.1. Understanding Replicator States	207
8.3.2. Replicator States During Operations	207
8.3.3. Changing Replicator States	208
8.4. Managing Transaction Failures	208
8.4.1. Identifying a Transaction Mismatch	209
8.4.2. Skipping Transactions	211
8.5. Provision or Reprovision a Slave	211
8.6. Creating a Backup	212
8.6.1. Using a Different Backup Tool	213
8.6.2. Using a Different Directory Location	213
8.6.3. Creating an External Backup	213
8.7. Restoring a Backup	213
8.7.1. Restoring a Specific Backup	214
8.7.2. Restoring an External Backup	214
8.7.3. Restoring from Another Slave	215
8.7.4. Manually Recovering from Another Slave	215
8.8. Deploying Automatic Replicator Recovery	216
8.9. Migrating and Seeding Data	217
8.9.1. Migrating from MySQL Native Replication 'In-Place'	217
8.9.2. Migrating from MySQL Native Replication Using a New Service	218
8.9.3. Seeding Data through MySQL	220
8.9.4. Seeding Data through tungsten_provision_thl	220
8.10. Using the Parallel Extractor	220
8.10.1. Advanced Configuration Parameters	222
8.11. Switching Master Hosts	223
8.12. Configuring Parallel Replication	224
8.13. Performing Database or OS Maintenance	226
8.13.1. Performing Maintenance on a Single Slave	226
8.13.2. Performing Maintenance on a Master	226
8.13.3. Performing Maintenance on an Entire Dataservice	227
8.13.4. Upgrading or Updating your JVM	227
8.14. Making Online Schema Changes	228
8.15. Upgrading Tungsten Replicator	229
8.15.1. Upgrading Tungsten Replication to use tpm	229
8.15.2. Upgrading Tungsten Replication using tpm	230
8.15.3. Installing an Upgraded JAR Patch	232
8.16. Monitoring Tungsten Replication	233

8.16.1. Managing Log Files with <code>logrotate</code>	233
8.16.2. Monitoring Status Using <code>cacti</code>	233
8.16.3. Monitoring Status Using <code>nagios</code>	235
9. Command-line Tools	236
9.1. The <code>check_tungsten_latency</code> Command	236
9.2. The <code>check_tungsten_online</code> Command	237
9.3. The <code>check_tungsten_services</code> Command	237
9.4. The <code>deployall</code> Command	238
9.5. The <code>ddlscan</code> Command	238
9.5.1. Optional Arguments	240
9.5.2. Supported Templates and Usage	240
9.5.2.1. <code>ddl-check-pkeys.vm</code>	241
9.5.2.2. <code>ddl-mysql-hive-0.10.vm</code>	241
9.5.2.3. <code>ddl-mysql-hive-0.10-staging.vm</code>	242
9.5.2.4. <code>ddl-mysql-hive-metadata.vm</code>	243
9.5.2.5. <code>ddl-mysql-oracle.vm</code>	243
9.5.2.6. <code>ddl-mysql-oracle-cdc.vm</code>	244
9.5.2.7. <code>ddl-mysql-redshift.vm</code>	244
9.5.2.8. <code>ddl-mysql-redshift-staging.vm</code>	244
9.5.2.9. <code>ddl-mysql-vertica.vm</code>	245
9.5.2.10. <code>ddl-mysql-vertica-staging.vm</code>	246
9.5.2.11. <code>ddl-oracle-mysql.vm</code>	246
9.5.2.12. <code>ddl-oracle-mysql-pk-only.vm</code>	247
9.6. The <code>dsctl</code> Command	247
9.6.1. <code>dsctl get</code> Command	248
9.6.2. <code>dsctl set</code> Command	248
9.6.3. <code>dsctl reset</code> Command	248
9.6.4. <code>dsctl help</code> Command	248
9.7. <code>env.sh</code> Script	249
9.8. The <code>load-reduce-check</code> Tool	249
9.8.1. Generating Staging DDL	249
9.8.2. Generating Live DDL	249
9.8.3. Materializing a View	249
9.8.4. Compare Loaded Data	249
9.9. The <code>materialize</code> Command	249
9.10. The <code>multi_trepctl</code> Command	249
9.10.1. <code>multi_trepctl</code> Options	250
9.10.2. <code>multi_trepctl</code> Commands	252
9.10.2.1. <code>multi_trepctl backups</code> Command	252
9.10.2.2. <code>multi_trepctl heartbeat</code> Command	252
9.10.2.3. <code>multi_trepctl masterof</code> Command	253
9.10.2.4. <code>multi_trepctl list</code> Command	253
9.10.2.5. <code>multi_trepctl run</code> Command	253
9.11. The <code>replicator</code> Command	253
9.12. The <code>setupCDC.sh</code> Command	255
9.13. The <code>startall</code> Command	258
9.14. The <code>stopall</code> Command	259
9.15. The <code>thl</code> Command	259
9.15.1. <code>thl list</code> Command	259
9.15.2. <code>thl index</code> Command	262
9.15.3. <code>thl purge</code> Command	262
9.15.4. <code>thl info</code> Command	264
9.15.5. <code>thl help</code> Command	264
9.16. The <code>trepctl</code> Command	264
9.16.1. <code>trepctl</code> Options	265
9.16.2. <code>trepctl</code> Global Commands	265
9.16.2.1. <code>trepctl kill</code> Command	266
9.16.2.2. <code>trepctl services</code> Command	266
9.16.2.3. <code>trepctl version</code> Command	267
9.16.3. <code>trepctl</code> Service Commands	268
9.16.3.1. <code>trepctl backup</code> Command	268
9.16.3.2. <code>trepctl capabilities</code> Command	269
9.16.3.3. <code>trepctl check</code> Command	270
9.16.3.4. <code>trepctl clear</code> Command	270
9.16.3.5. <code>trepctl clients</code> Command	270
9.16.3.6. <code>trepctl flush</code> Command	271
9.16.3.7. <code>trepctl heartbeat</code> Command	271

9.16.3.8. <code>trepctl load</code> Command	273
9.16.3.9. <code>trepctl offline</code> Command	273
9.16.3.10. <code>trepctl offline-deferred</code> Command	273
9.16.3.11. <code>trepctl online</code> Command	274
9.16.3.12. <code>trepctl properties</code> Command	277
9.16.3.13. <code>trepctl purge</code> Command	278
9.16.3.14. <code>trepctl reset</code> Command	278
9.16.3.15. <code>trepctl restore</code> Command	279
9.16.3.16. <code>trepctl setrole</code> Command	279
9.16.3.17. <code>trepctl shard</code> Command	279
9.16.3.18. <code>trepctl status</code> Command	280
9.16.3.19. <code>trepctl unload</code> Command	287
9.16.3.20. <code>trepctl wait</code> Command	287
9.17. The <code>tpasswd</code> Command	288
9.18. The <code>tungsten_provision_thl</code> Command	288
9.18.1. Provisioning from RDS	289
9.18.2. <code>tungsten_provision_thl</code> Reference	289
9.19. The <code>tungsten_provision_slave</code> Script	293
9.20. The <code>tungsten_read_master_events</code> Script	294
9.21. The <code>tungsten_set_position</code> Script	295
9.22. The <code>undeployall</code> Command	296
9.23. The <code>updateCDC.sh</code> Command	296
9.24. The <code>vmrr</code> Command	297
10. The <code>tpm</code> Deployment Command	298
10.1. Comparing Staging and <code>INI</code> <code>tpm</code> Methods	299
10.2. Processing Installs and Upgrades	301
10.3. <code>tpm</code> Staging Configuration	302
10.3.1. Configuring default options for all services	302
10.3.2. Configuring a single service	302
10.3.3. Configuring a single host	303
10.3.4. Reviewing the current configuration	303
10.3.5. Installation	303
10.3.5.1. Installing a set of specific services	303
10.3.5.2. Installing a set of specific hosts	303
10.3.6. Upgrades from a Staging Directory	303
10.3.7. Configuration Changes from a Staging Directory	304
10.3.8. Converting from INI to Staging	304
10.4. <code>tpm</code> INI File Configuration	305
10.4.1. Creating an INI file	305
10.4.2. Installation with INI File	306
10.4.3. Upgrades with an INI File	306
10.4.4. Configuration Changes with an INI file	306
10.4.5. Converting from Staging to INI	306
10.5. <code>tpm</code> Commands	307
10.5.1. <code>tpm configure</code> Command	308
10.5.2. <code>tpm diag</code> Command	308
10.5.3. <code>tpm fetch</code> Command	308
10.5.4. <code>tpm firewall</code> Command	309
10.5.5. <code>tpm help</code> Command	309
10.5.6. <code>tpm install</code> Command	309
10.5.7. <code>tpm mysql</code> Command	310
10.5.8. <code>tpm query</code> Command	310
10.5.8.1. <code>tpm query config</code>	310
10.5.8.2. <code>tpm query dataservices</code>	311
10.5.8.3. <code>tpm query deployments</code>	311
10.5.8.4. <code>tpm query manifest</code>	311
10.5.8.5. <code>tpm query modified-files</code>	311
10.5.8.6. <code>tpm query staging</code>	311
10.5.8.7. <code>tpm query version</code>	312
10.5.9. <code>tpm reset</code> Command	312
10.5.10. <code>tpm reset-thl</code> Command	312
10.5.11. <code>tpm restart</code> Command	312
10.5.12. <code>tpm reverse</code> Command	312
10.5.13. <code>tpm ssh-copy-cert</code> Command	313
10.5.14. <code>tpm start</code> Command	313
10.5.15. <code>tpm stop</code> Command	314
10.5.16. <code>tpm update</code> Command	314

10.5.17. <code>tpm validate</code> Command	314
10.5.18. <code>tpm validate-update</code> Command	315
10.6. <code>tpm</code> Common Options	315
10.7. <code>tpm</code> Configuration Options	315
10.7.1. A <code>tpm</code> Options	327
10.7.2. B <code>tpm</code> Options	330
10.7.3. C <code>tpm</code> Options	331
10.7.4. D <code>tpm</code> Options	335
10.7.5. E <code>tpm</code> Options	342
10.7.6. F <code>tpm</code> Options	346
10.7.7. H <code>tpm</code> Options	346
10.7.8. I <code>tpm</code> Options	347
10.7.9. J <code>tpm</code> Options	348
10.7.10. L <code>tpm</code> Options	350
10.7.11. M <code>tpm</code> Options	351
10.7.12. N <code>tpm</code> Options	355
10.7.13. O <code>tpm</code> Options	356
10.7.14. P <code>tpm</code> Options	358
10.7.15. Q <code>tpm</code> Options	362
10.7.16. R <code>tpm</code> Options	362
10.7.17. S <code>tpm</code> Options	364
10.7.18. T <code>tpm</code> Options	368
10.7.19. U <code>tpm</code> Options	370
10.7.20. V <code>tpm</code> Options	370
10.7.21. W <code>tpm</code> Options	370
11. Replication Filters	371
11.1. Enabling/Disabling Filters	372
11.2. Enabling Additional Filters	373
11.3. Filter Status	374
11.4. Filter Reference	374
11.4.1. <code>ansiquotes.js</code> Filter	376
11.4.2. <code>BidiRemoteSlave (BidiSlave)</code> Filter	377
11.4.3. <code>breadcrumbs.js</code> Filter	377
11.4.4. <code>BuildAuditTable</code> Filter	378
11.4.5. <code>BuildIndexTable</code> Filter	379
11.4.6. <code>CaseMapping (CaseTransform)</code> Filter	379
11.4.7. <code>CDCMetadata (CustomCDC)</code> Filter	379
11.4.8. <code>ColumnName</code> Filter	380
11.4.9. <code>ConsistencyCheck</code> Filter	381
11.4.10. <code>DatabaseTransform (dbtransform)</code> Filter	381
11.4.11. <code>dbrename.js</code> Filter	382
11.4.12. <code>dbselector.js</code> Filter	382
11.4.13. <code>dbupper.js</code> Filter	383
11.4.14. <code>dropcolumn.js</code> Filter	384
11.4.15. <code>dropcomments.js</code> Filter	385
11.4.16. <code>dropmetadata.js</code> Filter	385
11.4.17. <code>dropstatementdata.js</code> Filter	386
11.4.18. Dummy Filter	386
11.4.19. <code>EnumToString</code> Filter	386
11.4.20. <code>EventMetadata</code> Filter	387
11.4.21. <code>foreignkeychecks.js</code> Filter	388
11.4.22. Heartbeat Filter	388
11.4.23. <code>insertsonly.js</code> Filter	389
11.4.24. Logging Filter	389
11.4.25. <code>MySQLSessionSupport (mysqlsessions)</code> Filter	389
11.4.26. NetworkClient Filter	389
11.4.26.1. Network Client Configuration	390
11.4.26.2. Network Filter Protocol	391
11.4.26.3. Sample Network Client	393
11.4.27. <code>nocreatedbifnotexists.js</code> Filter	395
11.4.28. OptimizeUpdates Filter	395
11.4.29. PrimaryKey Filter	396
11.4.30. PrintEvent Filter	397
11.4.31. Rename Filter	398
11.4.31.1. Rename Filter Examples	399
11.4.32. ReplicateColumns Filter	400
11.4.33. Replicate Filter	400

11.4.34. SetToString Filter	401
11.4.35. Shard Filter	402
11.4.36. <code>shardbyseqno.js</code> Filter	403
11.4.37. <code>shardbytable.js</code> Filter	403
11.4.38. TimeDelay (delay) Filter	404
11.4.39. <code>tosingledb.js</code> Filter	404
11.4.40. <code>truncatetext.js</code> Filter	405
11.4.41. <code>zerodate2null.js</code> Filter	405
11.5. JavaScript Filters	406
11.5.1. Writing JavaScript Filters	407
11.5.1.1. Implementable Functions	407
11.5.1.2. Getting Configuration Parameters	408
11.5.1.3. Logging Information and Exceptions	408
11.5.1.4. Exposed Data Structures	408
12. Performance and Tuning	415
12.1. Block Commit	415
12.1.1. Monitoring Block Commit Status	416
12.2. Improving Network Performance	416
12.3. Tungsten Replicator Block Commit and Memory Usage	417
A. Release Notes	420
A.1. Tungsten Replicator 5.0.1 GA (23 Feb 2017)	420
A.2. Tungsten Replicator 5.0.0 GA (7 December 2015)	421
B. Prerequisites	426
B.1. Requirements	426
B.1.1. Operating Systems Support	426
B.1.2. Database Support	426
B.1.3. RAM Requirements	426
B.1.4. Disk Requirements	427
B.1.5. Java Requirements	427
B.1.6. Cloud Deployment Requirements	427
B.2. Staging Host Configuration	428
B.3. Host Configuration	429
B.3.1. Creating the User Environment	430
B.3.2. Configuring Network and SSH Environment	430
B.3.2.1. Network Ports	431
B.3.2.2. SSH Configuration	432
B.3.3. Directory Locations and Configuration	432
B.3.4. Configure Software	432
B.3.5. <code>sudo</code> Configuration	433
B.4. MySQL Database Setup	434
B.4.1. MySQL Version Support	434
B.4.2. MySQL Configuration	434
B.4.3. MySQL User Configuration	437
B.4.4. MySQL Unprivileged Users	437
B.5. Oracle Database Setup	438
B.5.1. Oracle Version Support	438
B.5.2. Oracle Environment Variables	438
C. Troubleshooting	439
C.1. Contacting Support	439
C.1.1. Support Request Procedure	439
C.1.2. Creating a Support Account	439
C.1.3. Generating Diagnostic Information	439
C.1.4. Open a Support Ticket	440
C.1.5. Open a Support Ticket via Email	440
C.1.6. Getting Updates for all Company Support Tickets	440
C.1.7. Support Severity Level Definitions	440
C.2. Error/Cause/Solution	441
C.2.1. There were issues configuring the sandbox MySQL server	441
C.2.2. Unable to update the configuration of an installed directory	441
C.2.3. 'subscription exists' when setting up CDC on Oracle	442
C.2.4. Attempt to write new log record with equal or lower fragno: seqno=3 previous stored fragno=32767 attempted new fragno=32768	442
C.2.5. The session variable <code>SQL_MODE</code> when set to include <code>ALLOW_INVALID_DATES</code> does not apply statements correctly on the slave	442
C.2.6. Too many open processes or files	443
C.2.7. Replicator runs out of memory	443
C.2.8. <code>MySQLExtractException: unknown data type 0</code>	444

C.2.9. Services requires a reset	444
C.2.10. OptimizeUpdatesFilter cannot filter, because column and key count is different. Make sure that it is defined before filters which remove keys (eg. PrimaryKeyFilter)	445
C.2.11. ORA-00257: ARCHIVER ERROR. CONNECT INTERNAL ONLY, UNTIL FREED	445
C.3. Known Issues	445
C.3.1. Triggers	445
C.4. Troubleshooting Timeouts	446
C.5. Troubleshooting Backups	446
C.6. Running Out of Diskspace	446
C.7. Troubleshooting SSH and <i>tpm</i>	446
C.8. Troubleshooting Data Differences	447
C.8.1. Identify Structural Differences	447
C.8.2. Identify Data Differences	447
C.9. Comparing Table Data	448
C.10. Troubleshooting Memory Usage	448
D. Files, Directories, and Environment	449
D.1. The Tungsten Replication Install Directory	449
D.1.1. The <i>backups</i> Directory	449
D.1.1.1. Automatically Deleting Backup Files	449
D.1.1.2. Manually Deleting Backup Files	450
D.1.1.3. Copying Backup Files	450
D.1.1.4. Relocating Backup Storage	451
D.1.2. The <i>releases</i> Directory	452
D.1.3. The <i>service_logs</i> Directory	452
D.1.4. The <i>share</i> Directory	453
D.1.5. The <i>thl</i> Directory	453
D.1.5.1. Purging THL Log Information on a Slave	454
D.1.5.2. Purging THL Log Information on a Master	454
D.1.5.3. Moving the THL File Location	455
D.1.5.4. Changing the THL Retention Times	456
D.1.6. The <i>tungsten</i> Directory	456
D.1.6.1. The <i>tungsten-replicator</i> Directory	457
D.2. Log Files	457
D.3. Environment Variables	457
E. Terminology Reference	458
E.1. Transaction History Log (THL)	458
E.1.1. THL Format	458
E.2. Generated Field Reference	461
E.2.1. Terminology: Fields <i>accessFailures</i>	461
E.2.2. Terminology: Fields <i>active</i>	461
E.2.3. Terminology: Fields <i>activeSeqno</i>	462
E.2.4. Terminology: Fields <i>appliedLastEventId</i>	462
E.2.5. Terminology: Fields <i>appliedLastSeqno</i>	462
E.2.6. Terminology: Fields <i>appliedLatency</i>	462
E.2.7. Terminology: Fields <i>applier.class</i>	462
E.2.8. Terminology: Fields <i>applier.name</i>	462
E.2.9. Terminology: Fields <i>applyTime</i>	462
E.2.10. Terminology: Fields <i>autoRecoveryEnabled</i>	462
E.2.11. Terminology: Fields <i>autoRecoveryTotal</i>	462
E.2.12. Terminology: Fields <i>averageBlockSize</i>	462
E.2.13. Terminology: Fields <i>blockCommitRowCount</i>	463
E.2.14. Terminology: Fields <i>cancelled</i>	463
E.2.15. Terminology: Fields <i>channel</i>	463
E.2.16. Terminology: Fields <i>channels</i>	463
E.2.17. Terminology: Fields <i>clusterName</i>	463
E.2.18. Terminology: Fields <i>commits</i>	463
E.2.19. Terminology: Fields <i>committedMinSeqno</i>	463
E.2.20. Terminology: Fields <i>criticalPartition</i>	463
E.2.21. Terminology: Fields <i>currentBlockSize</i>	463
E.2.22. Terminology: Fields <i>currentEventId</i>	463
E.2.23. Terminology: Fields <i>currentLastEventId</i>	463
E.2.24. Terminology: Fields <i>currentLastFragno</i>	463
E.2.25. Terminology: Fields <i>currentLastSeqno</i>	463
E.2.26. Terminology: Fields <i>currentTimeMillis</i>	463
E.2.27. Terminology: Fields <i>dataServerHost</i>	463
E.2.28. Terminology: Fields <i>discardCount</i>	463
E.2.29. Terminology: Fields <i>doChecksum</i>	463

E.2.30. Terminology: Fields <code>estimatedOfflineInterval</code>	464
E.2.31. Terminology: Fields <code>eventCount</code>	464
E.2.32. Terminology: Fields <code>extensions</code>	464
E.2.33. Terminology: Fields <code>extractTime</code>	464
E.2.34. Terminology: Fields <code>extractor.class</code>	464
E.2.35. Terminology: Fields <code>extractor.name</code>	464
E.2.36. Terminology: Fields <code>filter.#.class</code>	464
E.2.37. Terminology: Fields <code>filter.#.name</code>	464
E.2.38. Terminology: Fields <code>filterTime</code>	464
E.2.39. Terminology: Fields <code>flushIntervalMillis</code>	464
E.2.40. Terminology: Fields <code>fsyncOnFlush</code>	464
E.2.41. Terminology: Fields <code>headSeqno</code>	464
E.2.42. Terminology: Fields <code>intervalGuard</code>	464
E.2.43. Terminology: Fields <code>lastCommittedBlockSize</code>	464
E.2.44. Terminology: Fields <code>lastCommittedBlockTime</code>	464
E.2.45. Terminology: Fields <code>latestEpochNumber</code>	464
E.2.46. Terminology: Fields <code>logConnectionTimeout</code>	464
E.2.47. Terminology: Fields <code>logDir</code>	465
E.2.48. Terminology: Fields <code>logFileRetainMillis</code>	465
E.2.49. Terminology: Fields <code>logFileSize</code>	465
E.2.50. Terminology: Fields <code>masterConnectUri</code>	465
E.2.51. Terminology: Fields <code>masterListenUri</code>	465
E.2.52. Terminology: Fields <code>maxChannel</code>	465
E.2.53. Terminology: Fields <code>maxDelayInterval</code>	465
E.2.54. Terminology: Fields <code>maxOfflineInterval</code>	465
E.2.55. Terminology: Fields <code>maxSize</code>	465
E.2.56. Terminology: Fields <code>maximumStoredSeqNo</code>	465
E.2.57. Terminology: Fields <code>minimumStoredSeqNo</code>	465
E.2.58. Terminology: Fields <code>name</code>	465
E.2.59. Terminology: Fields <code>offlineRequests</code>	465
E.2.60. Terminology: Fields <code>otherTime</code>	466
E.2.61. Terminology: Fields <code>pendingError</code>	466
E.2.62. Terminology: Fields <code>pendingErrorCode</code>	466
E.2.63. Terminology: Fields <code>pendingErrorEventId</code>	466
E.2.64. Terminology: Fields <code>pendingErrorSeqno</code>	466
E.2.65. Terminology: Fields <code>pendingExceptionMessage</code>	466
E.2.66. Terminology: Fields <code>pipelineSource</code>	466
E.2.67. Terminology: Fields <code>processedMinSeqno</code>	466
E.2.68. Terminology: Fields <code>queues</code>	466
E.2.69. Terminology: Fields <code>readonly</code>	466
E.2.70. Terminology: Fields <code>relativeLatency</code>	466
E.2.71. Terminology: Fields <code>resourcePrecedence</code>	466
E.2.72. Terminology: Fields <code>rmiPort</code>	466
E.2.73. Terminology: Fields <code>role</code>	466
E.2.74. Terminology: Fields <code>seqnoType</code>	467
E.2.75. Terminology: Fields <code>serializationCount</code>	467
E.2.76. Terminology: Fields <code>serialized</code>	467
E.2.77. Terminology: Fields <code>serviceName</code>	467
E.2.78. Terminology: Fields <code>serviceType</code>	467
E.2.79. Terminology: Fields <code>shard_id</code>	467
E.2.80. Terminology: Fields <code>simpleServiceName</code>	467
E.2.81. Terminology: Fields <code>siteName</code>	467
E.2.82. Terminology: Fields <code>sourceId</code>	467
E.2.83. Terminology: Fields <code>stage</code>	467
E.2.84. Terminology: Fields <code>started</code>	467
E.2.85. Terminology: Fields <code>state</code>	467
E.2.86. Terminology: Fields <code>stopRequested</code>	467
E.2.87. Terminology: Fields <code>store.#</code>	467
E.2.88. Terminology: Fields <code>storeClass</code>	467
E.2.89. Terminology: Fields <code>syncInterval</code>	467
E.2.90. Terminology: Fields <code>taskCount</code>	467
E.2.91. Terminology: Fields <code>taskId</code>	468
E.2.92. Terminology: Fields <code>timeInStateSeconds</code>	468
E.2.93. Terminology: Fields <code>timeoutMillis</code>	468
E.2.94. Terminology: Fields <code>totalAssignments</code>	468
E.2.95. Terminology: Fields <code>transitioningTo</code>	468
E.2.96. Terminology: Fields <code>uptimeSeconds</code>	468

E.2.97. Terminology: Fields <i>version</i>	468
F. Internals	469
F.1. Extending Backup and Restore Behavior	469
F.1.1. Backup Behavior	469
F.1.2. Restore Behavior	469
F.1.3. Writing a Custom Backup/Restore Script	470
F.1.4. Enabling a Custom Backup Script	471
F.2. Character Sets in Database and Tungsten Replication	472
F.3. Understanding Replication of Date/Time Values	472
F.4. Memory Tuning and Performance	473
F.4.1. Understanding Tungsten Replicator Memory Tuning	473
F.5. Tungsten Replicator Stages	473
F.6. Tungsten Replication Schemas	474
G. Frequently Asked Questions (FAQ)	475
H. Ecosystem Support	476
H.1. Continuent Github Repositories	476
I. Configuration Property Reference	477

List of Figures

3.1. Topologies: Master/Slave	27
3.2. Topologies: Multiple-masters	30
3.3. Topologies: Fan-in	35
3.4. Topologies: Replicating Data Out of a Cluster	42
3.5. Topologies: Replicating into a Dataservice	47
4.1. Topologies: Heterogeneous Operation	50
5.1. Topologies: Oracle to Oracle with Redo Reader	54
5.2. Topologies: Oracle to Oracle with Redo Reader for Migration	72
5.3. Topologies: Oracle to MySQL with Redo Reader	82
5.4. Topologies: Oracle to Hadoop with Redo Reader	94
5.5. Topologies: Oracle to HP Vertica with Redo Reader	98
5.6. Topologies: MySQL to Oracle	116
5.7. Topologies: Oracle to MySQL	116
5.8. Topologies: Oracle to Oracle	117
5.9. Oracle Extraction with Synchronous CDC	118
5.10. Oracle Extraction with Asynchronous CDC	119
6.1. Topologies: MySQL to Oracle	138
6.2. Topologies: MySQL to Hadoop	143
6.3. Topologies: MySQL to Hadoop Replication Operation	144
6.4. Topologies: MySQL to Amazon Redshift	165
6.5. Topologies: MySQL to Redshift Replication Operation	165
6.6. Topologies: MySQL to Vertica	174
7.1. Batchloading: JavaScript	190
8.1. Migration: Migrating Native Replication using a New Service	219
8.2. Parallel Extractor: Extraction Sequence	221
8.3. Parallel Extractor: Extraction Operation	221
8.4. Cacti Monitoring: Example Graphs	234
10.1. tpm Staging Based Deployment	299
10.2. tpm INI Based Deployment	300
11.1. Filters: Pipeline Stages on Masters	371
11.2. Filters: Pipeline Stages on Slaves	372
B.1. Tungsten Deployment	428

List of Tables

2.1. Key Terminology	20
5.1. Data Type differences when replicating data from MySQL to Oracle	119
5.2. Data Type Differences when Replicating from Oracle to MySQL or Oracle	120
5.3. <code>setupCDC.conf</code> Configuration File Parameters	128
6.1. Data Type differences when replicating data from MySQL to Oracle	139
6.2. Hadoop Replication Directory Locations	145
9.1. <code>check_tungsten_latency</code> Options	236
9.2. <code>check_tungsten_online</code> Options	237
9.3. <code>check_tungsten_services</code> Options	238
9.4. <code>ddlscan</code> Command-line Options	239
9.5. <code>ddlscan</code> Supported Templates	240
9.6. <code>dsctl</code> Commands	247
9.7. <code>dsctl</code> Command-line Options	247
9.8. <code>dsctl</code> Command-line Options	248
9.9. <code>multi_trepctl</code> Command-line Options	250
9.10. <code>multi_trepctl--output</code> Option	251
9.11. <code>multi_trepctl</code> Commands	252
9.12. <code>replicator</code> Commands	253
9.13. <code>replicator</code> Commands Options for <code>condrestart</code>	254
9.14. <code>replicator</code> Commands Options for <code>console</code>	254
9.15. <code>replicator</code> Commands Options for <code>restart</code>	254
9.16. <code>replicator</code> Commands Options for <code>start</code>	255
9.17. <code>setupCDC.conf</code> Configuration Options	255
9.18. <code>thl</code> Options	259
9.19. <code>trepctl</code> Command-line Options	265
9.20. <code>trepctl</code> Replicator Wide Commands	266
9.21. <code>trepctl</code> Service Commands	268
9.22. <code>trepctl backup</code> Command Options	268
9.23. <code>trepctl clients</code> Command Options	270
9.24. <code>trepctl offline-deferred</code> Command Options	273
9.25. <code>trepctl online</code> Command Options	274
9.26. <code>trepctl purge</code> Command Options	278
9.27. <code>trepctl reset</code> Command Options	278
9.28. <code>trepctl setrole</code> Command Options	279
9.29. <code>trepctl shard</code> Command Options	280
9.30. <code>trepctl status</code> Command Options	280
9.31. <code>trepctl wait</code> Command Options	287
9.32. <code>tpasswd</code> Common Options	288
9.33. <code>tungsten_provision_slave</code> Command-line Options	293
9.34. <code>tungsten_read_master_events</code> Command-line Options	294
9.35. <code>tungsten_set_position</code> Command-line Options	295
10.1. TPM Deployment Methods	300
10.2. <code>tpm</code> Core Options	307
10.3. <code>tpm</code> Commands	307
10.4. <code>tpm</code> Common Options	315
10.5. <code>tpm</code> Configuration Options	316
D.1. Continuent Tungsten Directory Structure	449
D.2. Continuent Tungsten <code>tungsten</code> Sub-Directory Structure	456
E.1. THL Event Format	459

Preface

This manual documents Tungsten Replication 5.0 up to and including 5.0.1 build 136. Differences between minor versions are highlighted stating the explicit minor release version, such as 5.0.1.x.

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

1. Legal Notice

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

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

All rights reserved.

2. Conventions

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

- *Text in this style* is used to show an important element or piece of information. It may be used and combined with other text styles as appropriate to the context.
- **Text in this style** is used to show a section heading, table heading, or particularly important emphasis of some kind.
- Program or configuration options are formatted using *this style*. Options are also automatically linked to their respective documentation page when this is known. For example, `tpm` and `--hosts` [347] both link automatically to the corresponding reference page.
- Parameters or information explicitly used to set values to commands or options is formatted using *this style*.
- Option values, for example on the command-line are marked up using this format: `--help`. Where possible, all option values are directly linked to the reference information for that option.
- Commands, including sub-commands to a command-line tool are formatted using **Text in this style**. Commands are also automatically linked to their respective documentation page when this is known. For example, `tpm` links automatically to the corresponding reference page.
- *Text in this style* indicates literal or character sequence text used to show a specific value.
- Filenames, directories or paths are shown like this `/etc/passwd`. Filenames and paths are automatically linked to the corresponding reference page if available.

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

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

```
shell> cd /opt/staging  
shell> unzip tungsten-replicator-5.0.1-136.zip
```

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

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

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

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

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

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

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

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

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

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

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

3. Quickstart Guide

- Are you planning on completing your first installation?
 - Have you followed the [Appendix B, Prerequisites](#)?
 - Have you chosen your deployment type? See [Chapter 2, Deployment](#)
 - Is this a [Master/Slave deployment](#)?
 - Are you replicating to or from [Oracle](#)?
 - Are you trying to configure [Hadoop](#) replication?
 - Are you trying to configure [Vertica](#) replication?
- 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 10.1, "Comparing Staging and INI tpm Methods"](#).

- Do you want to upgrade to the latest version?

See [Section 10.5.16, "tpm update Command"](#).

- Are you trying to update or change the configuration of your system?

See [Section 10.5.16, "tpm update Command"](#).

- Would you like to perform database or operating system maintenance?

See [Section 8.13, "Performing Database or OS Maintenance"](#).

- Do you need to backup or restore your system?

For backup instructions, see [Section 8.6, "Creating a Backup"](#), and to restore a previously made backup, see [Section 8.7, "Restoring a Backup"](#).

Chapter 1. Introduction

Tungsten Replicator™ is an open source replication engine supporting a variety of different extractor and applier modules. Data can be extracted from MySQL, Oracle and Amazon RDS, and applied to transactional stores, including MySQL, Oracle, and Amazon RDS; NoSQL stores such as MongoDB, and datawarehouse stores such as Vertica, InfiniDB, Hadoop and Amazon Redshift.

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

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

Features in Tungsten Replicator 3.0

- Includes support for replicating into Hadoop (including Apache Hadoop, Cloudera, HortonWorks, MapR, Amazon EMR)
- Includes support for replicating into Amazon Redshift, including storing change data within Amazon S3
- Includes support for replicating to and from Amazon RDS (MySQL) deployments
- SSL Support for managing MySQL deployments
- Network Client filter for handling complex data translation/migration needs during replication

1.1. Tungsten Replicator

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

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

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

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

- **Extractor**

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

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

- **Applier**

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

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

- **Transaction History Log (THL)**

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

As the THL data is stored additional information, such as the metadata and options in place when the statement or row data was extracted are recorded. Each transaction is also recorded with an incremental global transaction ID. This ID enables individual transactions with-

in the THL to be identified, for example to retrieve their content, or to determine whether different replicators within a replication topology have written a specific transaction to a data server.

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

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

1.1.1. Extractor

Extractors exist for reading information from the following sources:

- MySQL
- Oracle

1.1.2. Appliers

The replicator commits transactions using block commit meaning it only commits on x transactions. This improves performance but when using a non-transactional engine it can cause the problems you have seen. By default this is set to 10 (the value is `replicator.global.buffer.size`). It is possible to set this to 1 which will remove the problem with MyISAM tables but it will impact the performance of the replicators

Available appliers include:

- MongoDB
- MySQL
- Oracle
- Vertica

For more information on how the replicator for heterogeneous works, see [Section 4.1, “How Heterogeneous Replication Works”](#). For more information on the batch applier, which works with datawarehouse targets, see [Section 7.2, “Batch Loading for Data Warehouses”](#).

1.1.3. Transaction History Log (THL)

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

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

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

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

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

1.1.4. Filtering

For more information on the filters available, and how to use them, see [Chapter 11, Replication Filters](#).

Chapter 2. Deployment

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

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

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

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

Table 2.1. Key Terminology

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

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

1. [Setup a staging host](#) from which you will configure and manage your installation.
2. [Configure each host](#) that will be used within your dataservice.
3. Depending on the database or environment you are using, you may need to perform additional configuration steps for the dataserver:
 - [Configure your MySQL installation](#), so that Tungsten Replication can work with the database.
 - [Configure your Oracle installation](#), so that Tungsten Replication can work with the database.

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

2.1. Deployment Sources

Tungsten Replication is available in a number of different distribution types, and the methods for configuration available for these different packages differs. See [Section 10.1, "Comparing Staging and INI tpm Methods"](#) for more information on the available installation methods.

Deployment Type/Package	TAR/GZip	RPM/DEB
tpm Command-line Configuration	Yes	Yes
tpm INI File Configuration	Yes	Yes
Deploy Entire Cluster	Yes	No
Deploy Per Machine	No	Yes

Two primary deployment sources are available:

- [Tar/GZip](#)

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

- [RPM/DEB Packages](#)

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

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

```
tungsten-replicator-5.0.1-136.tar.gz
```

The version number is `5.0.1` and build number `136`. Build numbers indicate which build a particular release version is based on, and may be useful when installing patches provided by support.

2.1.1. Using the TAR/GZipped files

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

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-replicator-5.0.1-136.tar.gz
```

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

```
shell> cd tungsten-replicator-5.0.1-136
```

2.1.2. Using the RPM and DEB package files

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

Installation

Installing the RPM or DEB package will do the following:

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

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

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

Note

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

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

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

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

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

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

- Download from an existing download page.
- For **yum** platforms (RHEL/CentOS/Amazon Linux), add the package source to your **yum** configuration. For the current stable (GA) release packages:

```
root-shell> rpm -i http://releases.continuent.com.s3.amazonaws.com/replicator-release-stable-0.0-1.x86_64.rpm
```

For nightly builds:

```
root-shell> rpm -i http://releases.continuent.com.s3.amazonaws.com/replicator-release-nightly-0.0-1.x86_64.rpm
```

- For Ubuntu/Debian packages:

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

Nightly builds are also available:

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

Then update your **apt** repository:

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

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

- On RHEL/CentOS/Amazon Linux:

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

- On Ubuntu/Debian:

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

Upgrades

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

1. Unpack the software into `/opt/continuent/software`
2. Run **tpm update** if the `/etc/tungsten.ini` or `/etc/tungsten/tungsten.ini` file exists

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

2.2. Best Practices

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

2.2.1. Best Practices: Deployment

- Identify the best deployment method for your environment and use that in production and testing. See [Section 10.1, “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.3.3, “Directory Locations and Configuration”](#).
 - When using ssh and/or SSL, ensure that the ssh key or certificates are suitably protected. See [Section B.3.2.2, “SSH Configuration”](#).
 - Use a firewall, such as **iptables** to protect the network ports that you need to use. The best solution is to ensure that only known hosts can connect to the required ports for Tungsten Replication. For more information on the network ports required for Tungsten Replication operation, see [Section B.3.2.1, “Network Ports”](#).
 - If possible, use authentication and SSL connectivity between hosts to protect your data and authorisation for the tools used in your deployment.

See [Section 7.5, “Deployment Security”](#) for more information.

- Choose your topology from the deployment section and verify the configuration matches the basic settings. Additional settings may be included for custom features but the basics are needed to ensure proper operation. If your configuration is not listed or does not match our documented settings; we cannot guarantee correct operation.
- If you are using `ROW` replication, any triggers that run additional `INSERT/UPDATE/DELETE` operations must be updated so they do not run on the slave servers. See <http://kb.vmware.com/kb/2112599>.
- Make sure you know the structure of the Tungsten Replication home directory and how to initialize your environment for administration. See [Section 8.1, "The Tungsten Replication Home Directory"](#) and [Section 8.2, "Establishing the Shell Environment"](#).
- Prior to migrating applications to Tungsten Replication test failover and recovery procedures from [Chapter 8, Operations Guide](#). Be sure to try recovering a failed master and reprovisioning failed slaves.

2.2.2. Best Practices: Operations

- Setup proper monitoring for all servers as described in [Section 8.16, "Monitoring Tungsten Replication"](#).
- Configure the Tungsten Replication services to startup and shutdown along with the server. See [Section 2.7, "Configuring Startup on Boot"](#).

2.2.3. Best Practices: Maintenance

- Your license allows for a testing cluster. Deploy a cluster that matches your production cluster and test all operations and maintenance operations there.
- Disable any automatic operating system patching processes. The use of automatic patching will cause issues when all database servers automatically restart without coordination. See [Section 8.13.3, "Performing Maintenance on an Entire Dataservice"](#).
- Regularly check for maintenance releases and upgrade your environment. Every version includes stability and usability fixes to ease the administrative process.

2.3. Prepare Hosts

Using Puppet is the fastest way to prepare a host for Tungsten Replication. These instructions will show you how to install Puppet and prepare a host to run Tungsten Replication. 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 > 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 > apt-get update
shell > apt-get install -y ruby ruby-dev puppet
```

- Install the Continuent Puppet module.

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

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

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

2.3.1. Prepare MySQL Hosts

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

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

```

installMysql => true,
replicationUser => 'tungsten',
replicationPassword => 'secret',
appUser => 'app_user',
appPassword => 'secret',
} "

```

2.3.2. Deploy SSH Keys

The [tpm](#) script uses SSH to execute commands on each host. There are two simple ways to install these keys.

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

```

shell > puppet apply -e "class { 'tungsten' :
sshPublicKey => "-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAxoTELWB3x3f2FhpYk6PFpI0h18+TF9AjJVCmmYXrCuOPOSn
Qo1ZCcDjU85yZfGKvxcZS12eQmLkQKL5REt4W7MdbhH81jLq0E5xOrBH64AxMAZ
aFBrxw3pyHAoFr7wuUE+5wSO13KHwfj7FzsugvXriGnuM+BL88Wqh9m6cO8H6g
oz6Rah5bd93Bj1OxbNgcMq4Ob1qHu6Dr0ohvXdf0io+g+p8b4ST14tAg680HfeC
snoXXmzfNpxi4OBLLX9rKUXria+OgWalj7z9G5YAO1tbODZhsW9kX8KT3KoJ8B/XT
wJq81MfA18yVtcsLk0UnaDzgbnXjbTE+DulwIDAQABoIBAcn6+4lpAAzjh9vG
uIKIOIzYtTddwsTHcuPUzvXm65gCSU++UvtxaFlXnPtxYdfW+rJMQVx5M0V4F
zz5isqQsjSY70SN23Maba/BdcgkN09kHdtd/ly6yx0k1wy1hn1Qmd+6q+A91Ph
bn8K1J/5z8KLHOTQi1XvpvC4/s8CVP+j/7CZMnua7Y5yjvYVv3NCULySYgYci1w
VsG5V+1cB9pvrs86k/yjb20nL64cozX2Iy8g3aCYT8MPk7babKsfcfzwP5yTUS
cauNNMBY81WIKuTQ23Tzh5y/iu7dlnwi3svzIDLApc/XUrv9ovr4R+0E0JG01Fps
2tbbGaECgYEAS5s40W9ndDs00PaOp5Kx1agPcLy8Qm/aIxdsxzNeutlHuUvQHuz
W1Kw2pziT+QeQxa6RjbvgRle2eg5T9fI/QPRKHvkQVJ/xuB7qPGdn0ibNHJMpQsB
sqPKk5btcpCnTiM0VYCNJNzwF884JoIKj4axf0oetcbg/rKPAmnX1EcgYEAs3cAg
9c7iHACoZBTGP52cm0rQgFugtz382L517sbJtfvtyTMjXoIjzWu53U8Au+sfhwg
F5Do0AmdqJ11hXiqyD5oIRAQwp1Dm3gCvU278WucOfLkCxtTDOBog8aC4Dj+
TyisczxTxLlv1S2paKs4Y3GL3DD8wNpEgN4dBwcCgYAzjycUkrCDpCrKHg2vDbJ
n2XTAcX4onVI0P98K+QD8wn7lssBqXKZLGGcZGK8EgODyCFIXsaE3ulzP6091SL
EMOpXGX2hQgvDyeixAb8/d3k+jdrzJLpxoAuHhtRz8nnzkl3zv1Wa8ck+kT8HsL
4MDgoIEXLWkgabOveZ761QKBgHXKJLftWPX+86rULiQeIAOkzfQgt9IePzzyhKL
JPO+gXGLHozlq7jejzWrdGEq5rlmpXZfz8V/oQG8j/Eoo8Brc3o0G+31o+JcfwX
V30u2XJ38teIOrjdBVvMHDYimtKKLrvA7KMMUQC1h2xNIwgRdxrRUDgGUAI/rY
ARppAoGae6GOoxEdic748n4+Khlj/bHZwdID7QhjExCJ83NQ3Jd11IWqR+9B0tDs
90jpedj4K24TLd8i12+zQ135/j1XQaQezXXOSwfNfgRqvnxGm6IIoNX86VijXV1
nIO2XfxjbVcom391bgGfnxk78vSLuvHM3Iva6cGmBwqTzI7lxQ=
-----END RSA PRIVATE KEY-----",
sshPrivateCert => "AAAAB3NzaC1yc2EAAAQABAAQDGHMQtYHfHd/YWGliTo8WmJCHXz5MX0CMlUKaZhdGsK4485KdCiVkJwMm7znJ18Yq/Fx1KXZ5CYuRAosv1ES3hbxslu
) "

```

- 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 on each host before proceeding with deployment.

```

shell > ./tools/tpm ssh-copy-cert
mkdir -p ~/.ssh
echo "-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAxoTELWB3x3f2FhpYk6PFpI0h18+TF9AjJVCmmYXrCuOPOSn
Qo1ZCcDjU85yZfGKvxcZS12eQmLkQKL5REt4W7MdbhH81jLq0E5xOrBH64AxMAZ
aFBrxw3pyHAoFr7wuUE+5wSO13KHwfj7FzsugvXriGnuM+BL88Wqh9m6cO8H6g
oz6Rah5bd93Bj1OxbNgcMq4Ob1qHu6Dr0ohvXdf0io+g+p8b4ST14tAg680HfeC
snoXXmzfNpxi4OBLLX9rKUXria+OgWalj7z9G5YAO1tbODZhsW9kX8KT3KoJ8B/XT
wJq81MfA18yVtcsLk0UnaDzgbnXjbTE+DulwIDAQABoIBAcn6+4lpAAzjh9vG
uIKIOIzYtTddwsTHcuPUzvXm65gCSU++UvtxaFlXnPtxYdfW+rJMQVx5M0V4F
zz5isqQsjSY70SN23Maba/BdcgkN09kHdtd/ly6yx0k1wy1hn1Qmd+6q+A91Ph
bn8K1J/5z8KLHOTQi1XvpvC4/s8CVP+j/7CZMnua7Y5yjvYVv3NCULySYgYci1w
VsG5V+1cB9pvrs86k/yjb20nL64cozX2Iy8g3aCYT8MPk7babKsfcfzwP5yTUS
cauNNMBY81WIKuTQ23Tzh5y/iu7dlnwi3svzIDLApc/XUrv9ovr4R+0E0JG01Fps
2tbbGaECgYEAS5s40W9ndDs00PaOp5Kx1agPcLy8Qm/aIxdsxzNeutlHuUvQHuz
W1Kw2pziT+QeQxa6RjbvgRle2eg5T9fI/QPRKHvkQVJ/xuB7qPGdn0ibNHJMpQsB
sqPKk5btcpCnTiM0VYCNJNzwF884JoIKj4axf0oetcbg/rKPAmnX1EcgYEAs3cAg
9c7iHACoZBTGP52cm0rQgFugtz382L517sbJtfvtyTMjXoIjzWu53U8Au+sfhwg
F5Do0AmdqJ11hXiqyD5oIRAQwp1Dm3gCvU278WucOfLkCxtTDOBog8aC4Dj+
TyisczxTxLlv1S2paKs4Y3GL3DD8wNpEgN4dBwcCgYAzjycUkrCDpCrKHg2vDbJ
n2XTAcX4onVI0P98K+QD8wn7lssBqXKZLGGcZGK8EgODyCFIXsaE3ulzP6091SL
EMOpXGX2hQgvDyeixAb8/d3k+jdrzJLpxoAuHhtRz8nnzkl3zv1Wa8ck+kT8HsL
4MDgoIEXLWkgabOveZ761QKBgHXKJLftWPX+86rULiQeIAOkzfQgt9IePzzyhKL
JPO+gXGLHozlq7jejzWrdGEq5rlmpXZfz8V/oQG8j/Eoo8Brc3o0G+31o+JcfwX
V30u2XJ38teIOrjdBVvMHDYimtKKLrvA7KMMUQC1h2xNIwgRdxrRUDgGUAI/rY
ARppAoGae6GOoxEdic748n4+Khlj/bHZwdID7QhjExCJ83NQ3Jd11IWqR+9B0tDs
90jpedj4K24TLd8i12+zQ135/j1XQaQezXXOSwfNfgRqvnxGm6IIoNX86VijXV1
nIO2XfxjbVcom391bgGfnxk78vSLuvHM3Iva6cGmBwqTzI7lxQ=
-----END RSA PRIVATE KEY-----" > ~/.ssh/id_rsa
echo "ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQDGHMQtYHfHd/YWGliTo8WmJCHXz5MX0CMlUKaZhdGsK4485KdCiVkJwMm7znJ18Yq/Fx1KXZ5CYuRAosv1ES3hbxslu
touch ~/.ssh/authorized_keys
cat ~/.ssh/id_rsa.pub>>~/.ssh/authorized_keys
chmod 700 ~/.ssh

```

```
chmod 600 ~/.ssh/*
```

2.4. Deploy License Keys

License keys are provided to all customers with an active support contract. Login to my.vmware.com to identify your support contract and the associated license keys. After collecting the license keys, they should be placed into `/etc/tungsten/continuent.licenses` or `/opt/continuent/share/continuent.licenses`. The `/opt/continuent` path should be replaced with your value for `--install-directory` [347]. Place each license on a new line in the file and make sure it is readable by the tungsten system user.

If you are testing VMware Continuent or don't have your license key, talk with your sales contact for assistance. You may enable a trial-mode by using the license key `TRIAL`. This will not affect the runtime operation of VMware Continuent but may impact your ability to get rapid support.

The `tpm` script will display a warning if license keys are not provided or if the provided license keys are not valid.

2.5. 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 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 10.5.16, “`tpm update Command`” for more information on using `tpm update`.

- `--datasource-systemctl-service` [339]

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` [339] to specify the name of the tool.

The format 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` [339], the servicename, and the status (i.e. `stop`).

2.6. Starting and Stopping Tungsten Replicator

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

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

Note

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

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

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

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

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

For some scenarios, such as initiating a load within a heterogeneous environment, the replicator can be started up in the `OFFLINE` [207] state:

```
shell> replicator start offline
```

If the cluster was configured with `auto-enable=false` [329] then you will need to put each node online individually.

2.7. Configuring Startup on Boot

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

```
shell> sudo deployall
```

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

```
shell> sudo undeployall
```

2.8. Removing Datasources from a Deployment

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

In this section:

- [Section 2.8.1, “Removing a Datasource from an Existing Deployment”](#)

2.8.1. Removing a Datasource from an Existing Deployment

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

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

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

```
shell> replicator stop
```

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

1. Now you must remove the node from the configuration, although the exact method depends on which installation method used with `tpm`:

- If you are using staging directory method with `tpm`:
 - a. Change to the staging directory. The current staging directory can be located using `tpm query staging`:

```
shell> tpm query staging
tungsten@host1:/home/tungsten/tungsten-replicator-5.0.1-136
shell> cd /home/tungsten/tungsten-replicator-5.0.1-136
```

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

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

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

- Remove the INI configuration file:

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

2. Remove the installed software directory:

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

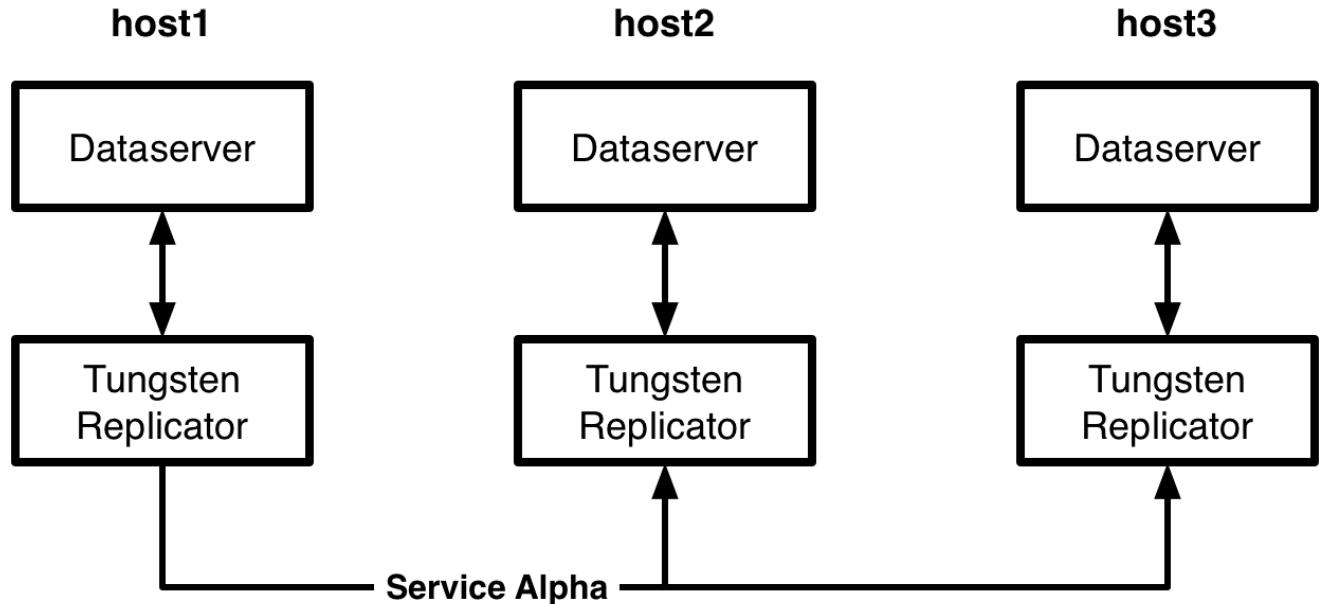
Chapter 3. MySQL-only Deployments

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

3.1. Deploying a Master/Slave Topology

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

Figure 3.1. Topologies: Master/Slave



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

```
shell> ./tools/tpm install alpha \
    --topology=master-slave \
    --master=host1 \
    --replication-user=tungsten \
    --replication-password=password \
    --install-directory=/opt/continuent \
    --members=host1,host2,host3 \
    --start
```

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

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

- [tpm install](#)

Executes **tpm** in **install** mode to create the service [alpha](#).

- [--master=host1 \[351\]](#)

Specifies which host will be the master.

- [--replication-user=tungsten \[363\]](#)

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

- [--replication-password=password \[363\]](#)

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

- `--install-directory=/opt/continuent [347]`

Directory where Tungsten Replication will be installed.

- `--members=host1,host2,host3 [351]`

List of all the hosts within the cluster, including the master host. Hosts in this list that do not appear in the `--master [351]` option will be configured as slaves.

- `--start [365]`

Starts the service once installation is complete.

If the MySQL configuration file cannot be located, the `--datasource=mysql-conf [337]` option can be used to specify its location:

```
shell> ./tools/tpm install alpha \
    --topology=master-slave \
    --master=host1 \
    --replication-user=tungsten \
    --replication-password=password \
    --datasource=mysql-conf=/etc/mysql/my.cnf \
    --install-directory=/opt/continuent \
    --members=host1,host2,host3 \
    --start
```

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

Once the installation has been completed, the service will be started and ready to use. For information on checking the running service, see [Section 3.1.1, “Monitoring a Master/Slave Dataservice”](#).

For information on starting and stopping Tungsten Replication see [Section 2.6, “Starting and Stopping Tungsten Replicator”](#); configuring init scripts to startup and shutdown when the system boots and shuts down, see [Section 2.7, “Configuring Startup on Boot”](#).

3.1.1. Monitoring a Master/Slave Dataservice

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

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

The key fields are:

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

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

```
shell> treectl status
Processing status command...
NAME          VALUE
----          -----
appliedLastEventId : mysql-bin.000009:0000000000001033;0
appliedLastSeqno  : 3593
appliedLatency   : 1.074
channels        : 1
clusterName     : default
currentEventId  : mysql-bin.000009:0000000000001033
currentTimeMillis: 1373615598598
dataServerHost   : host1
extensions      :
latestEpochNumber: 3589
masterConnectUri :
```

```

masterListenUri      : thl://host1:2112/
maximumStoredSeqNo  : 3593
minimumStoredSeqNo  : 0
offlineRequests     : NONE
pendingError        : NONE
pendingErrorCode    : NONE
pendingErrorEventId : NONE
pendingErrorSeqno   : -1
pendingExceptionMessage: NONE
pipelineSource      : jdbc:mysql:thin://host1:3306/
relativeLatency    : 604904.598
resourcePrecedence : 99
rmiPort             : 10000
role                : master
seqnoType           : java.lang.Long
serviceName          : alpha
serviceType          : local
simpleServiceName   : alpha
siteName            : default
sourceId            : host1
state               : ONLINE
timeInStateSeconds : 604903.621
transitioningTo    :
uptimeSeconds       : 1202137.328
version             : Tungsten Replicator 5.0.1 build 136
Finished status command...

```

Checking a remote slave:

```

shell> trepctl -host host2 status
Processing status command...
NAME          VALUE
----          -----
appliedLastEventId : mysql-bin.000009:000000000001033:0
appliedLastSeqno  : 3593
appliedLatency   : 605002.401
channels        : 5
clusterName     : default
currentEventId  : NONE
currentTimeMillis: 1373615698912
dataServerHost   : host2
extensions      :
latestEpochNumber: 3589
masterConnectUri: thl://host1:2112/
masterListenUri  : thl://host2:2112/
maximumStoredSeqNo: 3593
minimumStoredSeqNo: 0
offlineRequests  : NONE
pendingError     : NONE
pendingErrorCode : NONE
pendingErrorEventId: NONE
pendingErrorSeqno: -1
pendingExceptionMessage: NONE
pipelineSource   : thl://host1:2112/
relativeLatency : 605004.912
resourcePrecedence: 99
rmiPort          : 10000
role              : slave
seqnoType         : java.lang.Long
serviceName        : alpha
serviceType        : local
simpleServiceName : alpha
siteName          : default
sourceId          : host2
state             : ONLINE
timeInStateSeconds: 2.944
transitioningTo   :
uptimeSeconds     : 1202243.752
version           : Tungsten Replicator 5.0.1 build 136
Finished status command...

```

For more information on using `trepctl`, see [Section 9.16, “The `trepctl` Command”](#).

Definitions of the individual field descriptions in the above example output can be found in [Section E.2, “Generated Field Reference”](#).

For more information on management and operational detailed for managing your cluster installation, see [Chapter 8, *Operations Guide*](#).

3.2. Deploying a Multi-master Topology

When configuring a multi-master topology, `tpm` automatically creates a number of individual services that are used to define a master/slave topology between each group of hosts. In a three-node multimaster setup, three different services are created, each service cre-

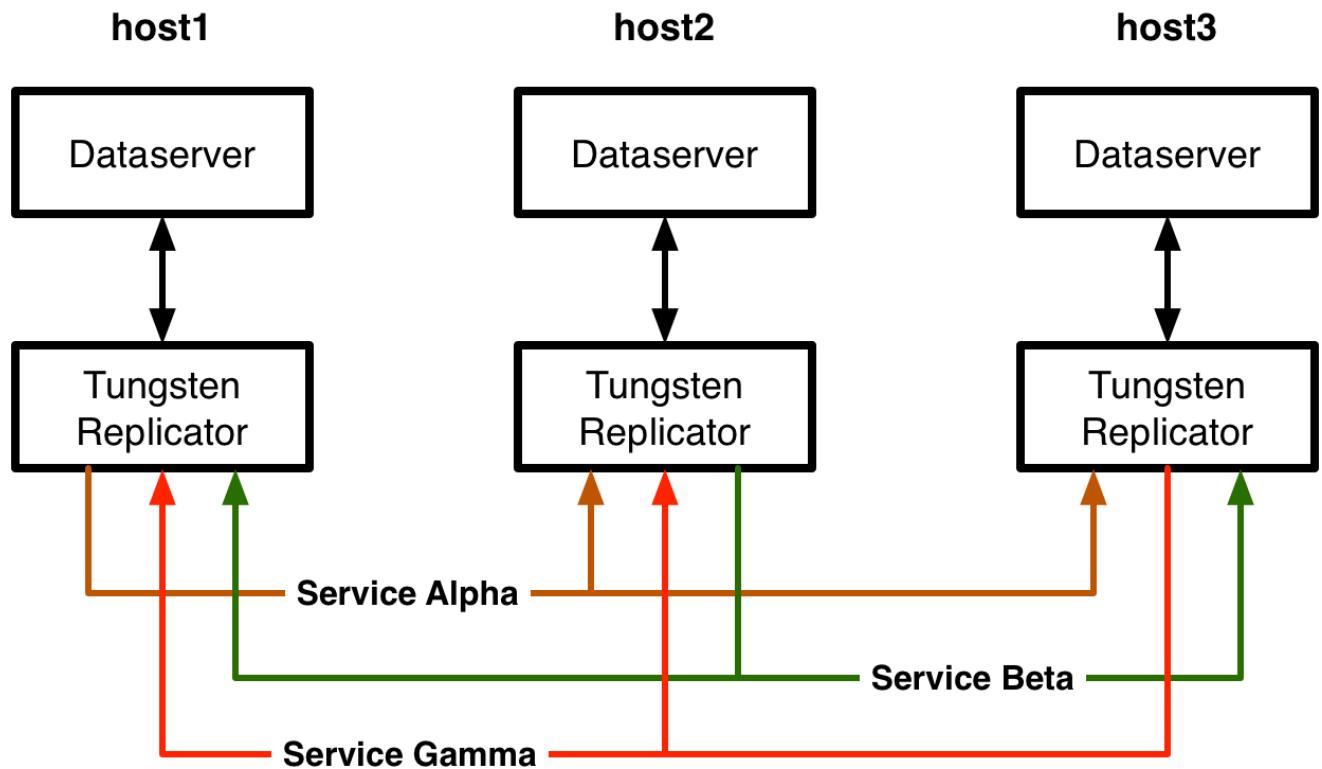
ates a master/slave relationship between a primary host and the slaves. A change on any individual host will be replicated to the other databases in the topology creating the multi-master configuration.

For example, with three hosts, `host1`, `host2`, and `host3`, three separate configurations are created:

- `host1` is the master, and `host2` and `host3` are slaves of `host1` (Service *Alpha*, yellow)
- `host2` is the master, and `host1` and `host3` are slaves of `host2` (Service *Beta*, green)
- `host3` is the master, and `host1` and `host2` are slaves of `host3` (Service *Gamma*, red)

[Figure 3.2, "Topologies: Multiple-masters"](#) shows the structure of the configuration replication.

Figure 3.2. Topologies: Multiple-masters



These three individual services, one for each host and two slave scenario, effectively create a multi-master topology, since a change on any single master will be replicated to the slaves.

3.2.1. Preparing Hosts for Multimaster

Some considerations must be taken into account for any multi-master scenario:

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

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

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

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

```
binlog-format = row
```

- Beware of triggers. Triggers can cause problems during replication because if they are applied on the slave as well as the master you can get data corruption and invalid data. Tungsten Replication cannot prevent triggers from executing on a slave, and in a multi-master 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"](#).
- Ensure that the `server-id` for each MySQL configuration has been modified and is different on each host. This will help to prevent the application of data originating on the a server being re-applied if the transaction is replicated again from another master after the initial replication. Tungsten Replication is designed not to replicate these statements, and uses the server ID as part of the identification process.

3.2.2. Installing Multimaster Deployments

To create the configuration use `tpm` to create the entire configuration with just one command. Before starting the installation, the prerequisites must have been completed (see [Appendix B, Prerequisites](#)). This takes the list of hosts, and a list of master services that will be configured, and then creates each service automatically:

1. On your staging server, download the release package.
2. Unpack the release package:

```
shell> tar zxf continuum-tungsten-5.0.1-136.tar.gz
```

3. Change to the unpackaged directory:

```
shell> cd continuum-tungsten-5.0.1-136
```

4. Create the installation using the `tpm`:

```
shell> ./tools/tpm install epsilon \
--topology=all-masters \
--install-directory=/opt/continuum \
--replication-user=tungsten \
--replication-password=secret \
--master=host1,host2,host3 \
--members=host1,host2,host3 \
--master-services=alpha,beta,gamma \
--start
```

Host and service information is extracted in corresponding sequence as provided in the command-line options.

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.

- Creates a service, `alpha`, with `host1` as master and the other hosts as slaves.
- Creates a service, `beta`, with `host2` as master and the other hosts as slaves.
- Creates a service, `gamma`, with `host3` as master and the other hosts as slaves.

The different options set the values and configuration for the system as follows:

Different options set the configuration for the system for different deployment types; click the icon to hide this detail:

Click the icon to show a detailed description of the different options set the configuration for the system for different deployment types:

- [--topology=all-masters \[369\]](#)

Configures the topology type, in this case, `all-masters` indicates that a multi-master topology is required.

- [--install-directory=/opt/continuum \[347\]](#)

Set the installation directory for Tungsten Replication.

- [--replication-user=tungsten \[363\]](#)

Set the user to be used by Tungsten Replication when applying data to a database.

- [--replication-password=secret \[363\]](#)

Set the password to be used by Tungsten Replication when applying data to a database.

- `--master=host1,host2,host3 [351]`

Sets the list of master hosts. As we are configuring a multi-master topology, all three hosts in the cluster are listed as masters.

- `--members=host1,host2,host3 [351]`

Sets the list of member hosts of the dataservice. As we are configuring a multi-master topology, all three hosts in the cluster are listed as members.

- `--master-services=alpha,beta,gamma [351]`

Specifies the list of service names to be used to identify each individual master/slave service.

- `--start [365]`

Indicates that the services should be started once the configuration and installation has been completed.

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

Once `tpm` has completed, the service will be started and the replication will be enabled between hosts.

3.2.3. Management and Monitoring of Multimaster Deployments

To check the configured services use the `services` parameter to `trepctl`:

```
shell> trepctl services
Processing services command...
NAME          VALUE
----          -----
appliedLastSeqno: 44
appliedLatency : 0.692
role           : master
serviceName    : alpha
serviceType   : local
started        : true
state          : ONLINE
NAME          VALUE
----          -----
appliedLastSeqno: 40
appliedLatency : 0.57
role           : slave
serviceName    : beta
serviceType   : remote
started        : true
state          : ONLINE
NAME          VALUE
----          -----
appliedLastSeqno: 41
appliedLatency : 0.06
role           : slave
serviceName    : gamma
serviceType   : remote
started        : true
state          : ONLINE
Finished services command...
```

The output shows the three individual services created in the multimaster configuration, `alpha`, `beta`, and `gamma`, and information about the current latency, status and role of the current host. This gives you an overview of the service state for this host.

To get detailed information about dataservices, each individual dataservice must be checked individually, and explicitly stated on the command-line to `trepctl` as there are now multiple dataservices configured. To check the dataservice status the current host will be displayed, in the example below, `host1`:

```
shell> trepctl -service alpha status
Processing status command...
NAME          VALUE
----          -----
appliedLastEventId   : mysql-bin.000011:0000000000006905;0
appliedLastSeqno     : 44
appliedLatency      : 0.692
channels          : 1
clusterName        : alpha
currentEventId     : mysql-bin.000011:0000000000006905
currentTimeMillis  : 1373891837668
```

```

dataServerHost      : host1
extensions        :
latestEpochNumber : 28
masterConnectUri  : thl://localhost:/
masterListenUri   : thl://host1:2112/
maximumStoredSeqNo: 44
minimumStoredSeqNo: 0
offlineRequests   : NONE
pendingError      : NONE
pendingErrorCode  : NONE
pendingErrorEventId: NONE
pendingErrorSeqno : -1
pendingExceptionMessage: NONE
pipelineSource    : jdbc:mysql:thin://host1:13306/
relativeLatency   : 254295.667
resourcePrecedence: 99
rmiPort           : 10000
role              : master
seqnoType         : java.lang.Long
serviceName       : alpha
serviceType       : local
simpleServiceName: alpha
siteName          : default
sourceId          : host1
state             : ONLINE
timeInStateSeconds: 254530.987
transitioningTo   :
uptimeSeconds     : 254532.724
version           : Tungsten Replicator 5.0.1 build 136
Finished status command...

```

In the above example, the `alpha` dataservice is explicitly requested (a failure to specify a service will return an error, as multiple services are configured).

To get information about a specific host, use the `-host` option. This can be used with the `treptl services` command:

```

shell> treptl -host host3 services
Processing services command...
NAME      VALUE
----      -----
appliedLastSeqno: 44
appliedLatency  : 1.171
role        : slave
serviceName  : alpha
serviceType  : remote
started     : true
state       : ONLINE
NAME      VALUE
----      -----
appliedLastSeqno: 40
appliedLatency  : 1.658
role        : slave
serviceName  : beta
serviceType  : remote
started     : true
state       : ONLINE
NAME      VALUE
----      -----
appliedLastSeqno: 41
appliedLatency  : 0.398
role        : master
serviceName  : gamma
serviceType  : local
started     : true
state       : ONLINE
Finished services command...

```

In the above output, you can see that this host is the master for the dataservice `gamma`, but a slave for the other two services.

Other important fields in this output:

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

Or in combination with the `-service` to get detailed status on a specific host/service combination:

```
shell> treptl -host host3 -service alpha status
```

```

Processing status command...
NAME          VALUE
----          -----
appliedLastEventId : mysql-bin.000011:0000000000006905:0
appliedLastSeqno : 44
appliedLatency   : 1.171
channels        : 1
clusterName     : alpha
currentEventId  : NONE
currentTimeMillis: 1373894128902
dataServerHost   : host3
extensions      :
latestEpochNumber: 28
masterConnectUri: thl://host1:2112/
masterListenUri : thl://host3:2112/
maximumStoredSeqNo: 44
minimumStoredSeqNo: 0
offlineRequests  : NONE
pendingError     : NONE
pendingErrorCode : NONE
pendingErrorEventId: NONE
pendingErrorSeqno: -1
pendingExceptionMessage: NONE
pipelineSource   : thl://host1:2112/
relativeLatency  : 256586.902
resourcePrecedence: 99
rmiPort         : 10000
role            : slave
seqnoType       : java.lang.Long
serviceName      : alpha
serviceType     : remote
simpleServiceName: alpha
siteName        : default
sourceId        : host3
state           : ONLINE
timeInStateSeconds: 256820.611
transitioningTo :
uptimeSeconds    : 256820.779
version         : Tungsten Replicator 5.0.1 build 136
Finished status command...

```

The following sequence number combinations should match between the different hosts on each service:

Master Service	Master Host	Slave Host
alpha	host1	host2,host3
beta	host2	host1,host3
gamma	host3	host1,host2

The sequence numbers on corresponding services should match across all hosts.

For more information on using `trepcctl`, see [Section 9.16, “The trepcctl Command”](#).

Definitions of the individual field descriptions in the above example output can be found in [Section E.2, “Generated Field Reference”](#).

For more information on management and operational detailed for managing your cluster installation, see [Chapter 8, Operations Guide](#).

3.2.4. Alternative Multimaster Deployments

The multimaster deployment can be used for a wide range of different scenarios, and using any number of hosts. The `tpm` command used could, for example, be expanded to four or five hosts by adding them to the list of members and master hosts in the configuration command.

The basis for the multimaster deployment can also be used in multiple site configurations. For more information on multisite/multimaster deployments, see [Deploying Multisite/Multimaster Clusters](#).

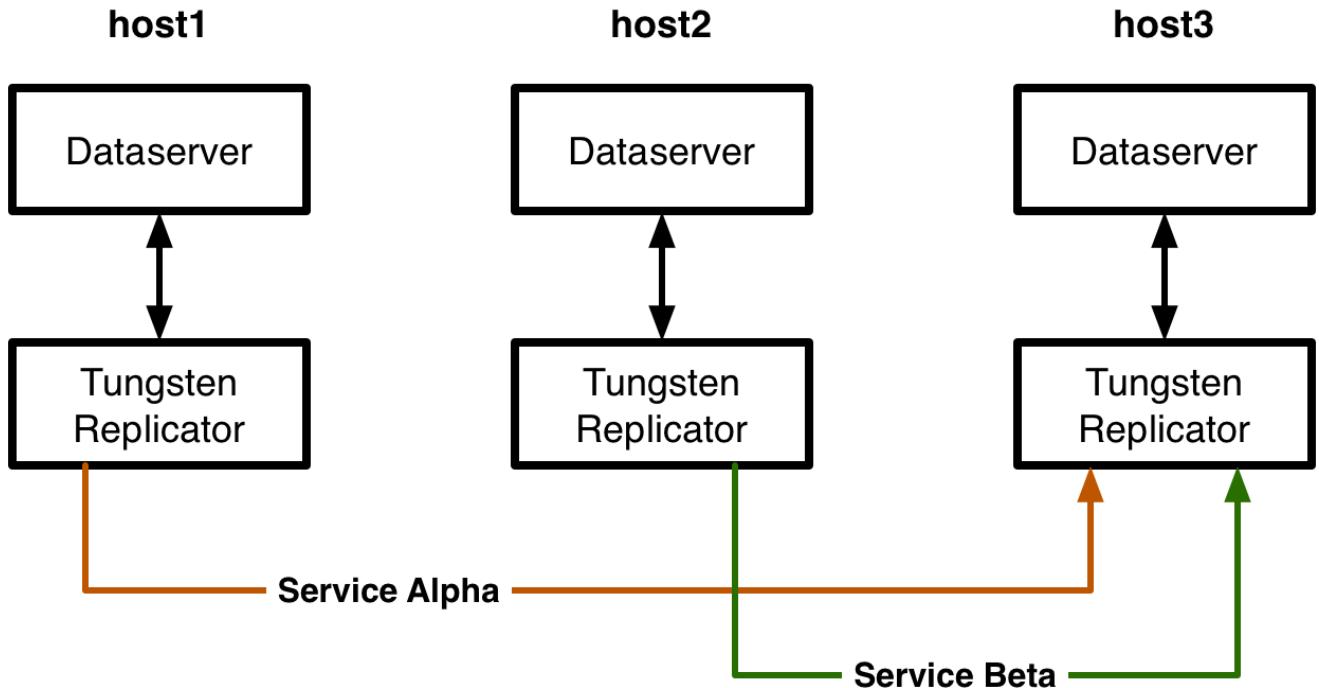
3.3. Deploying a Fan-In Topology

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

Within the fan-in topology:

- host1 is the master replicating to host3
- host2 is the master replicating to host3

Figure 3.3. Topologies: Fan-in



Some additional considerations need to be made when using fan-in topologies:

- If the same tables from each each machine are being merged together, it is possible to get collisions in the data where auto increment is used. The effects can be minimized by using increment offsets within the MySQL configuration:

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

- Fan-in can work more effectively, and be less prone to problems with the corresponding data by configuring specific tables at different sites. For example, with two sites in New York and San Jose databases and tables can be prefixed with the site name, i.e. `sjc_sales` and `nyc_sales`.

Alternatively, a filter can be configured to rename the database `sales` dynamically to the corresponding location based tables. See [Section 11.4.31, “Rename Filter”](#) for more information.

- Statement-based replication will work for most instances, but where your statements are updating data dynamically within the statement, in fan-in the information may get increased according to the name of fan-in masters. Update your configuration file to explicitly use row-based replication by adding the following to your `my.cnf` file:

```
binlog-format = row
```

- Triggers can cause problems during fan-in replication if two different statements from each master are replicated to the slave and cause the operations to be triggered multiple times. Tungsten Replication cannot prevent triggers from executing on the concentrator host and there is no way to selectively disable triggers. Check at the trigger level whether you are executing on a master or slave. For more information, see [Section C.3.1, “Triggers”](#).

To create the configuration the masters and services must be specified, the topology specification takes care of the actual configuration:

```
shell> ./tools/tpm install epsilon \
  --replication-user=tungsten \
  --replication-password=password \
  --install-directory=/opt/continuent \
  --masters=host1,host2 \
  --members=host1,host2,host3 \
  --master-services=alpha,beta \
  --topology=fan-in \
```

`--start`

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

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

- `tpm install`

Executes `tpm` in `install` mode to create the service `alpha`.

- `--replication-user=tungsten [363]`

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

- `--replication-password=password [363]`

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

- `--install-directory=/opt/continuent [347]`

Directory where Tungsten Replication will be installed.

- `--masters=host1,host2 [351]`

In a fan-in topology each master supplies information to the fan-in server.

- `--members=host1,host2,host3 [351]`

List of all the hosts within the cluster, including the master hosts. The fan-in host will be identified as the host not specified as a master.

- `--master-services=alpha,beta [351]`

A list of the services that will be created, one for each master in the fan-in configuration.

- `--topology=fan-in [369]`

Specifies the topology to be used when creating the replication configuration.

- `--start [365]`

Starts the service once installation is complete.

For additional options supported for configuration with `tpm`, see [Chapter 10, The tpm Deployment Command](#).

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

Once the installation has been completed, the service will be started and ready to use.

3.3.1. Management and Monitoring Fan-in Deployments

Once the service has been started, a quick view of the service status can be determined using `treptctl`. Because there are multiple services, the service name and host name must be specified explicitly. The master connection of one of the fan-in hosts:

```
shell> treptctl -service alpha -host host1 status
Processing status command...
NAME          VALUE
-----
appliedLastEventId : mysql-bin.000012:0000000000000418;0
appliedLastSeqno : 0
appliedLatency  : 1.194
channels       : 1
clusterName    : alpha
currentEventId : mysql-bin.000012:0000000000000418
currentTimeMillis : 1375451438898
dataServerHost : host1
extensions     :
latestEpochNumber : 0
masterConnectUri : th1://localhost:/
masterListenUri  : th1://host1:2112/
maximumStoredSeqNo : 0
minimumStoredSeqNo : 0
offlineRequests   : NONE
pendingError      : NONE
```

```

pendingErrorCode      : NONE
pendingErrorEventId   : NONE
pendingErrorSeqno     : -1
pendingExceptionMessage: NONE
pipelineSource        : jdbc:mysql:thin://host1:13306/
relativeLatency       : 6232.897
resourcePrecedence    : 99
rmiPort               : 10000
role                  : master
seqnoType              : java.lang.Long
serviceName             : alpha
serviceType             : local
simpleServiceName      : alpha
siteName               : default
sourceId                : host1
state                  : ONLINE
timeInStateSeconds     : 6231.881
transitioningTo        :
uptimeSeconds           : 6238.061
version                 : Tungsten Replicator 5.0.1 build 136
Finished status command...

```

The corresponding master service from the other host is `beta` on `host2`:

```

shell> treptl -service beta -host host2 status
Processing status command...
NAME          VALUE
-----
appliedLastEventId : mysql-bin.000012:00000000000000415;0
appliedLastSeqno  : 0
appliedLatency   : 0.941
channels        : 1
clusterName      : beta
currentEventId   : mysql-bin.000012:00000000000000415
currentTimeMillis: 1375451493579
dataServerHost    : host2
extensions       :
latestEpochNumber: 0
masterConnectUri  : th1://localhost:/
masterListenUri   : th1://host2:2112/
maximumStoredSeqNo: 0
minimumStoredSeqNo: 0
offlineRequests   : NONE
pendingError      : NONE
pendingErrorCode   : NONE
pendingErrorEventId: NONE
pendingErrorSeqno  : -1
pendingExceptionMessage: NONE
pipelineSource     : jdbc:mysql:thin://host2:13306/
relativeLatency    : 6286.579
resourcePrecedence: 99
rmiPort            : 10000
role               : master
seqnoType          : java.lang.Long
serviceName         : beta
serviceType         : local
simpleServiceName  : beta
siteName            : default
sourceId           : host2
state               : ONLINE
timeInStateSeconds: 6285.823
transitioningTo    :
uptimeSeconds       : 6291.053
version             : Tungsten Replicator 5.0.1 build 136
Finished status command...

```

Note that because this is a fan-in topology, the sequence numbers and applied sequence numbers will be different for each service, as each service is independently storing data within the fan-in hub database.

The following sequence number combinations should match between the different hosts on each service:

Master Service	Master Host	Slave Host
alpha	host1	host3
beta	host1	host3

The sequence numbers between `host1` and `host2` will not match, as they are two independent services.

For more information on using `treptl`, see [Section 9.16, “The treptl Command”](#).

Definitions of the individual field descriptions in the above example output can be found in [Section E.2, “Generated Field Reference”](#).

For more information on management and operational detailed for managing your cluster installation, see [Chapter 8, Operations Guide](#).

3.4. Deploying Multiple Replicators on a Single Host

It is possible to install multiple replicators on the same host. This can be useful, either when building complex topologies with multiple services, and in heterogeneous environments where you are reading from one database and writing to another that may be installed on the same single server.

When installing multiple replicator services on the same host, different values must be set for the following configuration parameters:

3.4.1. Prepare: Multiple Replicators

Before continuing with deployment you will need the following:

1. The name to use for the service.
2. The list of datasources in the service. These are the servers which will be running MySQL.
3. The username and password of the MySQL replication user.

All servers must be prepared with the proper prerequisites. See [Section 2.3, “Prepare Hosts”](#) and [Appendix B, Prerequisites](#) for additional details.

- RMI network port used for communicating with the replicator service.

Set through the `--rmi-port` [363] parameter to `tpm`. Note that RMI ports are configured in pairs; the default port is 10000, port 10001 is used automatically. When specifying an alternative port, the subsequent port must also be available. For example, specifying port 10002 also requires 10003.

- THL network port used for exchanging THL data.

Set through the `--thl-port` parameter to `tpm`. The default THL port is 2112. This option is required for services operating as masters (extractors).

- Master THL port, i.e. the port from which a slave will read THL events from the master

Set through the `--master-thl-port` parameter to `tpm`. When operating as a slave, the explicit THL port should be specified to ensure that you are connecting to the THL port correctly.

- Master hostname

Set through the `--master-thl-host` parameter to `tpm`. This is optional if the master hostname has been configured correctly through the `--master` [351] parameter.

- Installation directory used when the replicator is installed.

Set through the `--install-directory` [347] or `--install-directory` [347] parameters to `tpm`. This directory must have been created, and be configured with suitable permissions before installation starts. For more information, see [Section B.3.3, “Directory Locations and Configuration”](#).

3.4.2. Install: Multiple Replicators

- Staging Configuration — [Section 3.4.2.1, “Deploying Multiple Replicators on a Single Host \(Staging Use Case\)”](#)
- INI Configuration — [Section 3.4.2.2, “Deploying Multiple Replicators on a Single Host \(INI Use Case\)”](#)

3.4.2.1. Deploying Multiple Replicators on a Single Host (Staging Use Case)

For example, to create two services, one that reads from MySQL and another that writes to MongoDB on the same host:

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

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-replicator-5.0.1-136.tar.gz
```

2. Change to the Tungsten Replicator directory:

```
shell> cd tungsten-replicator-5.0.1-136
```

3. Extractor reading from MySQL:

```
shell> ./tools/tpm configure mysql2mongodb \
--install-directory=/opt/extractor \
--java-file-encoding=UTF8 \
--master=host1 \
--members=host1 \
--mysql-enable-enumtostring=true \
--mysql-enable-settostring=true \
--mysql-use-bytes-for-string=false \
--replication-password=password \
--replication-user=tungsten \
--start=true \
--svc-extractor-filters=colnames,pkey
```

This is a standard configuration using the default ports, with the directory `/opt/extractor`.

4. Reset the configuration:

```
shell> ./tools/tpm configure defaults --reset
```

5. Applier for writing to MongoDB:

```
shell> ./tools/tpm configure mysql2mongodb \
--datasource-type=mongodb \
--role=slave \
--install-directory=/opt/applier \
--java-file-encoding=UTF8 \
--master=host1 \
--members=host1 \
--skip-validation-check=InstallerMasterSlaveCheck \
--start=true \
--svc-parallelization-type=none \
--topology=master-slave \
--rmi-port=10002 \
--master-thl-port=2112 \
--master-thl-host=host1 \
--thl-port=2113
```

In this configuration, the master THL port is specified explicitly, along with the THL port used by this replicator, the RMI port used for administration, and the installation directory `/opt/applier`.

When multiple replicators have been installed, checking the replicator status through `trepctl` depends on the replicator executable location used. If `/opt/extractor/tungsten/tungsten-replicator/bin/trepctl`, the extractor service status will be reported. If `/opt/applier/tungsten/tungsten-replicator/bin/trepctl` is used, then the applier service status will be reported.

Alternatively, a specific replicator can be checked by explicitly specifying the RMI port of the service. For example, to check the extractor service:

```
shell> trepctl -port 10000 status
```

Or to check the applier service:

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

When an explicit port has been specified in this way, the executable used is irrelevant. Any valid `trepctl` instance will work.

Further, either path may be used to get a summary view using `multi_trepctl`:

```
shell> /opt/extractor/tungsten/tungsten-replicator/scripts/multi_trepctl
| host    | servicename | role   | state  | appliedlastseqno | appliedlatency |
| host1  | extractor   | master | ONLINE |          0 |      1.724 |
| host1  | applier     | slave  | ONLINE |          0 |      0.000 |
```

3.4.2.2. Deploying Multiple Replicators on a Single Host (INI Use Case)

It is possible to install multiple replicators on the same host. This can be useful, either when building complex topologies with multiple services, and in heterogeneous environments where you are reading from one database and writing to another that may be installed on the same single server.

When installing multiple replicator services on the same host, different values must be set for the following configuration parameters:

- RMI network port used for communicating with the replicator service.

Set through the `rmi-port` [363] parameter to `tpm`. Note that RMI ports are configured in pairs; the default port is 10000, port 10001 is used automatically. When specifying an alternative port, the subsequent port must also be available. For example, specifying port 10002 also requires 10003.

- THL network port used for exchanging THL data.

Set through the `thl-port` parameter to `tpm`. The default THL port is 2112. This option is required for services operating as masters (extractors).

- Master THL port, i.e. the port from which a slave will read THL events from the master

Set through the `master-thl-port` parameter to `tpm`. When operating as a slave, the explicit THL port should be specified to ensure that you are connecting to the THL port correctly.

- Master hostname

Set through the `master-thl-host` parameter to `tpm`. This is optional if the master hostname has been configured correctly through the `master` [351] parameter.

- Installation directory used when the replicator is installed.

Set through the `install-directory` [347] or `install-directory` [347] parameters to `tpm`. This directory must have been created, and be configured with suitable permissions before installation starts. For more information, see [Section B.3.3, “Directory Locations and Configuration”](#).

For example, to create two services, one that reads from MySQL and another that writes to MongoDB on the same host:

1. Install the Tungsten Replicator™ package (`.rpm`), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-replicator-5.0.1-136.tar.gz
```

2. Change to the Tungsten Replicator directory:

```
shell> cd tungsten-replicator-5.0.1-136
```

3. Create the proper directories with appropriate ownership and permissions:

```
shell> sudo mkdir /opt/applier /opt/extractor
shell> sudo chown tungsten: /opt/applier/ /opt/extractor/
shell> sudo chmod 700 /opt/applier/ /opt/extractor/
```

4. Create `/etc/tungsten/tungsten-extractor.ini` with the following configuration:

```
[mysql2mongodb]
install-directory=/opt/extractor
master=host1
members=host1
mysql-enable-enumtostring=true
mysql-enable-settostring=true
mysql-use-bytes-for-string=false
replication-password=password
replication-user=tungsten
svc-extractor-filters=colnames,pkey
java-file-encoding=UTF8
start-and-report=true
```

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.

- Service `[mysql2mongodb]` —defines the Extractor process reading from MySQL

This is a standard configuration using the default ports, with the directory `/opt/extractor`.

- `start-and-report=true` [365]

Starts the service once installation is complete.

5. Create `/etc/tungsten/tungsten-applier.ini` with the following configuration:

```
[mysql2mongodb]
```

```

install-directory=/opt/applier
topology=master-slave
role=slave
datasource-type=mongodb
master=host1
members=host1
skip-validation-check=InstallerMasterSlaveCheck
svc-parallelization-type=none
master-thl-host=host1
master-thl-port=2112
thl-port=2113
rmi-port=10002
java-file-encoding=UTF8
start-and-report=true

```

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.

- Service [\[mysql2mongodb\]](#) —defines the Applier process for writing to MongoDB

In this configuration, the master THL port is specified explicitly, along with the THL port used by this replicator, the RMI port used for administration, and the installation directory [/opt/applier](#).

- [start-and-report=true \[365\]](#)

Starts the service once installation is complete.

6. Run [tpm](#) to install the software with the INI-based 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 [start-and-report \[365\]](#) is set and the service starts correctly, you should see the configuration and current status of the service.

7. Initialize your [PATH](#) and environment.

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

8. Check the replication status.

When multiple replicators have been installed, checking the replicator status through [treptl](#) depends on the replicator executable location used. If [/opt/extractor/tungsten/tungsten-replicator/bin/treptl](#), the extractor service status will be reported. If [/opt/applier/tungsten/tungsten-replicator/bin/treptl](#) is used, then the applier service status will be reported.

Alternatively, a specific replicator can be checked by explicitly specifying the RMI port of the service. For example, to check the extractor service:

```
shell> treptl -port 10000 status
```

Or to check the applier service:

```
shell> treptl -port 10002 status
```

When an explicit port has been specified in this way, the executable used is irrelevant. Any valid [treptl](#) instance will work.

Further, either path may be used to get a summary view using [multi_treptl](#):

```
shell> /opt/extractor/tungsten/tungsten-replicator/scripts/multi_treptl
| host   | servicename | role    | state   | appliedlastseqno | appliedlatency |
| host1  | extractor   | master  | ONLINE  | 0                | 1.724          |
| host1  | applier     | slave   | ONLINE  | 0                | 0.000          |
```

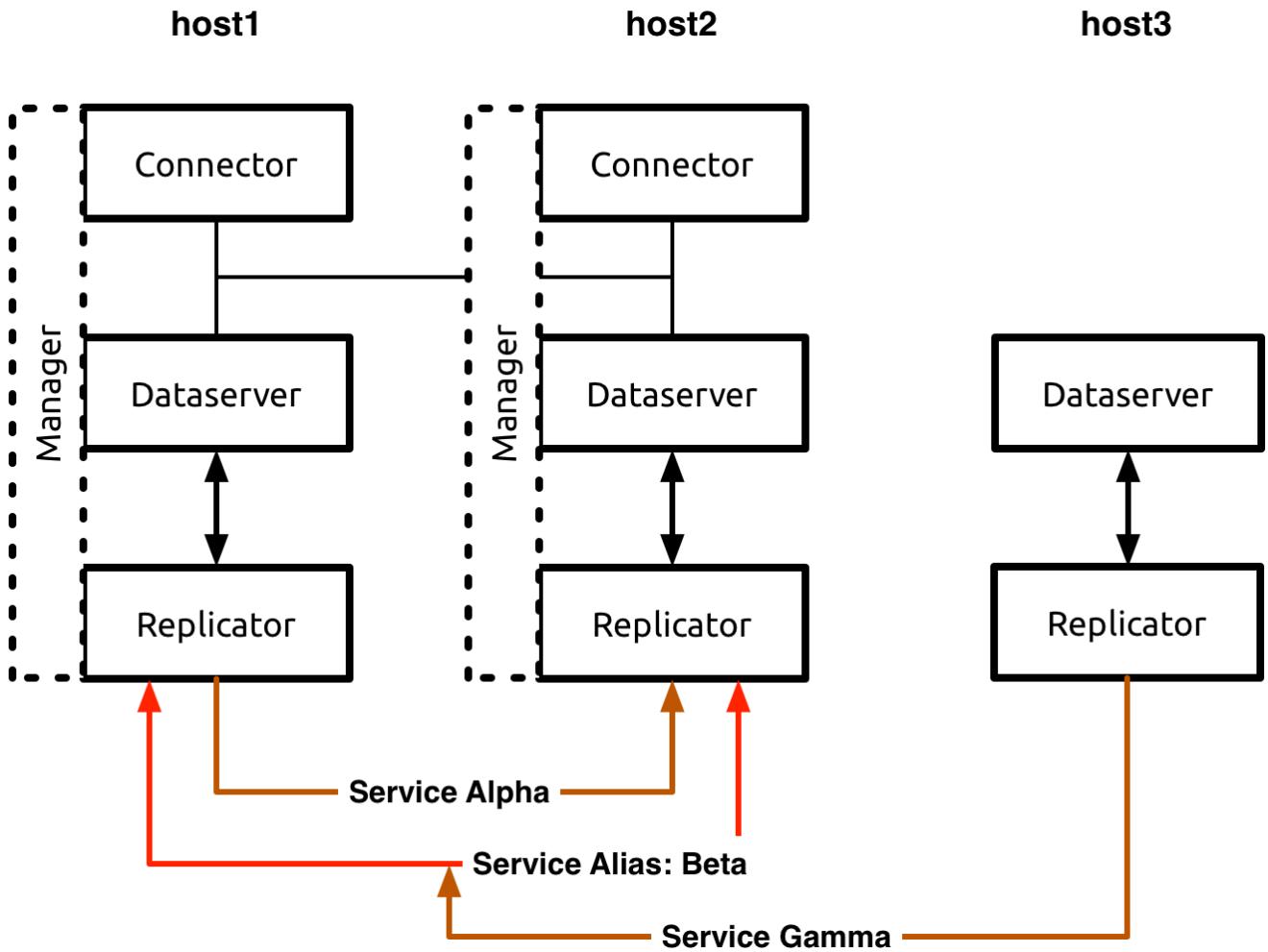
3.4.3. Best Practices: Multiple Replicators

Follow the guidelines in Section 2.2, “Best Practices”.

3.5. 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 3.4. Topologies: Replicating Data Out of a Cluster



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

The Tungsten Replicator will be installed to every server in the cluster-slave. 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.

3.5.1. 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` [344] 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.

3.5.2. Deploy: Replicating Data Out of a Cluster

- Staging Configuration — [Section 3.5.2.1, “Replicating from a Cluster to MySQL \(Staging Use Case\)”](#)
- INI Configuration — [Section 3.5.2.2, “Replicating from a Cluster to MySQL \(INI Use Case\)”](#)

3.5.2.1. Replicating from a Cluster to MySQL (Staging Use Case)

1. On your staging server, go to the software directory.

```
shell> cd /opt/continuent/software
```

2. Download Tungsten Replicator 2.2.0 or later.

3. Unpack the release package

```
shell> tar zxf tungsten-replicator-5.0.1-136.tar.gz
```

4. Change to the unpackaged directory:

```
shell> cd tungsten-replicator-5.0.1-136
```

5. Execute the `tpm` command to configure defaults for the installation.

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

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

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

- **`tpm configure defaults`**

This runs the `tpm` command. `configure defaults` indicates that we are setting options which will apply to all dataservices.

- **`--install-directory=/opt/continuent` [347]**

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

- **`--profile-script=~/.bashrc` [361]**

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 Replication™ are easier to use.

- **`--user=tungsten` [370]**

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

- **`--replication-user=tungsten` [363]**

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

- **`--replication-password=password` [363]**

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

- **`--replication-port=13306` [363]**

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

- **`--start-and-report` [365]**

Tells `tpm` to startup the service, and report the current configuration and status.

Important

If you are replicating to a non-MySQL server. Include the `--enable-heterogeneous-service=true` [344] option in the above command.

- Configure a cluster alias that points to the masters and slaves within the current Tungsten Replication 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` [369] has been specified. This alias is used in the cluster-slave section to define the source hosts for replication.

- [--master=host1](#) [351]

Specifies the hostname of the default master in the cluster.

- [--slaves=host2,host3](#) [365]

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](#) [369]

Define this as a cluster dataservice alias so `tpm` does not try to install software to the hosts.

Important

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

- Create the configuration that will replicate from cluster dataservice `alpha` into the database on the host specified by `--master=host6` [351]:

```
shell> ./tools/tpm configure omega \
--master=host6 \
--relay-source=alpha \
--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.

- [--master=host6](#) [351]

Specifies the hostname of the destination database into which data will be replicated.

- [--relay-source=alpha](#) [362]

Specifies the name of the source cluster dataservice alias (defined above) that will be used to read events to be replicated.

- [--topology=cluster-slave](#) [369]

Read source replication data from any host in the `alpha` dataservice.

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

```
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 should be installed and ready to use.

3.5.2.2. Replicating from a Cluster to MySQL (INI Use Case)

1. Create the configuration file `/etc/tungsten/tungsten.ini` on the destination DBMS host:

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

[alpha]
topology=cluster-alias
master=host1
members=host1,host2,host3
th1-port=2112

[omega]
topology=cluster-slave
master=host6
relay-source=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.

- **[defaults]**

`defaults` indicates that we are setting options which will apply to all cluster dataservices.

- **user=tungsten [370]**

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

- **install-directory=/opt/continuent [347]**

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 [363]**

The MySQL user name to use when connecting to the MySQL database.

- **replication-password=secret [363]**

The MySQL password for the user that will connect to the MySQL database.

- **replication-port=3306 [363]**

The TCP/IP port on the destination DBMS server that is listening for connections.

- **start-and-report=true [365]**

Tells `tpm` to startup the service, and report the current configuration and status.

- **profile-script=~/.bashrc [361]**

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 [369]` has been specified. This alias is used in the cluster-slave section to define the source hosts for replication.

- **topology=cluster-alias [369]**

Tells `tpm` this is a cluster dataservice alias.

- `members=host1,host2,host3` [351]

A comma separated list of all the hosts that are part of this cluster dataservice.

- `master=host1` [351]

The hostname of the server that is the current cluster master MySQL server.

- `th1-port=2112`

The TH1 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 `master=host6` [351].

- `topology=cluster-slave` [369]

Tells `tpm` this is a cluster-slave replication service which will have a list of all source cluster nodes available.

- `master=host6` [351]

The hostname of the destination DBMS server.

- `relay-source=alpha` [362]

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` [365] if you are taking over for MySQL native replication. See [Section 8.9.1, “Migrating from MySQL Native Replication ‘In-Place’”](#) for next steps after completing installation.

2. Download and install the Tungsten Replicator 2.2.0 or later package (`.rpm`), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software  
shell> tar zxf tungsten-replicator-5.0.1-136.tar.gz
```

3. Change to the Tungsten Replicator staging directory:

```
shell> cd tungsten-replicator-5.0.1-136
```

4. 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 replicator should be installed and ready to use.

3.5.3. Best Practices: Replicating Data Out of a Cluster

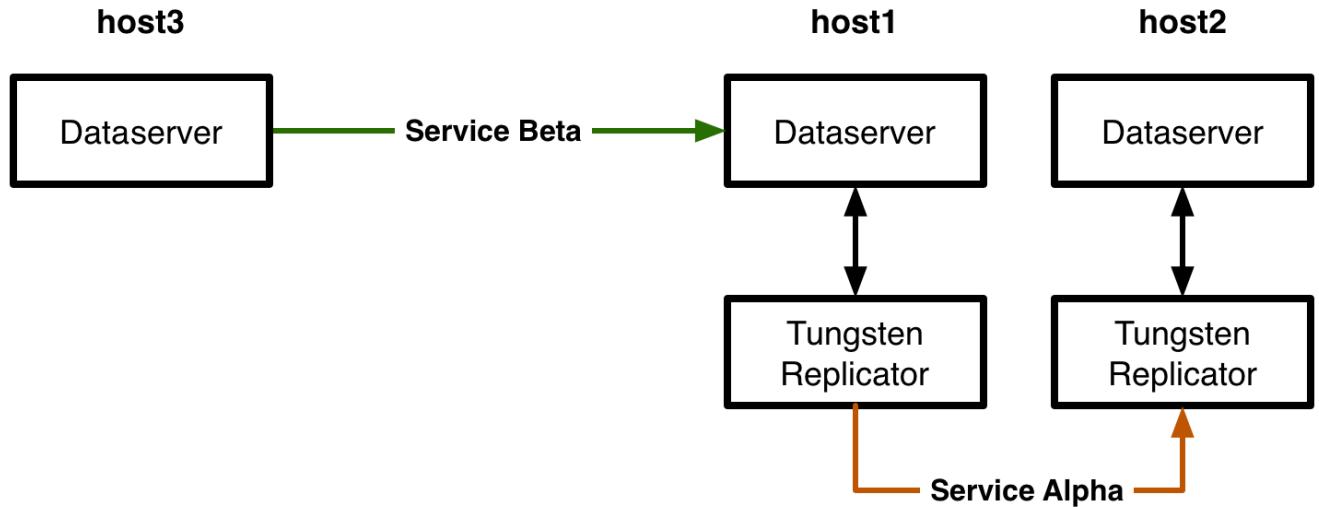
- Setup proper monitoring for all servers in the cluster-slave as described in [Section 8.16, “Monitoring Tungsten Replication”](#).

3.6. Replicating Data Into an Existing Dataservice

If you have an existing dataservice, data can be replicated from a standalone MySQL server into the service. The replication is configured by creating a service that reads from the standalone MySQL server and writes into the master of the target dataservice. By writing this way, changes are replicated to the master and slave in the new deployment.

Additionally, using a replicator that writes data into an existing data service can be used when migrating from an existing service into a new Tungsten Replication service. For more information on initially provisioning the data for this type of operation, see [Section 8.9.2, “Migrating from MySQL Native Replication Using a New Service”](#).

Figure 3.5. Topologies: Replicating into a Dataservice



In order to configure this deployment, there are two steps:

1. Create a new replicator on an existing server that replicates into a master of the destination dataservice
2. Create a new replicator that reads the binary logs directly from the external MySQL service through the master of the destination dataservice

There are also the following requirements:

- The host on which you want to replicate to must have Tungsten Replicator 2.2.0 or later.
- Hosts on both the replicator and cluster must be able to communicate with each other.
- The replication user on the source host must have the `RELOAD`, `REPLICATION SLAVE`, and `REPLICATION CLIENT GRANT` privileges.
- Replicator must be able to connect as the `tungsten` user to the databases within the cluster.

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

```

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

```

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

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

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

- [tpm configure](#)

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

- [--install-directory=/opt/continuent \[347\]](#)

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

- [--rmi-port=10002 \[363\]](#)

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

- `--user=tungsten [370]`

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

- `--replication-user=tungsten [363]`

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

- `--replication-password=password [363]`

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

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

```
shell> ./tools/tpm configure beta \
    --topology=direct \
    --master-host1 \
    --direct-datasource-host=host3 \
    --thl-port=2113
```

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 \
    --segno=0 --epoch=0 --service=beta \
    --source-id=host3 --event-id=mysql-bin.000026:1311
```

Now start the replicator:

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

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

```
shell> /opt/replicator/tungsten/tungsten-replicator/bin/trepctl status
Processing status command...
NAME          VALUE
----          -----
appliedLastEventId : mysql-bin.000026:0000000000001311:1252
appliedLastSeqno : 5
appliedLatency  : 0.748
channels       : 1
clusterName    : beta
currentEventId : mysql-bin.000026:0000000000001311
currentTimeMillis : 1390410611881
dataServerHost : host1
extensions     :
host           : host3
latestEpochNumber : 1
masterConnectUri : thl://host3:2112/
masterListenUri  : thl://host1:2113/
maximumStoredSeqNo : 5
minimumStoredSeqNo : 0
offlineRequests : NONE
pendingError    : NONE
pendingErrorCode : NONE
```

```
pendingErrorEventId      : NONE
pendingErrorSeqno        : -1
pendingExceptionMessage  : NONE
pipelineSource           : jdbc:mysql:thin://host3:13306/
relativeLatency          : 8408.881
resourcePrecedence       : 99
rmiPort                  : 10000
role                     : master
seqnoType                : java.lang.Long
serviceName               : beta
serviceType               : local
simpleServiceName         : beta
siteName                 : default
sourceId                 : host3
state                    : ONLINE
timeInStateSeconds       : 8408.21
transitioningTo          :
uptimeSeconds             : 8409.88
useSSLConnection          : false
version                  : Tungsten Replicator 5.0.1 build 136
Finished status command...
```

Chapter 4. Heterogeneous Deployments

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

- MySQL to Oracle, Oracle to MySQL and Oracle to Oracle, using either the [Redo Reader](#) method.
- [MySQL to Oracle, Oracle to MySQL and Oracle to Oracle using Oracle CDC](#) method.
- [MySQL or Oracle to Hadoop](#)
- [MySQL or Oracle to Amazon Redshift](#)
- [MySQL to Vertica](#)

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

4.1. How Heterogeneous Replication Works

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

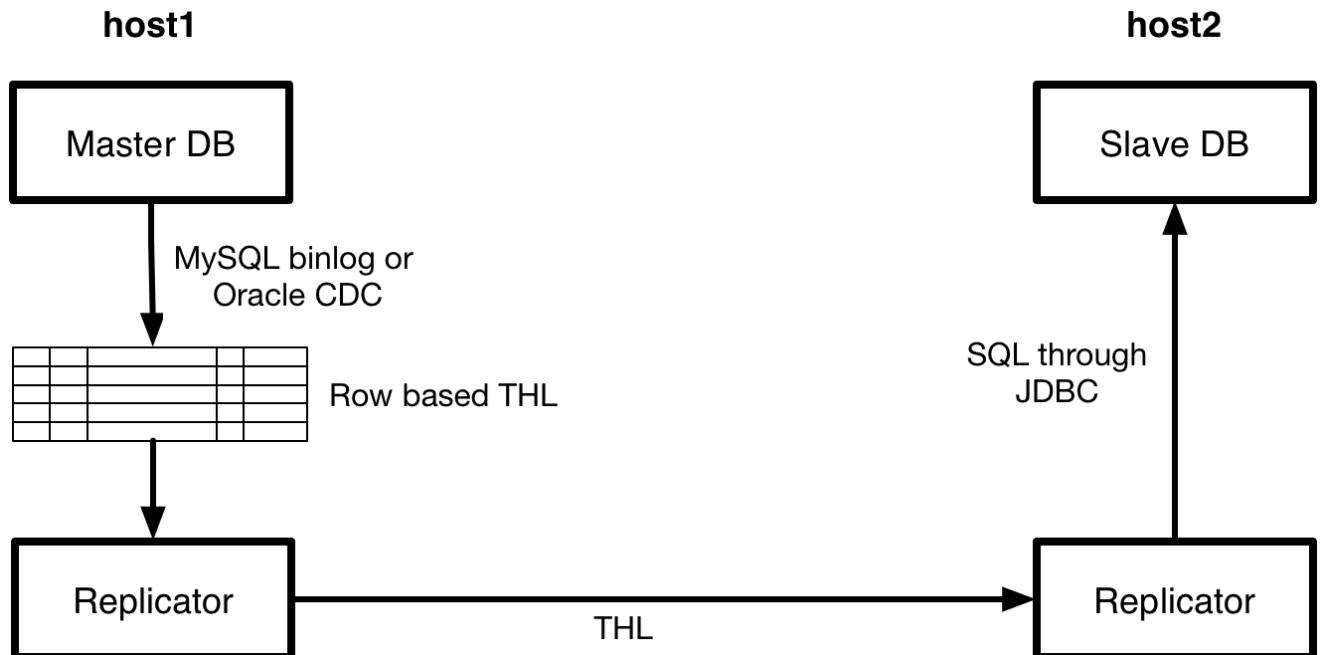
On targets that do not support SQL of any kind, such as MongoDB, replicating SQL statements would achieve nothing since they cannot be executed at all.

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

MySQL to Oracle, Oracle to MySQL, and Oracle to Oracle Replication

Replication between Oracle or MySQL, in either direction, or Oracle-to-Oracle replication, work as shown in Figure 4.1, “Topologies: Heterogeneous Operation”.

Figure 4.1. Topologies: Heterogeneous Operation



The process works as follows:

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

- **For MySQL:**

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

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

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

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

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

- **For Oracle CDC:**

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

For Oracle Redo:

The Oracle redo extractor works in a similar fashion to the MySQL binary logging extractor. A separate component, the redo log reader, reads transactions directly from the Oracle redo log and supplemental logs to construct the transaction information. This transaction data is then sent to the replicator and recorded into the THL format. The redo reader affords some performance advantages over the CDC method, in particular because the extraction of the data from Oracle is off-boarded from the database, there is no additional load on the Oracle database itself when reading the logs. Furthermore, it also some greater flexibility in terms of the deployment and supported operating system and environment. The redo reader component can be installed on a separate host than the Tungsten Replicator, and is supported on platforms not directly supported by Tungsten Replicator.

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

2. The row-based transactions stored in the THL are transferred from the master to the slave.
3. On the slave (or applier) side, the row-based event data is wrapped into a suitable SQL statement for the target database environment. Because the raw row data is available, it can be constructed into any suitable statement appropriate for the target database.

Native Applier Replication (e.g. MongoDB)

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

Batch Loading

For batch appliers, such as Vertica, the row-data is converted into CSV files in batches. The format of the CSV file includes both the original row data for all the columns of each table, and metadata on each line that contain the unique [SEQNO](#) [459] and the operation type (insert, delete or update). A modified form of the CSV is used in some cases where the operation type is only an insert or delete, with updates being translated into a delete followed by an insert of the updated information.

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

Depending on the batch environment, the loading of the data into the final destination tables is performed either within the same script, or by using a separate script. Both methods work in the same basic fashion; the base table is updated using the data from the staging table, with each row marked to be deleted, deleted, and the latest row (calculated from the highest [SEQNO](#) [459]) for each primary key) are then inserted

Schema Creation and Replication

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

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

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

For MongoDB, where no explicitly DDL needs to be generated, the use of [ddlsan](#) is not required.

Chapter 5. Heterogeneous Oracle Deployments

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

- MySQL to Oracle, Oracle to MySQL and Oracle to Oracle, using the [Redo Reader](#).
- MySQL to Oracle, Oracle to MySQL and Oracle to Oracle, using the [Oracle CDC](#).

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

5.1. Oracle Replication using Redo Reader

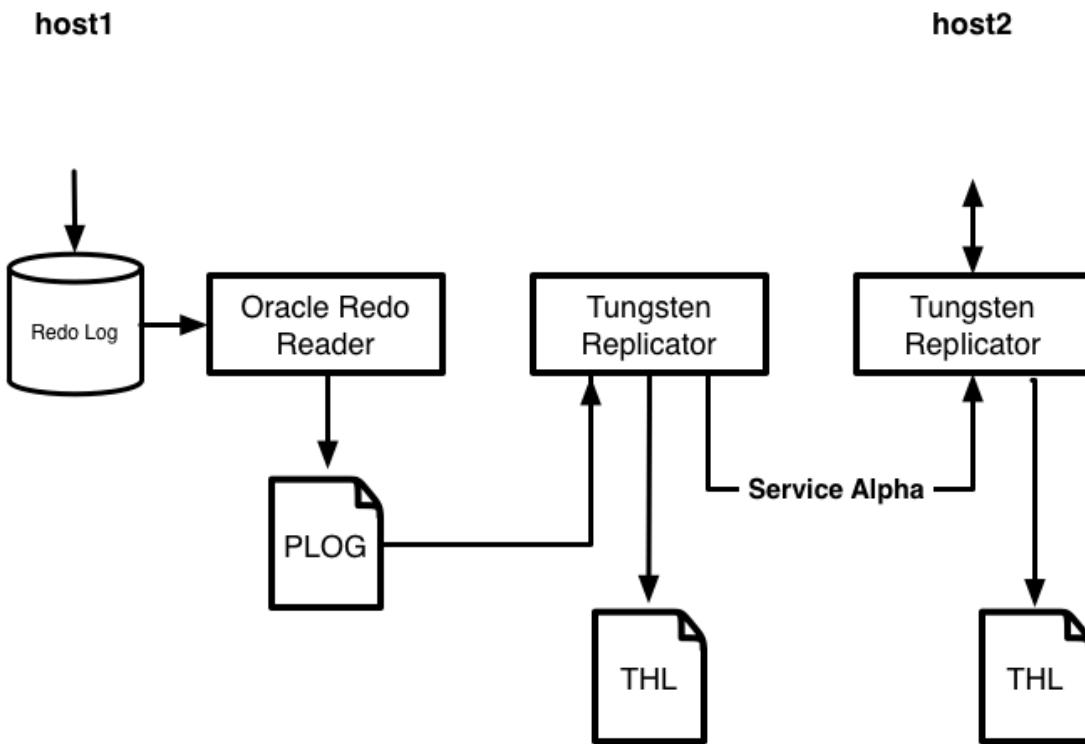
Replication Operation Support	
Statements Replicated	No
Rows Replicated	Yes
Schema Replicated	Yes, with limitations
<code>ddlscan</code> Supported	Yes, supported for mixed Oracle/MySQL, and data warehouse targets

To extract information from an Oracle database two options are available, the CDC process (for more information, see [Section 5.7, “Deploying Oracle Replication using CDC”](#)), and a system that reads information directly from the Oracle redo log. This includes another process that runs alongside the Replicator and extracts events from the Oracle Redo Log. The functionality listed here is only available in the release named vmware-continuent-replication-oracle-source. Contact your sales representative if you don't have access to that build.

Replication with the Oracle Read Reader operates as follows:

- A separate VMware Redo Reader process accesses and parses the content from the Oracle redo and supplemental logs, generating the individual transaction information into a format called the PLOG. The information in the PLOG contains each individual transaction, including limited DDL statements, in a format that can be processed by the Tungsten Replicator. This component is configured and installed automatically during the standard Tungsten Replicator deployment process.
- The Tungsten Replicator reads the information from the PLOG and converts it into the Transaction History Log (THL) used by the replicator to store events and transactions. Once the transactions have been converted into the internal THL format, the information can be used by any downstream replicator to apply the data into another Oracle instance, or heterogeneous target such as MySQL or data warehouses.

Figure 5.1. Topologies: Oracle to Oracle with Redo Reader



Tungsten Replicator with Redo Log processing is supported within the following environments:

- Oracle 9.2 (Experimental), 10g, 11g, 12c
- Enterprise Edition (EE), Standard Edition (SE), Standard One (SE1) and Express Edition (XE)
- Oracle RAC (Experimental)

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

- Tables must have primary keys
- Transparent data encryption is not supported
- Compressed data is not supported
- The following datatypes are not supported:
 - `XMLType`
 - `Geometric type`
 - `BINARY FLOAT`
 - `BINARY DOUBLE`
 - `ANYTYPE`
 - `ANYDATA`

5.1.1. Oracle Redo Reader Replication Operation

Tungsten Replicator Replication using the Oracle Redo Reader extracts data from Oracle using a specific sequence of operations, outlined as followed:

- Oracle Read Reader (`vnmr`) reads the Oracle redo logs directly, extracting information about individual transactions and storing them within a series of files called PLOGS.

5.1.2. Preparing the Oracle Environment for Replication

There are a few steps that you must take before the replicator is installed, including configuring the Oracle database and environment.

Ensure you have followed the general notes within the [Section 2.3, "Prepare Hosts"](#). For supported platforms, and environments, see [Section B.1, "Requirements"](#).

For your reference, below is the syntax for connecting to Oracle with a full username and password. Substitute your own values for all items in capital letters.

```
shell> sqlplus "USER/PASSWORD@HOSTNAME:1521/SERVICE"
```

5.1.2.1. Prepare the Oracle Driver

You must have the Oracle JDBC driver available. Verify that it exists in the Oracle software tree.

The file will be named `ojdbc6.jar` or `ojdbc7.jar`, located in the `$ORACLE_HOME/jdbc/lib/` directory.

If it is not there, you can copy either version of the driver jar file into the appropriate extracted Continuent software staging directory (`/vmware-continuent-replication-oracle-source-5.0.1-136/tungsten-replicator/lib/`) prior to installation, and the jar file will be copied along with the rest of the Continuent release files.

5.1.2.2. Prepare the Oracle System User

If using a system account for replication other than oracle (e.g., `tungsten`), ensure that the following are met:

- Account belongs to `oinstall` group so it can read Oracle files.
- Account has Oracle environment set correctly. This can be configured by ensuring that the correct system variables are enabled within your `.profile`, `.bash_profile`, or `.bashrc`:

```
export ORACLE_BASE=/app/oracle/local
export ORACLE_HOME=/app/oracle/local/product/11/db_1
export PATH=$PATH:$ORACLE_HOME/bin
export ORACLE_SID=ORCL
```

The values above are examples. The environment variables and assigned values should only be set if they are not specified yet. Please ensure that `ORACLE_HOME` is set to the location where Oracle is installed and that `ORACLE_BASE` is set to the base directory of the OFA installation. `ORACLE_SID` can have any value which identifies this particular database in the system.

5.1.2.3. Prepare the Oracle DBMS Servers

All of the Oracle servers need to be prepared with the following procedure. There are optional steps at the end, based on the role of the server.

If it does not already exist, create a tablespace that contains all the data, structures and components that will be replicated. This value will be used with the `--oracle-redo-tablespace` [357] argument and the alter user statement in the next step.

```
SQL> create tablespace tungsten_ts datafile '/app/oracle/data/tungsten_ts.dbf'
      size 1000M extent management local autoallocate segment space management auto;
```

If you get an error like ORA-01109: database not open, try using the `ALTER` command to prepare the database.

```
SQL> ALTER DATABASE OPEN;
```

If you get an error reporting that archive loggin has not been enabled, you should enable archive logging and supplemental logging. Doing so requires shutting down the Oracle database temporarily to enable logging mode.

```
sqlplus sys/oracle as sysdba
SQL> shutdown immediate;
SQL> startup mount;
SQL> alter database archivelog;
SQL> alter database add supplemental log data;
```

```
SQL> alter database open;
```

You can confirm the status of archive logging:

Checking the status again should show the archive log enabled:

```
sqlplus sys/oracle as sysdba
SQL> archive log list;
SQL> archive log list;
```

Create a replication user. The username and password will need to be used with the `--replication-user` [363] and `--replication-password` [363] arguments.

Important

For Heterogeneous Deployments like Oracle to Vertica or Oracle to Hadoop, please note the following critical step when creating the replication user in Oracle (and the replication service name):

The tracking schema in Oracle is created with the replication user name. This differs from other datasources where the schema of the tracking tables are created with the pattern `tungsten_{ServiceNameHere}`.

To make this work, replication user should be called `tungsten_ServiceNameHere`, where `ServiceNameHere` is the name of the replication service.

For example, if the desired replication service name is "oracle2vertica", then create an Oracle replication user called "`tungsten_oracle2vertica`", which automatically creates a tracking schema named `tungsten_oracle2vertica`.

```
SQL> create user tungsten identified by "secret" default tablespace tungsten_ts;
SQL> grant connect to tungsten;
SQL> alter user tungsten quota unlimited on tungsten_ts;
SQL> grant dba to tungsten;
```

For all servers except Migration Secondary targets, perform the following:

```
SQL> grant execute on dbms_flashback to tungsten;
```

The above GRANT is required for the VMware Redo Reader to extract properly. If the server ever becomes a Primary, this GRANT will be needed.

5.1.2.4. Prerequisite Checker Script

The `pre-req-checker.sql` script will install a PL/SQL Package in the DB instance, which you can then run to check for any unsupported features, data types etc. The SQL package should be installed and executed as follows:

1. Copy the `tungsten-replicator/support/oracle-pre-req-checker/pre-req-checker.sql` script to the Oracle host machine.
2. Login to `sqlplus` as `sys`:

```
SQL> sqlplus / as sysdba
```

or

```
SQL> sqlplus sys as sysdba
```

3. Install the package as follow:

```
SQL> @pre-req-checker.sql
```

The above package install step is one time only - once that package has been installed, you can run it as many times as you require.

To Execute the package and run the checks, type the following:

```
SQL> set serveroutput on
SQL> exec ContinuentPreReqCheck.run(target,schema,user,path);
```

Substituting variables as follows:

- `target` — The flavour of DB that you are replicating to, one of 'ORACLE','MYSQL55','MYSQL56'
- `schema` — A comma separated list of schemas that you will be replicating
- `user` — The replicator DB user on the local source instance (Typically `tungsten`)
- `path` — The local OS path for the results of the checker where the report will be logged

After executing, you will see a message that output has been written to a log file, the name and path will be given, and will look something similar to the following:

```
SQL> set serveroutput on
SQL> exec ContinuentPreReqCheck.run('ORACLE','LAB','TUNGSTEN','/home/oracle/log');
>> Checks complete, results in /home/oracle/log/prereq-201511020730.log
PL/SQL procedure successfully completed.
SQL>
```

You can then review the log file for the results, from the above example the log looks like the following:

```
shell> cat /home/oracle/log/prereq-201511020730.log
INFO >>Major Version : 11 Minor Version : 2
PASS >>Source Version Check [1]
ERROR>>Table [LAB.DEMOTABLE] : Column [COL_ERR] is of unsupported datatype [ROWID]
FAIL >>Data Type Check [0]
ERROR>>Table [LAB.DEMOTABLE] does not have a Primary Key
ERROR>>Table [LAB.LABTAB1] does not have a Primary Key
FAIL >>Primary Key Check [0]
PASS >>Unsupported Features Check [1]
INFO >>User [TUNGSTEN] exists
INFO >>[TUNGSTEN] user has DBA role
PASS >>User Check [1]
```

5.1.3. Installing an Oracle to Oracle Deployment

There are different techniques for installing your Oracle deployment, depending on how you want to execute the installation:

- [Section 5.2, "Deploying Oracle Replication for Disaster Recovery"](#) — Installs an Oracle to Oracle Disaster Recovery deployment. This can be accomplished as a single installation requiring the inclusion of Oracle system passwords in the `tpm` command, or a multi-step procedure which does not.
- [Section 5.3, "Deploying Oracle Replication for Migration"](#) — Installs an Oracle to Oracle Migration deployment. The extractor on the secondary/target will be disabled, since the secondary will never take on the primary role. Just like the DR deployment, this can be accomplished as a single installation requiring the inclusion of Oracle system passwords in the `tpm` command, or a multi-step procedure which does not.
- [Section 5.4, "Deploying Oracle to MySQL Replication"](#) — Installs an Oracle to MySQL deployment. This can be accomplished as a single installation requiring the inclusion of Oracle system passwords in the `tpm` command, or a multi-step procedure which does not.
- [Section 5.1.3.1, "Installing a Standalone Oracle Extractor for Heterogeneous Deployments"](#) — Installs an Oracle extractor only, which can be used in heterogeneous, data warehouse, and more complex deployments. This step requires including Oracle system passwords in the `tpm` installation.

5.1.3.1. Installing a Standalone Oracle Extractor for Heterogeneous Deployments

When replicating information from Oracle for the purposes of heterogeneous deployments, including replicating to MySQL, HP Vertica, or other data warehouse targets, a standalone Oracle extractor can be installed. This will extract information from Oracle and generate THL, but not apply the data to a target. The target applier can also be configured separately.

- Validate the Replicator and VMware Redo Reader
- Configure the Replicator
- Install the Replicator

To configure and install the replicator, follow these steps:

1. Unpack the software

```
shell> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
shell> cd vmware-continuent-replication-oracle-source-5.0.1-136
```

2. Locate the Oracle JDBC driver, the `ojdbc.jar` should either be located in your `vmware-continuent-replication-oracle-source-5.0.1-136/lib` directory within your release, or within your `$ORACLE_HOME` directory.
3. Configure the staging directory with the desired configuration, starting with the default configuration data:

```
shell> ./tools/tpm configure defaults \
--install-directory=/opt/continuent \
--replication-password=secret \
--replication-user=tungsten \
--start-and-report=true \
--user=tungsten
```

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

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

- [tpm install](#)

Executes `tpm` in `configure` mode, configuring the default settings for all parts of the service, and also resetting any existing configuration (using the `--reset` [363]) option.

- [--replication-user=tungsten](#) [363]

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

- [--user=tungsten](#) [370]

The operating system user that the service will be executed using.

- [--install-directory=/opt/continuent](#) [347]

Directory where Tungsten Replicator will be installed.

- [--replication-password=password](#) [363]

The password that will be used for the replication service when connecting to the datasource.

- [--start-and-report=true](#) [365]

Once the service has been installed, start it automatically, and the report on the status.

4. Now configured the replicator specifics:

```
shell> ./tools/tpm configure alpha \
--datasource-oracle-service=ORCL \
--datasource-type=oracle \
--enable-heterogeneous-service=true \
--install-vmware-redo-reader=true \
--master=host1 \
--members=host1 \
--oracle-extractor-method=redo \
--oracle-redo-replicate-tables=TUNGSTEN,LAB_DEMO \
--oracle-redo-tablespace=TUNGSTEN_TS \
--oracle-sys-user-password=password \
--oracle-system-user-password=password
```

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](#) [351]

Specifies which host will be the master.

- [--members=host1](#) [351]

The members of the service, in this case, the master (`host1`) only.

- [--datasource-type=oracle](#) [340]

Specifies the source database type, in this case, Oracle.

- [--datasource-oracle-service=ORCL](#) [338]

The Oracle service name. This must match the name of the Oracle service as defined in `oratab`.

- [--install-vmware-redo-reader=true](#) [348]

Enables automatic installation of the VMware Redo Reader to extract information from the Oracle Redo logs.

- [--oracle-redo-tablespace=TUNGSTEN_TS](#) [357]

Defines the tablespace name where the required user will be created for extracting data.

- [--oracle-redo-replicate-tables](#) [357]

that will be extracted from the source database. The VMware Redo Reader is configured to extract specific tablespaces from the source database.

Note

Note that the `tungsten_*` tables required by Tungsten Replicator to operate will also be included in the definition automatically. This is to ensure the correct replication of the information and validate the replication process.

- `--oracle-sys-user-password` [358]

The password of the Oracle `sys` user. This is required so that the required table configuration can take place.

- `--oracle-system-user-password` [358]

The password of the Oracle `SYSTEM` user. This is required so that the required table configuration can take place.

- `--oracle-extractor-method=redo` [356]

Specifies the extraction method to be used with Oracle. The `redo` method uses the VMware Redo Reader component.

- `--enable-heterogeneous-service=true` [344]

Enables heterogeneous service options, which include setting the correct filters and storage format so that information can effectively be replicated to non-Oracle databases.

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

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

Instead of providing the password on the command line, you may put the password into a file and provide the path of that file as the argument value. If you use this method, the password should be the only content in the file.

If the Oracle values are not correct, the `tpm` command may fail with an error like "There was an error while processing the VMware Redo Reader response file". Look at the `/tmp/tungsten-configure.log` file for more information about the root cause.

The replicator on a target host can be configured separately, using this host as the master.

5.1.4. Setting the Replication Start Position

Positions within the Oracle replication process can be identified and/or set using the Oracle System Change Number (SCN) reference. By setting the replication to start at a specific SCN you can take advantage of provisioning, or starting replication on specific known boundaries, for example, after a specific backup operation, or snapshot, has taken place.

By default `tpm` sets up Oracle replication to start at the current position of the Oracle DBMS, which is denoted by the current SCN. The replicator represents this value using the word `NOW`. To enable default behavior simply install normally and include the `tpm --start-and-report` option in the installation. The replicator will extract the first transaction committed after the time when the replicator was installed.

5.1.4.1. Setting a Specific Start Position

To start replication at a specific SCN in the past, you must bring the replicator online explicitly with an option that specifies the exact SCN where replication should start. Here are the exact steps.

1. Determine the SCN from which you would like to begin replication.
2. When installing the replicator do not include the `--start-and-report` option. `tpm` will install the replicator but will not start the service or bring it online.
3. After installation completes successfully, start the replicator on the source Oracle instance and bring it online using the following commands:

```
shell> replicator start offline
shell> trepctl online -from-event #####
```

4. Start the target replicator(s) normally using:

```
shell> replicator start
```

Replication will now start at the target SCN value. Here is a short example of how to select the target SCN and then bring the replicator online to pick up older changes.

```
SQL> select current_scn from v$database;
CURRENT_SCN
```

14140241

Depending on how old the SCN is the replicator may take a while to extract up to the current position of the Oracle instance. You can confirm that older data were extracted by looking in the log using the `thl` utility.

5.1.4.2. Resetting Existing Replication to a specific SCN

You can reset replicators to begin extracting from a specific SCN after replication has been installed using the following procedure:

1. Reset target and source replicators using the `reset` command to clear all logs:

```
shell> multi_trepctl offline
shell> multi_trepctl -service servicename reset -y
```

2. Bring the source replicator online starting at the desired SCN:

```
shell> trepctl -host sourcehost online -from-event 14140241
```

3. Bring the target replicator online normally:

```
shell> trepctl -host targethost online
```

5.1.4.3. Forcing replication to restart from a new SCN without resetting the THL

You can use the `-from-event` command to change the position of replication in the source Oracle instance. This allows you to skip over parts of the log or reposition replication without disturbing target replicators.

1. Bring the source replicator offline and reset the redo component only.

```
shell> trepctl -host sourcehost offline
shell> trepctl -host sourcehost -service servicename reset -redo -y
```

2. Bring the source replicator online starting at the desired SCN.

```
shell> trepctl -host sourcehost online -from-event 61754320
```

Target replicators will automatically reconnect to the source replicator.

5.1.5. Provisioning an Oracle Replication Solution

You can provision an empty target Oracle schema from an existing one and start Tungsten Replicator while positioning redo log extraction correctly.

The instructions below assume the following:

- Tungsten Replicator is not installed on each the source or the target servers.
- An active source Oracle installation with tables and data.
- An active target Oracle installation, without data, but with a matching schema.

Before starting the provisioning process:

- Make sure that Tungsten is not installed, no Tungsten Replicator internal tables exist (`trep_commit_seqno`) and no THL files are left from old installations.
- Make sure that VMware redo reader is not installed and PLOG files are left from previous installations.

5.1.5.1. Provisioning with Oracle Data Pump

Important

When using this method, no open transactions must be running. Active transactions are not extracted and result in inconsistencies

1. Login to Oracle as sysdba and get current SCN position:

```
shell> sqlplus / as sysdba
SQL> SELECT CURRENT_SCN FROM v$database;
CURRENT_SCN
```

```
-----  
6449931
```

2. On the source server export the full schema at this specific SCN position to generate a consistent dump across all tables (note the parameter `flashback_scn`).

```
shell> expdp system/Password11 directory=DATA_PUMP_DIR schemas=tungsten dumpfile=tungsten.dmp logfile=tungsten.log flashback_scn=6449931
```

3. Copy the generated files to the target server.

```
shell> scp /app/oracle/local/admin/ORCL/dpdump/tungsten.* oracleslave.example.com:/app/oracle/local/admin/ORCL/dpdump/
```

4. On the target server import the files.

```
shell> impdp tungsten/secret DIRECTORY=DATA_PUMP_DIR DUMPFILE=tungsten.dmp LOGFILE=tungsten.log SCHEMAS=tungsten
```

Note

If you get the following error messages when you try to import the data:

```
ORA-39002: invalid operation  
ORA-39070: Unable to open the log file.  
ORA-39087: directory name DATA_PUMP_DIR is invalid
```

Check the value of `DATA_PUMP_DIR` using:

```
SQL> select * from all_directories where directory_name = 'DATA_PUMP_DIR';
```

If you do not get a result, create the required directory:

```
SQL> CREATE DIRECTORY DATA_PUMP_DIR AS '/path';
```

If the directory already exists, then ensure the permissions for the user you are using to import the data:

```
SQL> GRANT READ, WRITE ON DIRECTORY DATA_PUMP_DIR TO tungsten;
```

5. Install replication on the master server, for example using the instructions in [Section 5.3.2.2.1, "Installing Oracle to Oracle Migration Replication - Basic Procedure using INI Files"](#).

However, when starting replication, ensure that you use the instructions for setting an explicit SCN number when the replication starts. See [Section 5.1.4.1, "Setting a Specific Start Position"](#) for more information on setting this parameter during replication startup.

```
shell> trepctl online -from-event 6449931
```

5.1.6. Oracle Redo Reader Tuning

The VMware Redo Reader extracts information from the Oracle Redo logs and places that information into an internal PLOG format. This contains the raw transactional data. Unlike the Transaction History Log (THL) used by Tungsten Replicator, the format of this information is not fully serialized. The serialization process is performed by the Tungsten Replicator while the data is being translated into the THL format.

The PLOG information is stored within the Tungsten Replicator installation directory, within a directory named after the corresponding service. For example, for the service alpha, the data is located in the `plog/alpha/mine/` directory. The log information in these files is automatically rotated and managed in the same manner as the THL files.

When processing very large transactions, for example when doing a bulk load or extraction, the information is temporarily serialized to disk. This can introduce additional disk space requirements when incorporating these large transactions into the THL.

5.2. Deploying Oracle Replication for Disaster Recovery

5.2.1. Prepare: Oracle Replication for Disaster Recovery

Prepare the Oracle environment as documented in section [Section 5.1.2, "Preparing the Oracle Environment for Replication"](#).

5.2.2. Install: Oracle Replication for Disaster Recovery

An Oracle to Oracle for Disaster Recovery deployment can be installed using the following methods.

- **Staging Configuration** — [Section 5.2.2.1, "Installing an Oracle for Disaster Recovery Topology \(Staging Use Case\)"](#)
- **INI Configuration** — [Section 5.2.2.2, "Installing an Oracle for Disaster Recovery Topology \(INI Use Case\)"](#)

5.2.2.1. Installing an Oracle for Disaster Recovery Topology (Staging Use Case)

Important

The VMware Redo Reader feature is only available in a specific commercial release of the Continuent Replicator. Please ensure you have downloaded the correct build (must begin with `vmware-continuent-replication-oracle-source-`), otherwise these procedures will fail.

5.2.2.1.1. Installing Oracle to Oracle DR Replication - Basic Procedure via SSH

This procedure will install Oracle to Oracle DR replication (including the VMware Redo Reader) to all member hosts via SSH from a single staging directory on a specified host.

This method requires including Oracle system passwords in the `tpm` configuration for the `sys` and `SYSTEM` Oracle users. If you are not comfortable with that, see [Section 5.2.2.1.2, “Installing Oracle to Oracle DR Replication - Advanced Procedure via SSH”](#).

The `tpm install` command will:

- Validate the Replicator and VMware Redo Reader
- Configure the VMware Redo Reader
- Configure the Replicator
- Start the VMware Redo Reader
- Start the Replicator

To install Tungsten Replication follow these steps:

1. Install the Tungsten Replicator™ package (`.rpm`), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell-staging> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Change to the Tungsten Replicator staging directory:

```
shell-staging> cd vmware-continuent-replication-oracle-source-5.0.1-136
```

3. Configure the staging directory with the desired defaults:

```
shell-staging> ./tools/tpm configure defaults --reset \
--install-directory=/opt/continuent \
--start-and-report=true \
--user=tungsten \
--replication-user=tungsten \
--replication-password=secret
```

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

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

- `tpm install`

Executes `tpm` in `configure` mode, configuring the default settings for all parts of the service, and also resetting any existing configuration (using the `--reset` [363]) option.

- `--replication-user=tungsten` [363]

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

- `--user=tungsten` [370]

The operating system user that the service will be executed using.

- `--install-directory=/opt/continuent` [347]

Directory where Tungsten Replication will be installed.

- `--replication-password=secret` [363]

The password that will be used for the replication service when connecting to the datasource.

- `--start-and-report=true` [365]

Once the service has been installed, start it automatically, and the report on the status.

4. Now configure the replicator specifics:

```
shell-staging> ./tools/tpm configure oracle2oracle \
--master=host1 \
--members=host1,host2 \
--datasource-type=oracle \
--datasource-oracle-service=ORCL \
--install-vmware-redo-reader=true \
--oracle-redo-tablespace=TUNGSTEN_TS \
--oracle-redo-replicate-tables=DB1,DB2,DB3 \
--oracle-sys-user-password=secret \
--oracle-system-user-password=secret \
--oracle-extractor-method=redo
```

Important

In Tungsten Replication 5.0.1 and higher, the `--enable-role-change` [345] option should be added to the configuration. This ensures that in the event of a switch between the two replicators, that both sides have the right configuration pipeline to support both master and slave operation, and that the replication position can be configured properly.

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` [351]

Specifies which host will be the master.

- `--members=host1,host2` [351]

The members of the service, in this case, the master (`host1`) and the slave (`host2`).

- `--datasource-type=oracle` [340]

Specifies the source database type, in this case, Oracle.

- `--datasource-oracle-service=ORCL` [338]

The Oracle service name. This must match the name of the Oracle service as defined in `oratab`.

- `--install-vmware-redo-reader=true` [348]

Enables automatic installation of the VMware Redo Reader to extract information from the Oracle Redo logs.

- `--oracle-redo-tablespace=TUNGSTEN_TS` [357]

Defines the tablespace name where the required user will be created for extracting data.

- `--oracle-redo-replicate-tables=SCHEMA[.TABLE][,...]` [357]

Specify the schemas and optionally the tables that will be extracted from the source database.

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the `TUNGSTEN` schema, and so it does NOT need to be added to the `--oracle-redo-replicate-tables` [357] configuration option.

The `TUNGSTEN` schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- `--oracle-sys-user-password` [358]

The password of the Oracle `sys` user. This is required so that the required table configuration can take place.

- `--oracle-system-user-password` [358]

The password of the Oracle `SYSTEM` user. This is required so that the required table configuration can take place.

- `--oracle-extractor-method=redo [356]`

Specifies the extraction method to be used with Oracle. The `redo` method uses the VMware Redo Reader component.

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

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

Instead of providing the password on the command line, you may put the password into a file and provide the path of that file as the argument value. If you use this method, the password should be the only content in the file.

If the Oracle values are not correct, the `tpm` command may fail with an error like "There was an error while processing the VMware Redo Reader response file". Look at the `/tmp/tungsten-configure.log` file for more information about the root cause.

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> vmrdrd_oracle2oracle status
shell-host1> trepctl status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> trepctl status
```

5.2.2.1.2. Installing Oracle to Oracle DR Replication - Advanced Procedure via SSH

This procedure will install Oracle to Oracle DR replication (excluding the VMware Redo Reader) to all member hosts via SSH from a single staging directory on a specified host.

The VMware Redo Reader will be installed separately on a per-host basis so that Oracle system passwords are not included in the `tpm` configuration for security reasons.

The `tpm` install command will:

- Validate the Replicator and VMware Redo Reader
- Configure the Replicator

To install Tungsten Replication follow these steps:

1. Install the Tungsten Replicator™ package (`.rpm`), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell-staging> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Change to the Tungsten Replicator staging directory:

```
shell-staging> cd vmware-continuent-replication-oracle-source-5.0.1-136
```

3. Configure the staging directory with the desired defaults:

```
shell-staging> ./tools/tpm configure defaults --reset \
--install-directory=/opt/continuent \
--user=tungsten \
--replication-user=tungsten \
--replication-password=secret
```

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

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

- [tpm install](#)

Executes `tpm` in `configure` mode, configuring the default settings for all parts of the service, and also resetting any existing configuration (using the `--reset` [363] option).

- [--replication-user=tungsten \[363\]](#)

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

- [--user=tungsten \[370\]](#)

The operating system user that the service will be executed using.

- `--install-directory=/opt/continuent [347]`

Directory where Tungsten Replication will be installed.

- `--replication-password=secret [363]`

The password that will be used for the replication service when connecting to the datasource.

4. Configure the Oracle extractor and overall topology:

```
shell-staging> $ staging> tools/tpm configure oracle2oracle \
--master=host1 \
--members=host1,host2 \
--datasource-type=oracle \
--datasource-oracle-service=ORCL \
--oracle-extractor-method=redo
```

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 [351]`

Specifies which host will be the master.

- `--members=host1,host2 [351]`

The members of the service, in this case, the master (`host1`) and the slave (`host2`).

- `--datasource-type=oracle [340]`

Specifies the source database type, in this case, Oracle.

- `--datasource-oracle-service=ORCL [338]`

The Oracle service name. This must match the name of the Oracle service as defined in `oratab`.

- `--install-vmware-redo-reader=true [348]`

Enables automatic installation of the VMware Redo Reader to extract information from the Oracle Redo logs.

- `--oracle-extractor-method=redo [356]`

Specifies the extraction method to be used with Oracle. The `redo` method uses the VMware Redo Reader component.

5. Once the pre-requisites and configuring of the installation has been completed, the software can be installed:

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

6. Install the VMware Redo Reader on host1 by running the installation from the installed Tungsten Replication directory:

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> tpm install-vmware-redo-reader oracle2oracle \
--oracle-redo-tablespace=TUNGSTEN_TS \
--oracle-sys-user-password=secret \
--oracle-system-user-password=secret \
--oracle-redo-replicate-tables=DB1,DB2,DB3
```

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.

- `--oracle-redo-tablespace=TUNGSTEN_TS [357]`

Defines the tablespace name where the required user will be created for extracting data.

- `--oracle-redo-replicate-tables=SCHEMA[.TABLE][,...] [357]`

Specify the schemas and optionally the tables that will be extracted from the source database.

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the `TUNGSTEN` schema, and so it does NOT need to be added to the `--oracle-redo-replicate-tables` [357] configuration option.

The `TUNGSTEN` schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- `--oracle-sys-user-password` [358]

The password of the Oracle `sys` user. This is required so that the required table configuration can take place.

- `--oracle-system-user-password` [358]

The password of the Oracle `SYSTEM` user. This is required so that the required table configuration can take place.

Instead of providing the password on the command line, you may put the password into a file and provide the path of that file as the argument value. If you use this method, the password should be the only content in the file.

If the Oracle values are not correct, the `tpm` command may fail with an error like "There was an error while processing the VMware Redo Reader response file". Look at the `/tmp/tungsten-configure.log` file for more information about the root cause.

7. Start all replication services on host1:

```
shell-host1> startall
```

8. Start all replication services on host2:

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> startall
```

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> vmrdd_oracle2oracle status
shell-host1> trepctl status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> trepctl status
```

5.2.2.2. Installing an Oracle for Disaster Recovery Topology (INI Use Case)

Important

The VMware Redo Reader feature is only available in a specific commercial release of the Continuent Replicator. Please ensure you have downloaded the correct build (must begin with `vmmware-continuent-replication-oracle-source-`), otherwise these procedures will fail.

5.2.2.2.1. Installing Oracle to Oracle DR Replication - Basic Procedure using INI Files

This procedure will install Oracle to Oracle DR replication (including the VMware Redo Reader) on a per-host basis using INI files. Repeat these steps on all member hosts.

This method requires including Oracle system passwords in the `tpm` ini file for the `sys` and `SYSTEM` Oracle users. If you are not comfortable with that, see [Section 5.2.2.2.2, "Installing Oracle to Oracle DR Replication - Advanced Procedure using INI Files"](#).

The `tpm install` command will:

- Validate the Replicator and VMware Redo Reader
- Configure the VMware Redo Reader
- Configure the Replicator
- Start the VMware Redo Reader
- Start the Replicator

To install Tungsten Replication follow these steps on ALL participating member hosts:

1. Install the Tungsten Replicator™ package ([.rpm](#)), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell-staging> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Create `/etc/tungsten/tungsten.ini` with the following configuration:

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

[oracle2oracle]
master=host1
members=host1,host2
datasource-type=oracle
datasource-oracle-service=ORCL
install-vmware-redo-reader=true
oracle-redo-tablespace=TUNGSTEN_TS
oracle-redo-replicate-tables=DB1,DB2,DB3
oracle-sys-user-password=secret
oracle-system-user-password=secret
oracle-extractor-method=redo
```

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 \[370\]](#)

The operating system user that the service will be executed using.

- [install-directory=/opt/continuent \[347\]](#)

Directory where Tungsten Replication will be installed.

- [replication-user=tungsten \[363\]](#)

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

- [replication-password=secret \[363\]](#)

The password that will be used for the replication service when connecting to the datasource.

- [start-and-report=true \[365\]](#)

Once the service has been installed, start it automatically, and the report on the status.

- [profile-script=~/.bash_profile \[361\]](#)

Tells `tpm` to add PATH information to the script to initialize the environment.

- [master=host1 \[351\]](#)

Specifies which host will be the master.

- [members=host1,host2 \[351\]](#)

The members of the service, in this case, the master (`host1`) and the slave (`host2`).

- [datasource-type=oracle \[340\]](#)

Specifies the source database type, in this case, Oracle.

- [datasource-oracle-service=ORCL \[338\]](#)

The Oracle service name. This must match the name of the Oracle service as defined in `oratab`.

- [install-vmware-redo-reader=true \[348\]](#)

Enables automatic installation of the VMware Redo Reader to extract information from the Oracle Redo logs.

- [oracle-redo-tablespace=TUNGSTEN_TS \[357\]](#)

Defines the tablespace name where the required user will be created for extracting data.

- `oracle-redo-replicate-tables=SCHEMA[.TABLE][,...]` [357]

Specify the schemas and optionally the tables that will be extracted from the source database.

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the `TUNGSTEN` schema, and so it does NOT need to be added to the `oracle-redo-replicate-tables` [357] configuration option.

The `TUNGSTEN` schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- `oracle-sys-user-password` [358]

The password of the Oracle `sys` user. This is required so that the required table configuration can take place.

- `oracle-system-user-password` [358]

The password of the Oracle `SYSTEM` user. This is required so that the required table configuration can take place.

- `oracle-extractor-method=redo` [356]

Specifies the extraction method to be used with Oracle. The `redo` method uses the VMware Redo Reader component.

3. Configure security. Use the procedure in section [Section 7.5, “Deployment Security”](#) to prepare and deploy the needed certificates, then edit the `/etc/tungsten/tungsten.ini` accordingly.

To proceed without using the secure installation mode, add the following to `/etc/tungsten/tungsten.ini` on all member hosts:

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

4. Run `tpm` from the extracted software staging directory on all member hosts to install the software using the INI-based configuration:

```
shell> cd vmware-continuent-replication-oracle-source-5.0.1-136
shell > ./tools/tpm install
```

During the startup and installation, `tpm` will notify you of any problems that need to be fixed before the service can be correctly installed and started. If `start-and-report` [365] is set and the service starts correctly, you should see the configuration and current status of the service.

Instead of providing the password on the command line, you may put the password into a file and provide the path of that file as the argument value. If you use this method, the password should be the only content in the file.

If the Oracle values are not correct, the `tpm` command may fail with an error like "There was an error while processing the VMware Redo Reader response file". Look at the `/tmp/tungsten-configure.log` file for more information about the root cause.

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> vmrdrd_oracle2oracle status
shell-host1> trepctl status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> trepctl status
```

5.2.2.2.2. Installing Oracle to Oracle DR Replication - Advanced Procedure using INI Files

This procedure will install Oracle to Oracle DR replication on a per-host basis using INI files. Repeat these steps on all member hosts.

The VMware Redo Reader and the Replicator will be installed separately so that Oracle system passwords are not included in the `tpm` configuration for security reasons.

The `tpm` install command will:

- Validate the Replicator and VMware Redo Reader

- Configure the Replicator

To install Tungsten Replication follow these steps:

1. Install the Tungsten Replicator™ package (.rpm), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell-staging> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Create `/etc/tungsten/tungsten.ini` with the following configuration:

```
[defaults]
user=tungsten
install-directory=/opt/continuent
replication-user=tungsten
replication-password=secret
profile-script=~/bash_profile

[oracle2oracle]
master=host1
members=host1,host2
datasource-type=oracle
datasource-oracle-service=ORCL
install-vmware-redo-reader=false
oracle-extractor-method=redo
```

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` [370]

The operating system user that the service will be executed using.

- `install-directory=/opt/continuent` [347]

Directory where Tungsten Replication will be installed.

- `replication-user=tungsten` [363]

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

- `replication-password=secret` [363]

The password that will be used for the replication service when connecting to the datasource.

- `profile-script=~/bash_profile` [361]

Tells `tpm` to add PATH information to the script to initialize the environment.

- `master=host1` [351]

Specifies which host will be the master.

- `members=host1,host2` [351]

The members of the service, in this case, the master (`host1`) and the slave (`host2`).

- `datasource-type=oracle` [340]

Specifies the source database type, in this case, Oracle.

- `datasource-oracle-service=ORCL` [338]

The Oracle service name. This must match the name of the Oracle service as defined in `oratab`.

- `install-vmware-redo-reader=false` [348]

Disables automatic installation of the VMware Redo Reader which extracts information from the Oracle Redo logs.

- `oracle-extractor-method=redo` [356]

Specifies the extraction method to be used with Oracle. The `redo` method uses the VMware Redo Reader component.

3. Configure security. Use the procedure in section [Section 7.5, “Deployment Security”](#) to prepare and deploy the needed certificates, then edit the `/etc/tungsten/tungsten.ini` accordingly.

To proceed without using the secure installation mode, add the following to `/etc/tungsten/tungsten.ini` on all member hosts:

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

- Once the pre-requisites and configuring of the installation have been completed, the software can be installed:

```
shell> cd vmware-continuent-replication-oracle-source-5.0.1-136
shell-staging> ./tools/tpm install
```

- Install the VMware Redo Reader on host1 by running the installation from the installed Tungsten Replication directory:

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> tpm install-vmware-redo-reader oracle2oracle \
--oracle-redo-tablespace=TUNGSTEN_TS \
--oracle-sys-user-password=secret \
--oracle-system-user-password=secret \
--oracle-redo-replicate-tables=DB1,DB2,DB3
```

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.

- `--oracle-redo-tablespace=TUNGSTEN_TS` [357]

Defines the tablespace name where the required user will be created for extracting data.

- `--oracle-redo-replicate-tables=SCHEMA[.TABLE][,...]` [357]

Specify the schemas and optionally the tables that will be extracted from the source database.

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the `TUNGSTEN` schema, and so it does NOT need to be added to the `--oracle-redo-replicate-tables` [357] configuration option.

The `TUNGSTEN` schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- `--oracle-sys-user-password` [358]

The password of the Oracle `sys` user. This is required so that the required table configuration can take place.

- `--oracle-system-user-password` [358]

The password of the Oracle `SYSTEM` user. This is required so that the required table configuration can take place.

Instead of providing the password on the command line, you may put the password into a file and provide the path of that file as the argument value. If you use this method, the password should be the only content in the file.

If the Oracle values are not correct, the `tpm` command may fail with an error like "There was an error while processing the VMware Redo Reader response file". Look at the `/tmp/tungsten-configure.log` file for more information about the root cause.

- Start all replication services on host1:

```
shell-host1> startall
```

- Start all replication services on host2:

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> startall
```

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> vmrdrd_oracle2oracle status
shell-host1> treptctl status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> treptctl status
```

5.2.3. Best Practices: Oracle Replication for Disaster Recovery

Follow the guidelines in Section 2.2, "Best Practices".

5.2.3.1. Management and Monitoring of Oracle Replication for Disaster Recovery

Check replication status on host1

```
shell> ./opt/continuent/share/env.sh  
shell> trepctl status  
shell> vmrrd_oracle2oracle status
```

Check replication status on host2

```
shell> ./opt/continuent/share/env.sh  
shell> trepctl status
```

5.2.3.2. Performing a Role Switch

To perform a role switch, for example, switching an existing primary (or master) to be a secondary (or slave), the following steps should be followed:

Important

To enable role switching, we recommend using Tungsten Replication 5.0.1 or later. You should also enable the `enable-role-change=true` [345] option to `tpm`.

1. Put the master replicator `OFFLINE` [207]:

```
shell> trepctl -host host1 offline
```

2. Put the slave replicator `OFFLINE` [207]:

```
shell> trepctl -host host2 offline
```

3. Change the role of the existing slave replicator to a master:

```
shell> trepctl -host host2 setrole -role master
```

4. Change the role of the existing master to be a slave, using the new master hostname within the configuration:

```
shell> trepctl -host host1 setrole -role slave -uri thls://host2
```

5. Put the new master `ONLINE` [207]:

```
shell> trepctl -host host2 online
```

6. Wait for a maximum of 120 seconds until the replicator is `ONLINE` [207]:

```
shell> trepctl -host host2 wait -state ONLINE -limit 120
```

7. Put the new slave `ONLINE` [207]:

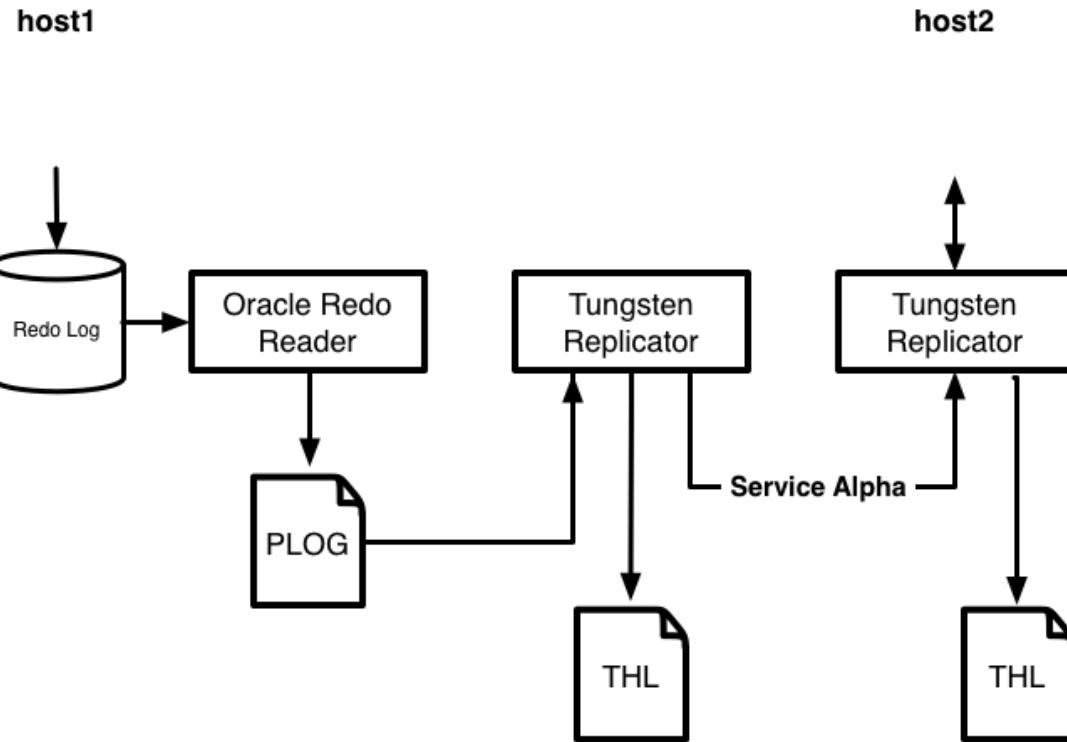
```
shell> trepctl -host host1 online
```

8. Wait for 120 seconds to ensure that the slave is `ONLINE` [207]:

```
shell> trepctl -host host1 wait -state ONLINE -limit 120
```

5.3. Deploying Oracle Replication for Migration

Figure 5.2. Topologies: Oracle to Oracle with Redo Reader for Migration



5.3.1. Prepare: Oracle Replication for Migration

Prepare the Oracle environment as documented in section [Section 5.1.2, "Preparing the Oracle Environment for Replication"](#).

5.3.2. Install: Oracle Replication for Migration

An Oracle to Oracle migration deployment can be installed using the following methods.

- Staging Configuration — [Section 5.3.2.1, "Installing an Oracle for Migration Topology \(Staging Use Case\)"](#)
- INI Configuration — [Section 5.3.2.2, "Installing an Oracle for Migration Topology \(INI Use Case\)"](#)

5.3.2.1. Installing an Oracle for Migration Topology (Staging Use Case)

Important

The VMware Redo Reader feature is only available in a specific commercial release of the Continuent Replicator. Please ensure you have downloaded the correct build (must begin with `vmware-continuent-replication-oracle-source-`), otherwise these procedures will fail.

5.3.2.1.1. Installing Oracle to Oracle Migration Replication - Basic Procedure via SSH

This procedure will install Oracle to Oracle Migration replication (including the VMware Redo Reader) to all member hosts via SSH from a single staging directory on a specified host.

Important

The extractor on the secondary/target will not be installed since it will never take on the primary role.

This method requires including Oracle system passwords in the `tpm` configuration for the `sys` and `SYSTEM` Oracle users. If you are not comfortable with that, see [Section 5.3.2.1.2, “Installing Oracle to Oracle Migration Replication - Advanced Procedure via SSH”](#).

The `tpm install` command will:

- Validate the Replicator and VMware Redo Reader
- Configure the VMware Redo Reader
- Configure the Replicator
- Start the VMware Redo Reader
- Start the Replicator

To install Tungsten Replication follow these steps:

1. Install the Tungsten Replicator™ package (`.rpm`), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell-staging> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Change to the Tungsten Replicator staging directory:

```
shell-staging> cd vmware-continuent-replication-oracle-source-5.0.1-136
```

3. Configure the staging directory with the desired defaults:

```
shell-staging> ./tools/tpm configure defaults --reset \
--install-directory=/opt/continuent \
--start-and-report=true \
--user=tungsten \
--replication-user=tungsten \
--replication-password=secret
```

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

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

- `tpm install`

Executes `tpm` in `configure` mode, configuring the default settings for all parts of the service, and also resetting any existing configuration (using the `--reset` [363]) option.

- `--replication-user=tungsten` [363]

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

- `--user=tungsten` [370]

The operating system user that the service will be executed using.

- `--install-directory=/opt/continuent` [347]

Directory where Tungsten Replication will be installed.

- `--replication-password=secret` [363]

The password that will be used for the replication service when connecting to the datasource.

- `--start-and-report=true` [365]

Once the service has been installed, start it automatically, and the report on the status.

4. Now configure the replicator specifics:

```
shell-staging> ./tools/tpm configure oracle2oracle \
--master-host1 \
--members=host1,host2 \
--datasource-type=oracle \
--datasource-oracle-service=ORCL \
--install-vmware-redo-reader=true \
```

```
--oracle-redo-tablespace=TUNGSTEN_TS \
--oracle-redo-replicate-tables=DB1,DB2,DB3 \
--oracle-sys-user-password=secret \
--oracle-system-user-password=secret \
--oracle-extractor-method=redo
```

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 \[351\]](#)

Specifies which host will be the master.

- [--members=host1,host2 \[351\]](#)

The members of the service, in this case, the master ([host1](#)) and the slave ([host2](#)).

- [--datasource-type=oracle \[340\]](#)

Specifies the source database type, in this case, Oracle.

- [--datasource-oracle-service=ORCL \[338\]](#)

The Oracle service name. This must match the name of the Oracle service as defined in [oratab](#).

- [--install-vmware-redo-reader=true \[348\]](#)

Enables automatic installation of the VMware Redo Reader to extract information from the Oracle Redo logs.

- [--oracle-redo-tablespace=TUNGSTEN_TS \[357\]](#)

Defines the tablespace name where the required user will be created for extracting data.

- [--oracle-redo-replicate-tables=SCHEMA\[.TABLE\]\[,...\] \[357\]](#)

Specify the schemas and optionally the tables that will be extracted from the source database.

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the [TUNGSTEN](#) schema, and so it does NOT need to be added to the [--oracle-redo-replicate-tables \[357\]](#) configuration option.

The [TUNGSTEN](#) schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- [--oracle-sys-user-password \[358\]](#)

The password of the Oracle [sys](#) user. This is required so that the required table configuration can take place.

- [--oracle-system-user-password \[358\]](#)

The password of the Oracle [SYSTEM](#) user. This is required so that the required table configuration can take place.

- [--oracle-extractor-method=redo \[356\]](#)

Specifies the extraction method to be used with Oracle. The [redo](#) method uses the VMware Redo Reader component.

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

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

Instead of providing the password on the command line, you may put the password into a file and provide the path of that file as the argument value. If you use this method, the password should be the only content in the file.

If the Oracle values are not correct, the tpm command may fail with an error like "There was an error while processing the VMware Redo Reader response file". Look at the [/tmp/tungsten-configure.log](#) file for more information about the root cause.

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
```

```
shell> vmmrd_oracle2oracle status
shell> trepctl status
```

Check replication status on host2

```
shell> . /opt/continuent/share/env.sh
shell> trepctl status
```

5.3.2.1.2. Installing Oracle to Oracle Migration Replication - Advanced Procedure via SSH

This procedure will install Oracle to Oracle DR replication (excluding the VMware Redo Reader) to all member hosts via SSH from a single staging directory on a specified host.

The VMware Redo Reader will be installed separately on a per-host basis so that Oracle system passwords are not included in the `tpm` configuration for security reasons.

Important

The extractor on the secondary/target will not be installed since it will never take on the primary role.

The `tpm install` command will:

- Validate the Replicator and VMware Redo Reader
- Configure the Replicator

To install Tungsten Replication follow these steps:

1. Install the Tungsten Replicator™ package (`.rpm`), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Change to the Tungsten Replicator staging directory:

```
shell> cd vmware-continuent-replication-oracle-source-5.0.1-136
```

3. Configure the staging directory with the desired defaults:

```
shell> ./tools/tpm configure defaults --reset \
--install-directory=/opt/continuent \
--user=tungsten \
--replication-user=tungsten \
--replication-password=secret
```

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

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

- `tpm install`

Executes `tpm` in `configure` mode, configuring the default settings for all parts of the service, and also resetting any existing configuration (using the `--reset` [363]) option.

- `--replication-user=tungsten` [363]

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

- `--user=tungsten` [370]

The operating system user that the service will be executed using.

- `--install-directory=/opt/continuent` [347]

Directory where Tungsten Replication will be installed.

- `--replication-password=secret` [363]

The password that will be used for the replication service when connecting to the datasource.

4. Configure the Oracle extractor and overall topology:

```
shell> tools/tpm configure oracle2oracle \
--master=host1 \
--members=host1,host2 \
--datasource-type=oracle \
```

```
--datasource-oracle-service=ORCL \
--oracle-extractor-method=redo
```

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 \[351\]](#)

Specifies which host will be the master.

- [--members=host1,host2 \[351\]](#)

The members of the service, in this case, the master ([host1](#)) and the slave ([host2](#)).

- [--datasource-type=oracle \[340\]](#)

Specifies the source database type, in this case, Oracle.

- [--datasource-oracle-service=ORCL \[338\]](#)

The Oracle service name. This must match the name of the Oracle service as defined in [oratab](#).

- [--install-vmware-redo-reader=true \[348\]](#)

Enables automatic installation of the VMware Redo Reader to extract information from the Oracle Redo logs.

- [--oracle-extractor-method=redo \[356\]](#)

Specifies the extraction method to be used with Oracle. The [redo](#) method uses the VMware Redo Reader component.

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

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

- Install the VMware Redo Reader on host1 by running the installation from the installed Tungsten Replication directory:

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> tpm install-vmware-redo-reader oracle2oracle \
--oracle-redo-tablespace=TUNGSTEN_TS \
--oracle-sys-user-password=secret \
--oracle-system-user-password=secret \
--oracle-redo-replicate-tables=DB1,DB2,DB3
```

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.

- [--oracle-redo-tablespace=TUNGSTEN_TS \[357\]](#)

Defines the tablespace name where the required user will be created for extracting data.

- [--oracle-redo-replicate-tables=SCHEMA\[.TABLE\]\[,...\] \[357\]](#)

Specify the schemas and optionally the tables that will be extracted from the source database.

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the [TUNGSTEN](#) schema, and so it does NOT need to be added to the [--oracle-redo-replicate-tables \[357\]](#) configuration option.

The [TUNGSTEN](#) schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- [--oracle-sys-user-password \[358\]](#)

The password of the Oracle [sys](#) user. This is required so that the required table configuration can take place.

- [--oracle-system-user-password \[358\]](#)

The password of the Oracle [SYSTEM](#) user. This is required so that the required table configuration can take place.

Instead of providing the password on the command line, you may put the password into a file and provide the path of that file as the argument value. If you use this method, the password should be the only content in the file.

If the Oracle values are not correct, the `tpm` command may fail with an error like "There was an error while processing the VMware Redo Reader response file". Look at the `/tmp/tungsten-configure.log` file for more information about the root cause.

7. Start all the services on host1:

```
shell-host1> startall
```

8. Start the replicator on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> startall
```

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> vmrdd_oracle2oracle status
shell-host1> trepctl status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> trepctl status
```

5.3.2.2. Installing an Oracle for Migration Topology (INI Use Case)

Important

The VMware Redo Reader feature is only available in a specific commercial release of the Continuent Replicator. Please ensure you have downloaded the correct build (must begin with `vmware-continuent-replication-oracle-source-`), otherwise these procedures will fail.

5.3.2.2.1. Installing Oracle to Oracle Migration Replication - Basic Procedure using INI Files

This procedure will install Oracle to Oracle DR replication (including the VMware Redo Reader) on a per-host basis using INI files. Repeat these steps on all member hosts.

Important

The extractor on the secondary/target will not be installed since it will never take on the primary role.

This method requires including Oracle system passwords in the `tpm` configuration for the `sys` and `SYSTEM` Oracle users. If you are not comfortable with that, see [Section 5.3.2.2.2, “Installing Oracle to Oracle Migration Replication - Advanced Procedure using INI Files”](#).

The `tpm install` command will:

- Validate the Replicator and VMware Redo Reader
- Configure the VMware Redo Reader
- Configure the Replicator
- Start the VMware Redo Reader
- Start the Replicator

To install Tungsten Replication follow these steps on ALL participating member hosts:

1. Install the Tungsten Replicator™ package (`.rpm`), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell-staging> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Create `/etc/tungsten/tungsten.ini` with the following configuration:

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

[oracle2oracle]
master=host1
```

```
members=host1,host2
datasource-type=oracle
datasource-oracle-service=ORCL
install-vmware-redo-reader=true
oracle-redo-tablespace=TUNGSTEN_TS
oracle-redo-replicate-tables=DB1,DB2,DB3
oracle-sys-user-password=secret
oracle-system-user-password=secret
oracle-extractor-method=redo
```

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 \[370\]](#)

The operating system user that the service will be executed using.

- [install-directory=/opt/continuent \[347\]](#)

Directory where Tungsten Replication will be installed.

- [replication-user=tungsten \[363\]](#)

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

- [replication-password=secret \[363\]](#)

The password that will be used for the replication service when connecting to the datasource.

- [start-and-report=true \[365\]](#)

Once the service has been installed, start it automatically, and the report on the status.

- [profile-script=~/.bash_profile \[361\]](#)

Tells `tpm` to add PATH information to the script to initialize the environment.

- [master=host1 \[351\]](#)

Specifies which host will be the master.

- [members=host1,host2 \[351\]](#)

The members of the service, in this case, the master (`host1`) and the slave (`host2`).

- [datasource-type=oracle \[340\]](#)

Specifies the source database type, in this case, Oracle.

- [datasource-oracle-service=ORCL \[338\]](#)

The Oracle service name. This must match the name of the Oracle service as defined in `oratab`.

- [install-vmware-redo-reader=true \[348\]](#)

Enables automatic installation of the VMware Redo Reader to extract information from the Oracle Redo logs.

- [oracle-redo-tablespace=TUNGSTEN_TS \[357\]](#)

Defines the tablespace name where the required user will be created for extracting data.

- [oracle-redo-replicate-tables=SCHEMA\[.TABLE\]\[,...\] \[357\]](#)

Specify the schemas and optionally the tables that will be extracted from the source database.

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the `TUNGSTEN` schema, and so it does NOT need to be added to the `oracle-redo-replicate-tables [357]` configuration option.

[!] The `TUNGSTEN` schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- `oracle-sys-user-password` [358]

The password of the Oracle `sys` user. This is required so that the required table configuration can take place.

- `oracle-system-user-password` [358]

The password of the Oracle `SYSTEM` user. This is required so that the required table configuration can take place.

- `oracle-extractor-method=redo` [356]

Specifies the extraction method to be used with Oracle. The `redo` method uses the VMware Redo Reader component.

3. Configure security. Use the procedure in section [Section 7.5, “Deployment Security”](#) to prepare and deploy the needed certificates, then edit the `/etc/tungsten/tungsten.ini` accordingly.

To proceed without using the secure installation mode, add the following to `/etc/tungsten/tungsten.ini` on all member hosts:

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

4. Run `tpm` from the extracted software staging directory on all member hosts to install the software using the INI-based configuration:

```
shell> cd vmware-continuent-replication-oracle-source-5.0.1-136
shell > ./tools/tpm install
```

During the startup and installation, `tpm` will notify you of any problems that need to be fixed before the service can be correctly installed and started. If `start-and-report` [365] is set and the service starts correctly, you should see the configuration and current status of the service.

Instead of providing the password on the command line, you may put the password into a file and provide the path of that file as the argument value. If you use this method, the password should be the only content in the file.

If the Oracle values are not correct, the `tpm` command may fail with an error like "There was an error while processing the VMware Redo Reader response file". Look at the `/tmp/tungsten-configure.log` file for more information about the root cause.

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> vmrdd_oracle2oracle status
shell-host1> trepctl status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> trepctl status
```

5.3.2.2.2. Installing Oracle to Oracle Migration Replication - Advanced Procedure using INI Files

This procedure will install Oracle to Oracle Migration replication on a per-host basis using INI files. The VMware Redo Reader and the Replicator will be installed separately so that Oracle system passwords are not included in the `tpm` configuration for security reasons. Repeat these steps on all member hosts.

The `tpm` install command will:

- Validate the Replicator and VMware Redo Reader
- Configure the Replicator

To install Tungsten Replication follow these steps:

1. Install the Tungsten Replicator™ package (`.rpm`), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell-staging> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Create `/etc/tungsten/tungsten.ini` with the following configuration:

```
[defaults]
user=tungsten
install-directory=/opt/continuent
replication-user=tungsten
```

```

replication-password=secret
profile-script=~/bash_profile

[oracle2oracle]
master=host1
members=host1,host2
datasource-type=oracle
datasource-oracle-service=ORCL
install-vmware-redo-reader=false
oracle-extractor-method=redo

```

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 \[370\]](#)

The operating system user that the service will be executed using.

- [install-directory=/opt/continuent \[347\]](#)

Directory where Tungsten Replication will be installed.

- [replication-user=tungsten \[363\]](#)

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

- [replication-password=secret \[363\]](#)

The password that will be used for the replication service when connecting to the datasource.

- [profile-script=~/bash_profile \[361\]](#)

Tells `tpm` to add PATH information to the script to initialize the environment.

- [master=host1 \[351\]](#)

Specifies which host will be the master.

- [members=host1,host2 \[351\]](#)

The members of the service, in this case, the master (`host1`) and the slave (`host2`).

- [datasource-type=oracle \[340\]](#)

Specifies the source database type, in this case, Oracle.

- [datasource-oracle-service=ORCL \[338\]](#)

The Oracle service name. This must match the name of the Oracle service as defined in `oratab`.

- [install-vmware-redo-reader=false \[348\]](#)

Disables automatic installation of the VMware Redo Reader which extracts information from the Oracle Redo logs.

- [oracle-extractor-method=redo \[356\]](#)

Specifies the extraction method to be used with Oracle. The `redo` method uses the VMware Redo Reader component.

3. Configure security. Use the procedure in section [Section 7.5, “Deployment Security”](#) to prepare and deploy the needed certificates, then edit the `/etc/tungsten/tungsten.ini` accordingly.

To proceed without using the secure installation mode, add the following to `/etc/tungsten/tungsten.ini` on all member hosts:

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

4. Once the pre-requisites and configuring of the installation has been completed, the software can be installed:

```

shell> cd vmware-continuent-replication-oracle-source-5.0.1-136
shell-staging> ./tools/tpm install

```

5. Install the VMware Redo Reader on host1 by running the installation from the installed Tungsten Replication directory:

```

shell-host1> ./opt/continuent/share/env.sh
shell-host1> tpm install-vmware-redo-reader oracle2oracle \

```

```
--oracle-redo-tablespace=TUNGSTEN_TS \
--oracle-sys-user-password=secret \
--oracle-system-user-password=secret \
--oracle-redo-replicate-tables=DB1,DB2,DB3
```

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.

- [--oracle-redo-tablespace=TUNGSTEN_TS \[357\]](#)

Defines the tablespace name where the required user will be created for extracting data.

- [--oracle-redo-replicate-tables=SCHEMA\[.TABLE\]\[,...\] \[357\]](#)

Specify the schemas and optionally the tables that will be extracted from the source database.

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the `TUNGSTEN` schema, and so it does NOT need to be added to the `--oracle-redo-replicate-tables [357]` configuration option.

The `TUNGSTEN` schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- [--oracle-sys-user-password \[358\]](#)

The password of the Oracle `sys` user. This is required so that the required table configuration can take place.

- [--oracle-system-user-password \[358\]](#)

The password of the Oracle `SYSTEM` user. This is required so that the required table configuration can take place.

Instead of providing the password on the command line, you may put the password into a file and provide the path of that file as the argument value. If you use this method, the password should be the only content in the file.

If the Oracle values are not correct, the `tpm` command may fail with an error like "There was an error while processing the VMware Redo Reader response file". Look at the `/tmp/tungsten-configure.log` file for more information about the root cause.

6. Start all replication services on host1:

```
shell-host1> startall
```

7. Start all replication services on host2:

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> startall
```

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> vmrdd_oracle2oracle status
shell-host1> trepctl status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> trepctl status
```

5.3.3. Best Practices: Oracle Replication for Migration

Follow the guidelines in Section 2.2, "Best Practices".

5.3.3.1. Management and Monitoring of Oracle Replication for Migration

Check replication status on host1

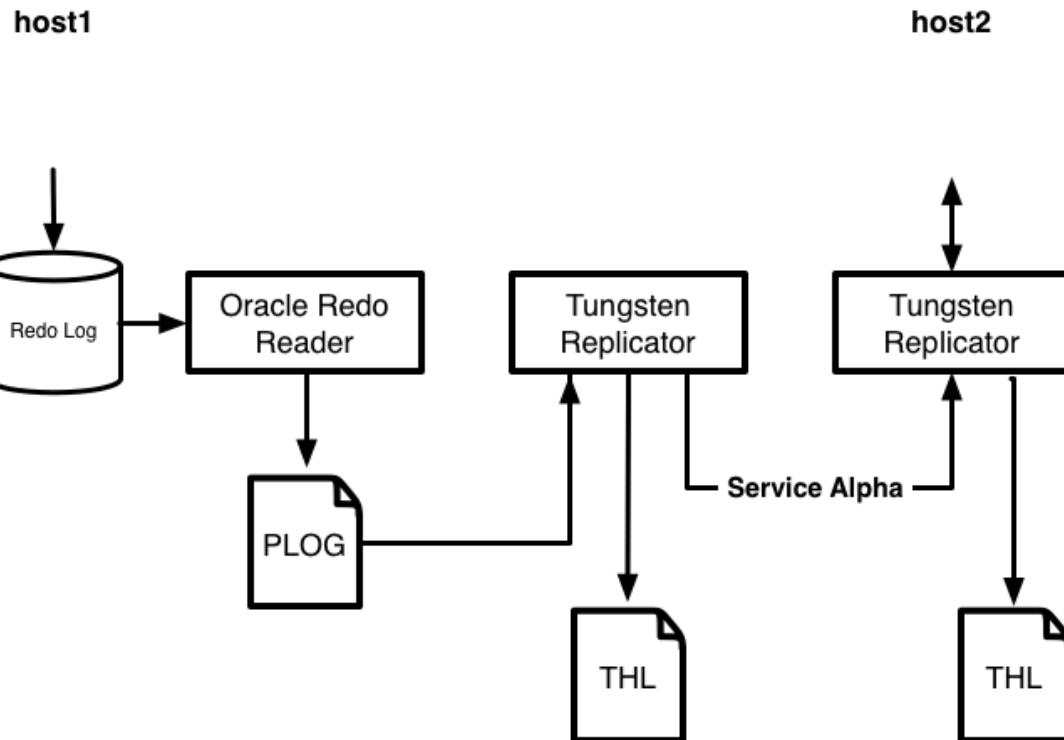
```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> trepctl status
shell-host1> vmrdd_oracle2oracle status
```

Check replication status on host2

```
shell> ./opt/continuent/share/env.sh
shell> trepctl status
```

5.4. Deploying Oracle to MySQL Replication

Figure 5.3. Topologies: Oracle to MySQL with Redo Reader



5.4.1. Prepare: Oracle to MySQL Replication

Prepare the Oracle environment as documented in section [Section 5.1.2, “Preparing the Oracle Environment for Replication”](#).

5.4.2. Install: Oracle to MySQL Replication

An Oracle to MySQL deployment can be installed using the following methods.

- **Staging Configuration** — [Section 5.4.2.1, “Installing an Oracle to MySQL Topology \(Staging Use Case\)”](#)
- **INI Configuration** — [Section 5.4.2.2, “Installing an Oracle to MySQL Topology \(INI Use Case\)”](#)

5.4.2.1. Installing an Oracle to MySQL Topology (Staging Use Case)

Important

The VMware Redo Reader feature is only available in a specific commercial release of the Continuent Replicator. Please ensure you have downloaded the correct build (must begin with `vmware-continuent-replication-oracle-source-`), otherwise these procedures will fail.

5.4.2.1.1. Installing Oracle to MySQL Replication - Basic Procedure via SSH

This procedure will install Oracle to MySQL replication (including the VMware Redo Reader) to all member hosts via SSH from a single staging directory on a specified host.

This method requires including Oracle system passwords in the `tpm` configuration for the `sys` and `SYSTEM` Oracle users. If you are not comfortable with that, see [Section 5.4.2.1.2, "Installing Oracle to MySQL Replication - Advanced Procedure via SSH"](#).

The `tpm install` command will:

- Validate the Replicator and VMware Redo Reader
- Configure the VMware Redo Reader
- Configure the Replicator
- Start the VMware Redo Reader
- Start the Replicator

To install Tungsten Replication follow these steps:

1. Install the Tungsten Replicator™ package (`.rpm`), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell-staging> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Change to the Tungsten Replicator staging directory:

```
shell-staging> cd vmware-continuent-replication-oracle-source-5.0.1-136
```

3. Configure the staging directory with the desired defaults:

```
shell-staging> ./tools/tpm configure defaults --reset \
--install-directory=/opt/continuent \
--start-and-report=true \
--user=tungsten \
--replication-user=tungsten \
--replication-password=secret
```

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.

- [--replication-user=tungsten \[363\]](#)

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

- [--user=tungsten \[370\]](#)

The operating system user that the service will be executed using.

- [--install-directory=/opt/continuent \[347\]](#)

Directory where Tungsten Replication will be installed.

- [--replication-password=secret \[363\]](#)

The password that will be used for the replication service when connecting to the datasource.

- [--start-and-report=true \[365\]](#)

Once the service has been installed, start it automatically, and the report on the status.

4. Now configure the replication topology:

```
shell-staging> ./tools/tpm configure oracle2mysql \
--enable-heterogeneous-service=true \
--master=host1 \
--members=host1,host2
```

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 [351]`

Specifies which host will be the master.

- `--members=host1,host2 [351]`

The members of the service, in this case, the master (`host1`) and the slave (`host2`).

- `--enable-heterogeneous-service=true [344]`

When configuring heterogeneous services it can be easier to configure them without security controls, as the certificates will need to be created centrally and applied on both servers during installation. For more information, see [Section 7.5, "Deployment Security"](#).

5. Configure the Oracle replicator for extracting data:

```
shell-staging> ./tools/tpm configure oracle2mysql \
  --hosts=host1 \
  --datasource-type=oracle \
  --datasource-oracle-service=ORCL \
  --install-vmware-redo-reader=true \
  --oracle-redo-tablespace=TUNGSTEN_TS \
  --oracle-sys-user-password=secret \
  --oracle-system-user-password=secret \
  --oracle-redo-replicate-tables=DB1,DB2,DB3 \
  --oracle-extractor-method=redo \
  --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.

- `--hosts=host1 [347]`

Configure the settings for this host only.

- `--datasource-type=oracle [340]`

Specifies the source database type, in this case, Oracle.

- `--datasource-oracle-service=ORCL [338]`

The Oracle service name. This must match the name of the Oracle service as defined in `oratab`.

- `--install-vmware-redo-reader=true [348]`

Enables automatic installation of the VMware Redo Reader to extract information from the Oracle Redo logs.

- `--oracle-redo-tablespace=TUNGSTEN_TS [357]`

Defines the tablespace name where the required user will be created for extracting data.

- `--oracle-redo-replicate-tables=SCHEMA[.TABLE][,...] [357]`

Specify the schemas and optionally the tables that will be extracted from the source database.

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the `TUNGSTEN` schema, and so it does NOT need to be added to the `--oracle-redo-replicate-tables [357]` configuration option.

The `TUNGSTEN` schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- `--oracle-sys-user-password [358]`

The password of the Oracle `sys` user. This is required so that the required table configuration can take place.

- `--oracle-system-user-password [358]`

The password of the Oracle `SYSTEM` user. This is required so that the required table configuration can take place.

- `--oracle-extractor-method=redo` [356]

Specifies the extraction method to be used with Oracle. The `redo` method uses the VMware Redo Reader component.

- `--user=tungsten` [370]

The operating system user that the service will be executed using.

6. Configure the MySQL applier:

```
shell-staging> ./tools/tpm configure oracle2mysql \
  --hosts=host2 \
  --datasource-type=mysql \
  --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.

- `--hosts=host2` [347]

Specifies which host will be the master.

- `--datasource-type=mysql` [340]

Specifies the source database type, in this case, MySQL.

- `--user=tungsten` [370]

The OS user that will be running the replicator.

7. Once the pre-requisites and configuring of the installation has been completed, the software can be installed:

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

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> trepctl status
shell-host1> vmrrd_oracle2mysql status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> trepctl status
```

5.4.2.1.2. Installing Oracle to MySQL Replication - Advanced Procedure via SSH

This procedure will install Oracle to MySQL replication (excluding the VMware Redo Reader) to all member hosts via SSH from a single staging directory on a specified host.

The VMware Redo Reader will be installed separately on a per-host basis so that Oracle system passwords are not included in the `tpm` configuration for security reasons.

The `tpm` install command will:

- Validate the Replicator and VMware Redo Reader
- Configure the Replicator

To install Tungsten Replication follow these steps:

1. Install the Tungsten Replicator™ package (`.rpm`), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell-staging> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Change to the Tungsten Replicator staging directory:

```
shell-staging> cd vmware-continuent-replication-oracle-source-5.0.1-136
```

3. Configure the staging directory with the desired defaults:

```
shell-staging> ./tools/tpm configure defaults --reset \
--install-directory=/opt/continuent \
```

```
--start-and-report=true \
--user=tungsten \
--replication-user=tungsten \
--replication-password=secret
```

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.

- [--replication-user=tungsten \[363\]](#)

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

- [--user=tungsten \[370\]](#)

The operating system user that the service will be executed using.

- [--install-directory=/opt/continuent \[347\]](#)

Directory where Tungsten Replication will be installed.

- [--replication-password=secret \[363\]](#)

The password that will be used for the replication service when connecting to the datasource.

- [--start-and-report=true \[365\]](#)

Once the service has been installed, start it automatically, and the report on the status.

4. Configure the basic topology:

```
shell-staging> ./tools/tpm configure oracle2mysql \
--master=host1 \
--members=host1,host2
```

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 \[351\]](#)

Specifies which host will be the master.

- [--members=host1,host2 \[351\]](#)

The members of the service, in this case, the master (`host1`) and the slave (`host2`).

5. Now configure the replicator for extraction from Oracle, without also automatically configuring the VMware Redo Reader:

```
shell-staging> ./tools/tpm configure oracle2mysql \
--hosts=host1 \
--user=tungsten \
--datasource-type=oracle \
--datasource-oracle-service=ORCL \
--oracle-extractor-method=redo
```

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.

- [--hosts=host1 \[347\]](#)

Set the configuration to apply only to this host.

- [--datasource-type=oracle \[340\]](#)

Specifies the source database type, in this case, Oracle.

- [--datasource-oracle-service=ORCL \[338\]](#)

The Oracle service name. This must match the name of the Oracle service as defined in `oratab`.

- [--oracle-extractor-method=redo \[356\]](#)

Specifies the extraction method to be used with Oracle. The `redo` method uses the VMware Redo Reader component.

6. Configure the applier to the MySQL database:

```
shell-staging> ./tools/tpm configure oracle2mysql \
--hosts=host2 \
--datasource-type=mysql \
--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.

- [--master=host2 \[351\]](#)

Set the configuration to apply only to this host.

- [--datasource-type=mysql \[340\]](#)

Specifies the target database type as MySQL.

- [--user=tungsten \[370\]](#)

Configure the operating system that will execute the replicator.

7. Once the pre-requisites and configuring of the installation has been completed, the software can be installed:

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

8. Install the VMware Redo Reader on host1 by running the installation from the installed Tungsten Replication directory:

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> tpm install-vmware-redo-reader oracle2mysql \
--oracle-redo-tablespace=TUNGSTEN_TS \
--oracle-sys-user-password=secret \
--oracle-system-user-password=secret \
--oracle-redo-replicate-tables=DB1,DB2,DB3
```

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.

- [--oracle-redo-tablespace=TUNGSTEN_TS \[357\]](#)

Defines the tablespace name where the required user will be created for extracting data.

- [--oracle-redo-replicate-tables=SCHEMA\[.TABLE\]\[,...\] \[357\]](#)

Specify the schemas and optionally the tables that will be extracted from the source database.

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the [TUNGSTEN](#) schema, and so it does NOT need to be added to the [--oracle-redo-replicate-tables \[357\]](#) configuration option.

The [TUNGSTEN](#) schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- [--oracle-sys-user-password \[358\]](#)

The password of the Oracle [sys](#) user. This is required so that the required table configuration can take place.

- [--oracle-system-user-password \[358\]](#)

The password of the Oracle [SYSTEM](#) user. This is required so that the required table configuration can take place.

9. Start all replication services on host1:

```
shell-host1> startall
```

Instead of providing the password on the command line, you may put the password into a file and provide the path of that file as the argument value. If you use this method, the password should be the only content in the file.

If the Oracle values are not correct, the `tpm` command may fail with an error like "There was an error while processing the VMware Redo Reader response file". Look at the `/tmp/tungsten-configure.log` file for more information about the root cause.

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> trepctl status
shell-host1> vmrdrd_oracle2mysql status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> trepctl status
```

5.4.2.2. Installing an Oracle to MySQL Topology (INI Use Case)

Important

The VMware Redo Reader feature is only available in a specific commercial release of the Continuent Replicator. Please ensure you have downloaded the correct build (must begin with `vmware-continuent-replication-oracle-source-`), otherwise these procedures will fail.

5.4.2.2.1. Installing Oracle to MySQL Replication - Basic Procedure using INI Files

This procedure will install Oracle to MySQL replication (including the VMware Redo Reader) on a per-host basis using INI files. Repeat these steps on all member hosts.

This method requires including Oracle system passwords in the `tpm` configuration for the `sys` and `SYSTEM` Oracle users. If you are not comfortable with that, see [Section 5.4.2.2.2, "Installing Oracle to MySQL Replication - Advanced Procedure using INI Files"](#).

The `tpm install` command will:

- Validate the Replicator and VMware Redo Reader
- Configure the VMware Redo Reader
- Configure the Replicator
- Start the VMware Redo Reader
- Start the Replicator

To install Tungsten Replication follow these steps:

1. Install the Tungsten Replicator™ package (`.rpm`), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell-staging> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. On the Oracle source primary host1, create `/etc/tungsten/tungsten.ini` with the following configuration:

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

[oracle2mysql]
datasource-type=oracle
datasource-oracle-service=ORCL
enable-heterogeneous-service=true
install-vmware-redo-reader=true
master=host1
members=host1,host2
oracle-redo-tablespace=TUNGSTEN_TS
oracle-redo-replicate-tables=DB1,DB2,DB3
oracle-sys-user-password=secret
oracle-system-user-password=secret
oracle-extractor-method=redo
```

On the MySQL target secondary host2, create `/etc/tungsten/tungsten.ini` with the following configuration:

```
[defaults]
install-directory=/opt/continuent
profile-script=~/bash_profile
replication-user=tungsten
replication-password=secret
```

```
start-and-report=true
user=tungsten

[oracle2mysql]
datasource-type=mysql
master=host1
members=host1,host2
```

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 \[347\]](#)

Directory where Tungsten Replication will be installed.

- [profile-script=~/.bash_profile \[361\]](#)

Tells **tpm** to add PATH information to the script to initialize the environment.

- [replication-user=tungsten \[363\]](#)

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

- [replication-password=secret \[363\]](#)

The password that will be used for the replication service when connecting to the datasource.

- [start-and-report=true \[365\]](#)

Once the service has been installed, start it automatically, and the report on the status.

- [user=tungsten \[370\]](#)

The operating system user that the service will be executed using.

- [master=host1 \[351\]](#)

Specifies which host will be the primary (master).

- [members=host1,host2 \[351\]](#)

The members of the service, in this case, the primary (master) ([host1](#)) and the secondary (slave) ([host2](#)).

- [enable-heterogeneous-service=true \[344\]](#)

When configuring heterogeneous services it can be easier to configure them without security controls, as the certificates will need to be created centrally and applied on both servers during installation. For more information, see [Section 7.5, "Deployment Security"](#).

- [datasource-type=mysql \[340\]](#)

Specifies the source database type for the secondary (slave), in this case, MySQL.

- [datasource-type=oracle \[340\]](#)

Specifies the source database type for the primary (master), in this case, Oracle.

- [datasource-oracle-service=ORCL \[338\]](#)

The Oracle service name. This must match the name of the Oracle service as defined in [oratab](#).

- [install-vmware-redo-reader=true \[348\]](#)

Enables automatic installation of the VMware Redo Reader to extract information from the Oracle Redo logs.

- [oracle-redo-tablespace=TUNGSTEN_TS \[357\]](#)

Defines the tablespace name where the required user will be created for extracting data.

- [oracle-redo-replicate-tables=SCHEMA\[.TABLE \]\[,... \] \[357\]](#)

Specify the schemas and optionally the tables that will be extracted from the source database.⁸⁹

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the [TUNGSTEN](#) schema, and so it does NOT need to be added to the [oracle-redo-replicate-tables](#) [357] configuration option.

The [TUNGSTEN](#) schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- [oracle-sys-user-password](#) [358]

The password of the Oracle [sys](#) user. This is required so that the required table configuration can take place.

- [oracle-system-user-password](#) [358]

The password of the Oracle [SYSTEM](#) user. This is required so that the required table configuration can take place.

- [oracle-extractor-method=redo](#) [356]

Specifies the extraction method to be used with Oracle. The [redo](#) method uses the VMware Redo Reader component.

3. Configure security. Use the procedure in section [Section 7.5, “Deployment Security”](#) to prepare and deploy the needed certificates, then edit the [/etc/tungsten/tungsten.ini](#) accordingly.

To proceed without using the secure installation mode, add the following to [/etc/tungsten/tungsten.ini](#) on all member hosts:

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

4. Run [tpm](#) from the extracted software staging directory on all member hosts to install the software using the INI-based configuration:

```
shell> cd vmware-continuent-replication-oracle-source-5.0.1-136
shell> ./tools/tpm install
```

During the startup and installation, [tpm](#) will notify you of any problems that need to be fixed before the service can be correctly installed and started. If [start-and-report](#) [365] is set and the service starts correctly, you should see the configuration and current status of the service.

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> trepctl status
shell-host1> vmrdd_oracle2mysql status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> trepctl status
```

5.4.2.2.2. Installing Oracle to MySQL Replication - Advanced Procedure using INI Files

This procedure will install Oracle to MySQL replication on a per-host basis using INI files. Repeat these steps on all member hosts.

The VMware Redo Reader will be installed separately on a per-host basis so that Oracle system passwords are not included in the [tpm](#) configuration for security reasons.

The [tpm](#) install command will:

- Validate the Replicator and VMware Redo Reader
- Configure the Replicator

To install Tungsten Replication follow these steps:

1. Install the Tungsten Replicator™ package ([.rpm](#)), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell-staging> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. On the Oracle source primary host1, create [/etc/tungsten/tungsten.ini](#) with the following configuration:

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

```

replication-user=tungsten
replication-password=secret
start-and-report=true
user=tungsten

[oracle2mysql]
datasource-type=oracle
datasource-oracle-service=ORCL
enable-heterogeneous-service=true
install-vmware-redo-reader=false
master=host1
members=host1,host2
oracle-extractor-method=redo

```

On the MySQL target secondary host2, create `/etc/tungsten/tungsten.ini` with the following configuration:

```

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

[oracle2mysql]
datasource-type=mysql
master=host1
members=host1,host2

```

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` [347]

Directory where Tungsten Replication will be installed.

- `profile-script=~/.bash_profile` [361]

Tells `tpm` to add PATH information to the script to initialize the environment.

- `replication-user=tungsten` [363]

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

- `replication-password=secret` [363]

The password that will be used for the replication service when connecting to the datasource.

- `start-and-report=true` [365]

Once the service has been installed, start it automatically, and the report on the status.

- `user=tungsten` [370]

The operating system user that the service will be executed using.

- `master=host1` [351]

Specifies which host will be the primary (master).

- `members=host1,host2` [351]

The members of the service, in this case, the primary (master) (`host1`) and the secondary (slave) (`host2`).

- `enable-heterogeneous-service=true` [344]

When configuring heterogeneous services it can be easier to configure them without security controls, as the certificates will need to be created centrally and applied on both servers during installation. For more information, see [Section 7.5, “Deployment Security”](#).

- `datasource-type=mysql` [340]

Specifies the source database type for the secondary (slave), in this case, MySQL.

- `datasource-type=oracle` [340]

Specifies the source database type for the primary (master), in this case, Oracle.

- `datasource-oracle-service=ORCL` [338]

The Oracle service name. This must match the name of the Oracle service as defined in `oratab`.

- `install-vmware-redo-reader=false` [348]

Disables automatic installation of the VMware Redo Reader which extracts information from the Oracle Redo logs.

- `oracle-extractor-method=redo` [356]

Specifies the extraction method to be used with Oracle. The `redo` method uses the VMware Redo Reader component.

3. Configure security. Use the procedure in section [Section 7.5, “Deployment Security”](#) to prepare and deploy the needed certificates, then edit the `/etc/tungsten/tungsten.ini` accordingly.

To proceed without using the secure installation mode, add the following to `/etc/tungsten/tungsten.ini` on all member hosts:

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

4. Run `tpm` from the extracted software staging directory on all member hosts to install the software using the INI-based configuration:

```
shell> cd vmware-continuent-replication-oracle-source-5.0.1-136
shell> ./tools/tpm install
```

During the startup and installation, `tpm` will notify you of any problems that need to be fixed before the service can be correctly installed and started. If `start-and-report` [365] is set and the service starts correctly, you should see the configuration and current status of the service.

5. Install the VMware Redo Reader on host1 by running the installation from the installed Tungsten Replication directory:

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> tpm install-vmware-redo-reader oracle2mysql \
--oracle-redo-tablespace=TUNGSTEN_TS \
--oracle-sys-user-password=password \
--oracle-system-user-password=password \
--oracle-redo-replicate-tables=DB1,DB2,DB3
```

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.

- `--oracle-redo-tablespace=TUNGSTEN_TS` [357]

Defines the tablespace name where the required user will be created for extracting data.

- `--oracle-redo-replicate-tables=SCHEMA[.TABLE][,...]` [357]

Specify the schemas and optionally the tables that will be extracted from the source database.

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the `TUNGSTEN` schema, and so it does NOT need to be added to the `--oracle-redo-replicate-tables` [357] configuration option.

The `TUNGSTEN` schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- `--oracle-sys-user-password` [358]

The password of the Oracle `sys` user. This is required so that the required table configuration can take place.

- `--oracle-system-user-password` [358]

The password of the Oracle `SYSTEM` user. This is required so that the required table configuration can take place.

6. Start all replication services on host1:

```
shell-host1> startall
```

Instead of providing the password on the command line, you may put the password into a file and provide the path of that file as the argument value. If you use this method, the password should be the only content in the file.

If the Oracle values are not correct, the tpm command may fail with an error like "There was an error while processing the VMware Redo Reader response file". Look at the [/tmp/tungsten-configure.log](#) file for more information about the root cause.

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> trepctl status
shell-host1> vmrdd oracle2mysql status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> trepctl status
```

5.4.3. Best Practices: Oracle to MySQL Replication

Follow the guidelines in Section 2.2, "Best Practices".

5.4.3.1. Management and Monitoring of Oracle to MySQL Deployments

Check replication status on host1

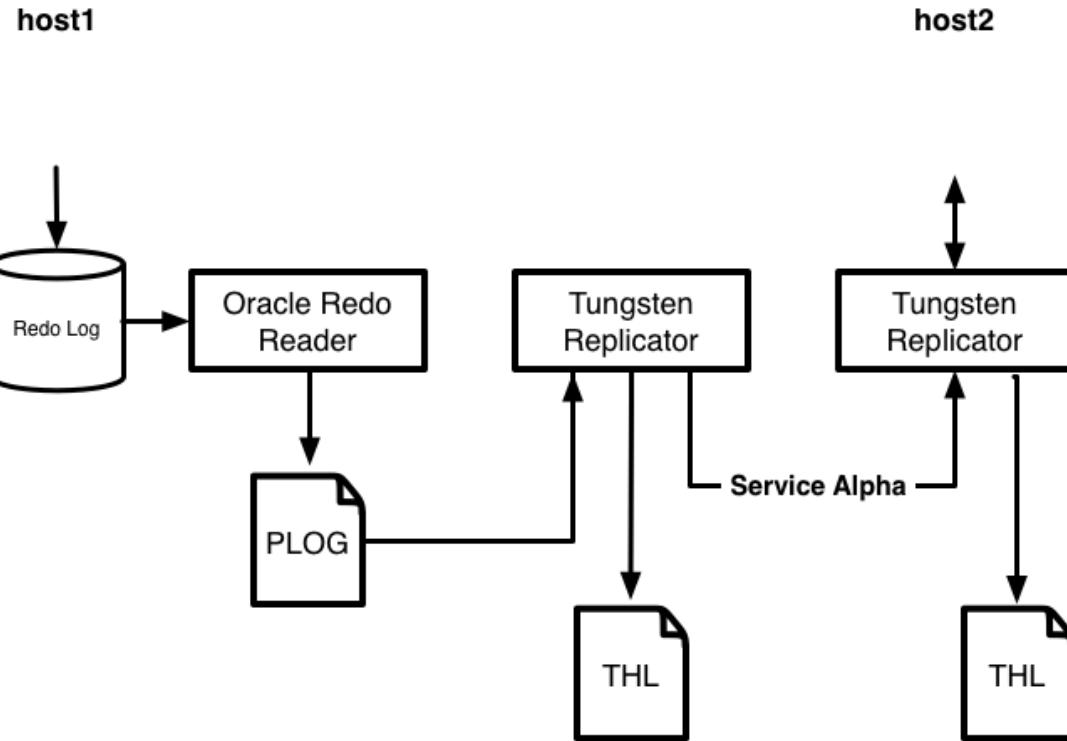
```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> trepctl status
shell-host1> vmrrd oracle2mysql status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> trepctl status
```

5.5. Deploying Oracle to Hadoop Replication

Figure 5.4. Topologies: Oracle to Hadoop with Redo Reader



5.5.1. Prepare: Oracle to Hadoop Replication

Prepare the Oracle environment as documented in section [Section 5.1.2, “Preparing the Oracle Environment for Replication”](#).

5.5.2. Install: Oracle to Hadoop Replication

An Oracle to Hadoop deployment can be installed using the following methods.

- **Staging Configuration** — [Section 5.5.2.1, “Installing an Oracle to Hadoop Topology \(Staging Use Case\)”](#)
- **INI Configuration** — [Section 5.5.2.2, “Installing an Oracle to Hadoop Topology \(INI Use Case\)”](#)

5.5.2.1. Installing an Oracle to Hadoop Topology (Staging Use Case)

Important

The VMware Redo Reader feature is only available in a specific commercial release of the Continuent Replicator. Please ensure you have downloaded the correct build (must begin with `vmware-continuent-replication-ora-
cle-source-`), otherwise these procedures will fail.

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

```
shell> cd /opt/continuent/software
```

```
shell> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Change to the Tungsten Replicator directory:

```
shell> cd vmware-continuent-replication-oracle-source-5.0.1-136
```

3. To create a master/slave using **tpm**:

```
shell> ./tools/tpm install alpha \
    --topology=master-slave \
    --master=host1 \
    --replication-user=tungsten \
    --replication-password=password \
    --install-directory=/opt/continuent \
    --members=host1,host2,host3 \
    --start
```

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

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

- **tpm install**

Executes **tpm** in **install** mode to create the service **alpha**.

- **--master=host1 [351]**

Specifies which host will be the master.

- **--replication-user=tungsten [363]**

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

- **--replication-password=password [363]**

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

- **--install-directory=/opt/continuent [347]**

Directory where Tungsten Replicator will be installed.

- **--members=host1,host2,host3 [351]**

List of all the hosts within the cluster, including the master host. Hosts in this list that do not appear in the **--master [351]** option will be configured as slaves.

- **--start [365]**

Starts the service once installation is complete.

If the MySQL configuration file cannot be located, the **--datasource-mysql-conf [337]** option can be used to specify its location:

```
shell> ./tools/tpm install alpha \
    --topology=master-slave \
    --master=host1 \
    --replication-user=tungsten \
    --replication-password=password \
    --datasource-mysql-conf=/etc/mysql/my.cnf \
    --install-directory=/opt/continuent \
    --members=host1,host2,host3 \
    --start-and-report
```

Once the installation has been completed, the service will be started and ready to use. For information on checking the running service, see [Section 5.5.3.1, "Management and Monitoring of Oracle to Hadoop Deployments"](#).

For information on starting and stopping Tungsten Replication see [Section 2.6, "Starting and Stopping Tungsten Replicator"](#); configuring init scripts to startup and shutdown when the system boots and shuts down, see [Section 2.7, "Configuring Startup on Boot"](#).

5.5.2.2. Installing an Oracle to Hadoop Topology (INI Use Case)

Important

The VMware Redo Reader feature is only available in a specific commercial release of the Continuent Replicator. Please ensure you have downloaded the correct build (must begin with `vmware-continuent-replication-oracle-source-`), otherwise these procedures will fail.

1. On all member hosts, install the Tungsten Replicator™ package ([.rpm](#)), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Change to the Tungsten Replicator directory:

```
shell> cd vmware-continuent-replication-oracle-source-5.0.1-136
```

3. Create `/etc/tungsten/tungsten.ini` with the following configuration:

```
[alpha]
user=tungsten
topology=master-slave
install-directory=/opt/continuent
master=host1
members=host1,host2,host3
replication-user=tungsten
replication-password=password
replication-port=3306
datasource-mysql-conf=/etc/mysql/my.cnf
start-and-report=true
profile-script=~/.bash_profile
```

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.

- `[alpha]`

`alpha` is the name and identity of the dataservice being created.

- `user=tungsten` [\[370\]](#)

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

- `topology=master-slave` [\[369\]](#)

The desired replicator installation topology. In this case designate the Master/Slave architecture.

- `install-directory=/opt/continuent` [\[347\]](#)

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

- `master=host1` [\[351\]](#)

Specifies which host will be the master.

- `members=host1,host2,host3` [\[351\]](#)

List of all the hosts within the cluster, including the master host. Hosts in this list that do not appear in the `master` [\[351\]](#) option will be configured as slaves.

- `replication-user=tungsten` [\[363\]](#)

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

- `replication-password=password` [\[363\]](#)

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

- `replication-port=3306` [\[363\]](#)

The port that the slave databases are listening on.

- `datasource-mysql-conf=/etc/mysql/my.cnf` [\[337\]](#)

Specify the exact location of the MySQL configuration file

- `start-and-report=true` [\[365\]](#)

Starts the service once installation is complete and outputs status.

Tells `tpm` to add PATH information to the script to initialize the environment.

4. Run `tpm` on all member hosts to install the software with the INI-based 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 `start-and-report` [365] is set and the service starts correctly, you should see the configuration and current status of the service.

5. Initialize your `PATH` and environment.

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

Once the installation has been completed, the service will be started and ready to use. For information on checking the running service, see [Section 5.5.3.1, “Management and Monitoring of Oracle to Hadoop Deployments”](#).

For information on starting and stopping Tungsten Replicator see [Section 2.6, “Starting and Stopping Tungsten Replicator”](#); configuring init scripts to startup and shutdown when the system boots and shuts down, see [Section 2.7, “Configuring Startup on Boot”](#).

5.5.3. Best Practices: Oracle to Hadoop Replication

Follow the guidelines in [Section 2.2, “Best Practices”](#).

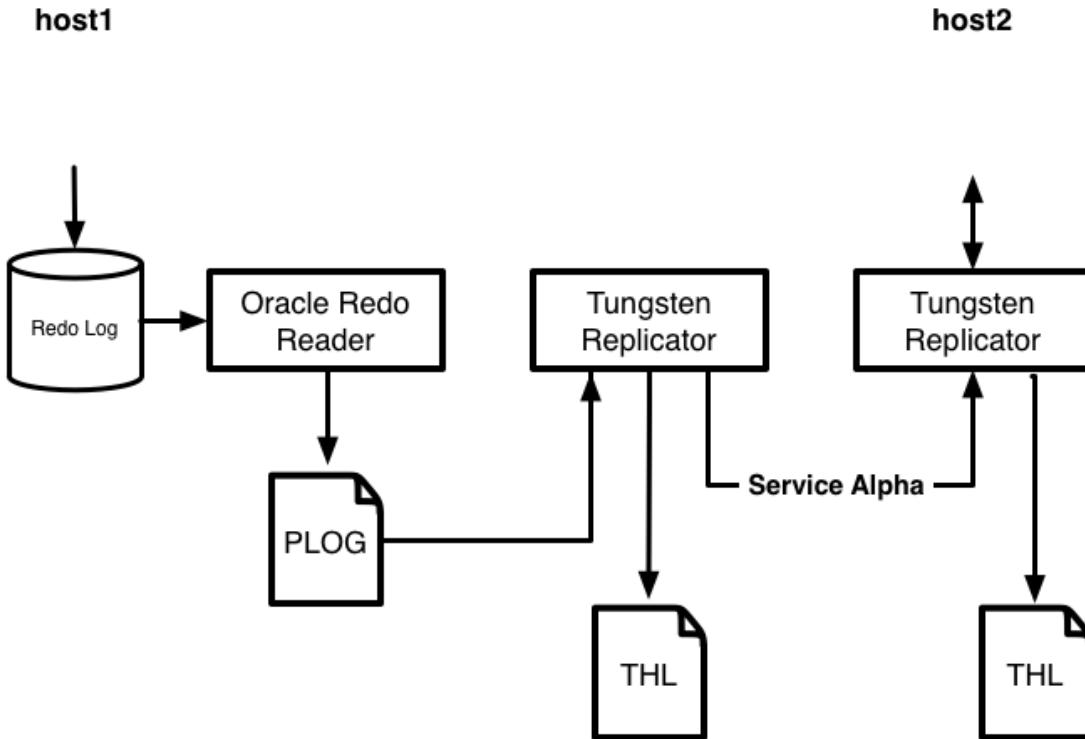
5.5.3.1. Management and Monitoring of Oracle to Hadoop Deployments

To check the configured services use the `services` parameter to `trepctl`:

5.6. Deploying Oracle to Vertica Replication

Hewlett-Packard's Vertica provides support for BigData, SQL-based analysis and processing. Integration with Oracle enables data to be replicated live from the Oracle database directly into Vertica without the need to manually export and import the data.

Figure 5.5. Topologies: Oracle to HP Vertica with Redo Reader



Replication to Vertica operates as follows:

- Data is extracted from the source database into THL.
- When extracting the data from the THL, the Vertica replicator writes the data into CSV files according to the name of the source tables. The files contain all of the row-based data, including the global transaction ID generated by Tungsten Replication during replication, and the operation type (insert, delete, etc) as part of the CSV data.
- The CSV data is then loaded into Vertica into staging tables.
- SQL statements are then executed to perform updates on the live version of the tables, using the CSV, batch loaded, information, deleting old rows, and inserting the new data when performing updates to work effectively within the confines of Vertica operation.

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

5.6.1. Prepare: Oracle to Vertica Replication

Oracle Preparation

Prepare the Oracle environment as documented in section [Section 5.1.2, “Preparing the Oracle Environment for Replication”](#).

Vertica Preparation

On the Vertica host, you need to perform some preparation of the destination database, first creating the database, and then creating the tables that are to be replicated.

- Create a database (if you want to use a different one than those already configured), and a schema that will contain the Tungsten data about the current replication position:

```
shell> vsql -Udbadmin -wsecret bigdata
Welcome to vsql, the Vertica Analytic Database v5.1.1-0 interactive terminal.

Type:  \h for help with SQL commands
      \? for help with vsql commands
      \g or terminate with semicolon to execute query
      \q to quit

bigdata=> create schema tungsten_oracle2vertica;
```

The schema will be used only by Tungsten Replication to store metadata about the replication process.

- Locate the Vertica JDBC driver. This can be downloaded separately from the Vertica website. The driver will need to be copied into the Tungsten Replication `lib` directory.

```
shell> cp vertica-jdbc-7.1.2-0.jar tungsten-replicator-5.0.1-136/tungsten-replicator/lib/
```

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

Please be sure to identify the name and location of the Oracle driver file `ojdbc6.jar` or `ojdbc7.jar` and define the `CLASSPATH` environment variable to point to that file.

To use `ddlscan`, the template for Vertica must be specified, along with the user/password information to connect to the source database to collect the schema definitions. The tool should be run from the templates directory.

The tool will need to be executed twice, the first time generates the live table definitions:

```
shell> export CP=/path/to/ojdbc6.jar
shell> cd tungsten-replicator-5.0.1-136
shell> cd tungsten-replicator/support/ddlscan/
shell> ../../bin/ddlscan -user tungsten -url 'jdbc:oracle:thin://host1:1521/ORCL' -pass secret -template ddl-oracle-vertica.vm -db DB1
/*
SQL generated on Fri Dec 04 07:26:48 EST 2015 by ./ddlscan utility of Tungsten

url = jdbc:oracle:thin:@//localhost:1521/ORCL
user = tungsten
dbName = DB1
*/
CREATE SCHEMA "DB1";

DROP TABLE "DB1"."EMPLOYEES";

CREATE TABLE "DB1"."EMPLOYEES"
(
    "EMP_NO" NUMBER ,
    "BIRTH_DATE" DATE ,
    "FIRST_NAME" VARCHAR(14) ,
    "LAST_NAME" VARCHAR(16) ,
    "GENDER" CHAR(1) ,
    "HIRE_DATE" DATE )
ORDER BY "EMP_NO"
;
...
```

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

```
shell> ../../bin/ddlscan -user tungsten -url 'jdbc:oracle:thin://host1:1521/ORCL' -pass secret -template ddl-oracle-vertica.vm -db DB1 > db1.ddl
```

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

The file can then be applied to Vertica:

```
shell> cat db1.ddl | vsql -Udbadmin -wsecret bigdata
```

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

```
shell> ../../bin/ddlscan -user tungsten -url 'jdbc:oracle:thin://host1:1521/ORCL' -pass secret -template ddl-oracle-vertica-staging.vm -db DB1 > db1-staging.ddl
```

Then applied to Vertica:

```
shell> cat db1-staging.ddl | vsql -Udbadmin -wsecret bigdata
```

Important

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

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

5.6.2. Install: Oracle to Vertica Replication

An Oracle to Vertica deployment can be installed using the following methods.

- **Staging Configuration** — [Section 5.6.2.1, “Installing Oracle to Vertica Replication \(Staging Method\)”](#)
- **INI Configuration** — [Section 5.6.2.2, “Installing an Oracle to Vertica \(INI Use Case\)”](#)

5.6.2.1. Installing Oracle to Vertica Replication (Staging Method)

Important

The VMware Redo Reader feature is only available in a specific commercial release of the Continuent Replicator. Please ensure you have downloaded the correct build (must begin with `vmware-continuent-replication-oracle-source-`), otherwise these procedures will fail.

5.6.2.1.1. Installing Oracle to Vertica Replication - Basic Procedure via SSH

This procedure will install Oracle to Vertica replication (including the VMware Redo Reader) to all member hosts via SSH from a single staging directory on a specified host.

This method requires including Oracle system passwords in the `tpm` configuration for the `sys` and `SYSTEM` Oracle users. If you are not comfortable with that, see [Section 5.6.2.1.2, “Installing Oracle to Vertica Replication - Advanced Procedure via SSH”](#).

The `tpm install` command will:

- Validate the Replicator and VMware Redo Reader
- Configure the VMware Redo Reader
- Configure the Replicator
- Start the VMware Redo Reader
- Start the Replicator

To install Tungsten Replication follow these steps:

1. Install the Tungsten Replicator™ package (`.rpm`), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell-staging> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Change to the Tungsten Replicator staging directory:

```
shell-staging> cd vmware-continuent-replication-oracle-source-5.0.1-136
```

3. Locate the Vertica JDBC driver. This can be downloaded separately from the Vertica website. The driver will need to be copied into the Tungsten Replication `lib` directory.

```
shell> cp vertica-jdbc-7.1.2-0.jar tungsten-replicator-5.0.1-136/tungsten-replicator/lib/
```

4. Configure the staging directory with the desired defaults:

```
shell-staging> ./tools/tpm configure defaults --reset \
--disable-relay-logs=true \
--enable-heterogeneous-service=true \
--install-directory=/opt/continuent \
--skip-validation-check=HostsFileCheck \
--start-and-report=true \
--user=tungsten
```

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

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

- **tpm configure defaults**

Executes **tpm** in **configure** mode to instantiate the replicator **defaults**. This sets up the configuration information without performing an installation, enabling further configuration to be applied.

- **--disable-relay-logs=true [342]**

Disable the relay logs.

- **--enable-heterogeneous-service=true [344]**

Set certain parameters that ensure that a heterogeneous deployment operates correctly, including setting files, Java file types, and other settings.

- **--install-directory=/opt/continuent [347]**

Directory where Tungsten Replication will be installed.

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

Disable checking the hosts during deployment

- **--start-and-report=true [365]**

Once the service has been installed, start it automatically, and the report on the status.

- **--user=tungsten [370]**

The operating system user that the service will be executed using.

5. Configure the oracle2vertica service replication topology:

```
shell-staging> ./tools/tpm configure oracle2vertica \
--master=host1 \
--members=host1,host2
```

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 oracle2vertica**

Executes **tpm** in **configure** mode to create the service **oracle2vertica**.

- **--master=host1 [351]**

Specifies which host will be the master.

- **--members=host1,host2 [351]**

Specifies the members; **host1** is the master, **host2** is the Vertica host.

6. Configure the replicator on the source Oracle primary for extracting data:

```
shell-staging> ./tools/tpm configure oracle2vertica \
--hosts=host1 \
--enable-heterogeneous-master=true \
--replication-user=tungsten \
--replication-password=secret \
--datasource-type=oracle \
--datasource-oracle-service=ORCL \
--install-vmware-redo-reader=true \
--oracle-redo-replicate-tables=DB1,DB2,DB3 \
--oracle-redo-tablespace=TUNGSTEN_TS \
--oracle-sys-user-password=secret \
--oracle-system-user-password=secret \
--oracle-extractor-method=redo
```

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 oracle2vertica**

Executes `tpm` in `configure` mode to modify the service `oracle2vertica`.

- `--hosts=host1 [347]`

By explicitly stating the list of hosts, only the configuration for the hosts listed will be updated. In this case, the configuration for the primary, `host1` is being updated.

- `--enable-heterogeneous-master=true [344]`

Ensures the correct file encoding and filters are applied to the master configuration for a heterogeneous deployment.

- `--replication-user=tungsten [363]`

The user name that will be used to apply replication updates the database.

- `--replication-password=secret [363]`

The password that will be used for the replication service when connecting to the datasource.

- `--datasource-type=oracle [340]`

Specifies the source database type, in this case, Oracle.

- `--datasource-oracle-service=ORCL [338]`

The Oracle service name. This must match the name of the Oracle service as defined in `oratab`.

- `--install-vmware-redo-reader=true [348]`

Enables automatic installation of the VMware Redo Reader to extract information from the Oracle Redo logs.

- `--oracle-redo-replicate-tables=SCHEMA[.TABLE][,...] [357]`

Specify the schemas and optionally the tables that will be extracted from the source database.

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the `TUNGSTEN` schema, and so it does NOT need to be added to the `--oracle-redo-replicate-tables [357]` configuration option.

The `TUNGSTEN` schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- `--oracle-redo-tablespace=TUNGSTEN_TS [357]`

Defines the tablespace name where the required user will be created for extracting data.

- `--oracle-sys-user-password [358]`

The password of the Oracle `sys` user. This is required so that the required table configuration can take place.

- `--oracle-system-user-password [358]`

The password of the Oracle `SYSTEM` user. This is required so that the required table configuration can take place.

- `--oracle-extractor-method=redo [356]`

Specifies the extraction method to be used with Oracle. The `redo` method uses the VMware Redo Reader component.

7. Configure the parameters for the replicator secondary host that will apply the events to Vertica:

```
shell> ./tools/tpm configure oracle2vertica \
    --hosts=host2 \
    --replication-user=dbadmin \
    --replication-password=secret \
    --batch-load-language=js \
    --batch-load-template=vertica \
    --datasource-type=vertica \
    --vertica-dbname=bigdata \
    --replication-host=host2 \
```

```
--replication-port=5433 \
--skip-validation-check=InstallerMasterSlaveCheck \
--svc-applier-block-commit-size=25000 \
--svc-applier-block-commit-interval=30s
```

This configures the Vertica replication secondary to accept transaction data from the primary.

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 oracle2vertica**

Executes **tpm** in **configure** mode to modify the service **oracle2vertica**.

- **--hosts=host2 [347]**

By explicitly stating the list of hosts, only the configuration for the hosts listed will be updated. In this case, the configuration for the master, **host2** is being updated.

- **--replication-user=dbadmin [363]**

Set the user for connecting to the Vertica database service.

- **--replication-password=secret [363]**

Set the password used to connect to the Vertica database service.

- **--batch-enabled=true [331]**

The Vertica applier uses the Tungsten Replication batch loading system to generate the load data imported.

- **--batch-load-language=js [331]**

The batch load language should be JavaScript, which supports the required scripts and loading processes.

- **--batch-load-template=vertica [331]**

The batch load templates configure how the batch load operation operates. These templates perform the necessary steps to load the generated CSV file, and execute the SQL statement that migrate the data from the seed tables.

- **--datasource-type=vertica [340]**

Specifies the datasource type, in this case Vertica. This ensures that the correct applier is being used to apply transactions in the target database.

- **--vertica-dbname=bigdata [370]**

Set the database name to be used when applying data to the Vertica database.

- **--replication-port=5433 [363]**

Set the port number to use when connecting to the Vertica database service.

- **--replication-host=host2 [363]**

Specify the host name of the secondary, in this case the Vertica database server, **host2**.

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

Skip the test for the master/slave check; this does not apply in a heterogeneous deployment.

- **--svc-applier-block-commit-size=25000 [366]**

Set the block commit size to 25,000 events. Because Vertica uses the batch applier, the commit process can submit a larger number of transactions simultaneously to Vertica.

For more information, see [Section 12.1, “Block Commit”](#).

- **--svc-applier-block-commit-interval=30s [366]**

Set the maximum interval between commits, regardless of the transaction count, to 3s.

For more information, see [Section 12.1, “Block Commit”](#).

8. Once the pre-requisites and configuring of the installation has been completed, the software can be installed:

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

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> trepcctl status
shell-host1> vmrrd_oracle2vertica status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> trepcctl status
```

5.6.2.1.2. Installing Oracle to Vertica Replication - Advanced Procedure via SSH

This procedure will install Oracle to Vertica replication (excluding the VMware Redo Reader) to all member hosts via SSH from a single staging directory on a specified host.

The VMware Redo Reader will be installed separately on a per-host basis so that Oracle system passwords are not included in the `tpm` configuration for security reasons.

The `tpm` install command will:

- Validate the Replicator and VMware Redo Reader
- Configure the Replicator

To install Tungsten Replication follow these steps:

1. Install the Tungsten Replicator™ package (`.rpm`), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell-staging> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Change to the Tungsten Replicator staging directory:

```
shell-staging> cd vmware-continuent-replication-oracle-source-5.0.1-136
```

3. Locate the Vertica JDBC driver. This can be downloaded separately from the Vertica website. The driver will need to be copied into the Tungsten Replication `lib` directory.

```
shell> cp vertica-jdbc-7.1.2-0.jar tungsten-replicator-5.0.1-136/tungsten-replicator/lib/
```

4. Configure the staging directory with the desired defaults:

```
shell-staging> ./tools/tpm configure defaults --reset \
--disable-relay-logs=true \
--enable-heterogeneous-service=true \
--install-directory=/opt/continuent \
--skip-validation-check=HostsFileCheck \
--start-and-report=true \
--user=tungsten
```

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

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

- [tpm configure defaults](#)

Executes `tpm` in `configure` mode to instantiate the replicator `defaults`. This sets up the configuration information without performing an installation, enabling further configuration to be applied.

- [--disable-relay-logs=true \[342\]](#)

Disable the relay logs.

- [--enable-heterogeneous-service=true \[344\]](#)

Set certain parameters that ensure that a heterogeneous deployment operates correctly, including setting files, Java file types, and other settings.

- `--install-directory=/opt/continuent [347]`

Directory where Tungsten Replication will be installed.

- `--skip-validation-check=HostsFileCheck`

Disable checking the hosts during deployment

- `--start-and-report=true [365]`

Once the service has been installed, start it automatically, and the report on the status.

- `--user=tungsten [370]`

The operating system user that the service will be executed using.

5. Configure the oracle2vertica service replication topology:

```
shell-staging> ./tools/tpm configure oracle2vertica \
--master=host1 \
--members=host1,host2
```

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 oracle2vertica`

Executes `tpm` in `configure` mode to create the service `oracle2vertica`.

- `--master=host1 [351]`

Specifies which host will be the master.

- `--members=host1,host2 [351]`

Specifies the members; `host1` is the master, `host2` is the Vertica host.

6. Configure the replicator on the source Oracle primary for extracting data:

```
shell-staging> ./tools/tpm configure oracle2vertica \
--hosts=host1 \
--enable-heterogeneous-master=true \
--replication-user=tungsten \
--replication-password=secret \
--datasource-type=oracle \
--datasource-oracle-service=ORCL \
--install-vmware-redo-reader=false \
--oracle-extractor-method=redo
```

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 oracle2vertica`

Executes `tpm` in `configure` mode to modify the service `oracle2vertica`.

- `--hosts=host1 [347]`

Configure the settings for this host only.

- `--enable-heterogeneous-master=true [344]`

Ensures the correct file encoding and filters are applied to the master configuration for a heterogeneous deployment.

- `--replication-user=tungsten [363]`

The user name that will be used to apply replication updates the database.

- `--replication-password=secret [363]`

The password that will be used for the replication service when connecting to the datasource.

- `--datasource-type=oracle [340]`

Specifies the source database type, in this case, Oracle.

- `--datasource-oracle-service=ORCL [338]`

The Oracle service name. This must match the name of the Oracle service as defined in `oratab`.

- `--install-vmware-redo-reader=false [348]`

Disables automatic installation of the VMware Redo Reader which extracts information from the Oracle Redo logs.

- `--oracle-extractor-method=redo [356]`

Specifies the extraction method to be used with Oracle. The `redo` method uses the VMware Redo Reader component.

7. Configure the parameters for the replicator secondary host that will apply the events to Vertica:

```
shell> ./tools/tpm configure oracle2vertica \
    --hosts=host2 \
    --replication-user=dbadmin \
    --replication-password=secret \
    --batch-load-language=js \
    --batch-load-template=vertica \
    --datasource-type=vertica \
    --vertica-database=bigdata \
    --replication-host=host2 \
    --replication-port=5433 \
    --skip-validation-check=InstallerMasterSlaveCheck \
    --svc-applier-block-commit-size=25000 \
    --svc-applier-block-commit-interval=30s
```

This configures the Vertica replication secondary to accept transaction data from the primary.

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 oracle2vertica`

Executes `tpm` in `configure` mode to modify the service `oracle2vertica`.

- `--hosts=host2 [347]`

By explicitly stating the list of hosts, only the configuration for the hosts listed will be updated. In this case, the configuration for the secondary, `host2` is being updated.

- `--replication-user=dbadmin [363]`

Set the user for connecting to the Vertica database service.

- `--replication-password=secret [363]`

Set the password used to connect to the Vertica database service.

- `--batch-enabled=true [331]`

The Vertica applier uses the Tungsten Replication batch loading system to generate the load data imported.

- `--batch-load-language=js [331]`

The batch load language should be JavaScript, which supports the required scripts and loading processes.

- `--batch-load-template=vertica [331]`

The batch load templates configure how the batch load operation operates. These templates perform the necessary steps to load the generated CSV file, and execute the SQL statement that migrate the data from the seed tables.

- `--datasource-type=vertica [340]`

Specifies the datasource type, in this case Vertica. This ensures that the correct applier is being used to apply transactions in the target database.

- `--vertica-database=bigdata [370]`

Set the database name to be used when applying data to the Vertica database.

- `--replication-port=5433` [363]

Set the port number to use when connecting to the Vertica database service.

- `--replication-host=host2` [363]

Set the replication host.

- `--skip-validation-check=InstallerMasterSlaveCheck`

Skip the test for the master/slave check; this does not apply in a heterogeneous deployment.

- `--svc-applier-block-commit-size=25000` [366]

Set the block commit size to 25,000 events. Because Vertica uses the batch applier, the commit process can submit a larger number of transactions simultaneously to Vertica.

For more information, see [Section 12.1, “Block Commit”](#).

- `--svc-applier-block-commit-interval=30s` [366]

Set the maximum interval between commits, regardless of the transaction count, to 3s.

For more information, see [Section 12.1, “Block Commit”](#).

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

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

- Install the VMware Redo Reader on host1 by running the installation from the installed Tungsten Replication directory:

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> tpm install-vmware-redo-reader oracle2vertica \
--oracle-redo-replicate-tables=DB1,DB2,DB3 \
--oracle-redo-tablespace=TUNGSTEN_TS \
--oracle-sys-user-password=secret \
--oracle-system-user-password=secret
```

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.

- `--oracle-redo-replicate-tables=SCHEMA[.TABLE][,...]` [357]

Specify the schemas and optionally the tables that will be extracted from the source database.

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the `TUNGSTEN` schema, and so it does NOT need to be added to the `--oracle-redo-replicate-tables` [357] configuration option.

The `TUNGSTEN` schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- `--oracle-redo-tablespace=TUNGSTEN_TS` [357]

Defines the tablespace name where the required user will be created for extracting data.

- `--oracle-sys-user-password` [358]

The password of the Oracle `sys` user. This is required so that the required table configuration can take place.

- `--oracle-system-user-password` [358]

The password of the Oracle `SYSTEM` user. This is required so that the required table configuration can take place.

- Start all replication services on host1:

```
shell-host1> startall
```

Instead of providing the password on the command line, you may put the password into a file and provide the path of that file as the argument value. If you use this method, the password should be the only content in the file.

If the Oracle values are not correct, the `tpm` command may fail with an error like "There was an error while processing the VMware Redo Reader response file". Look at the `/tmp/tungsten-configure.log` file for more information about the root cause.

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> trepctl status
shell-host1> vmrrd_oracle2vertica status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> trepctl status
```

5.6.2.2. Installing an Oracle to Vertica (INI Use Case)

Important

The VMware Redo Reader feature is only available in a specific commercial release of the Continuent Replicator. Please ensure you have downloaded the correct build (must begin with `vmware-continuent-replication-oracle-source-`), otherwise these procedures will fail.

5.6.2.2.1. Installing Oracle to Vertica Replication - Basic Procedure using INI Files

This procedure will install Oracle to Vertica replication (including the VMware Redo Reader) on a per-host basis using INI files. Repeat these steps on all member hosts.

This method requires including Oracle system passwords in the `tpm` configuration for the `sys` and `SYSTEM` Oracle users. If you are not comfortable with that, see [Section 5.6.2.2.2, "Installing Oracle to Vertica Replication - Advanced Procedure using INI Files"](#).

The `tpm install` command will:

- Validate the Replicator and VMware Redo Reader
- Configure the VMware Redo Reader
- Configure the Replicator
- Start the VMware Redo Reader
- Start the Replicator

To install Tungsten Replication follow these steps:

1. Install the Tungsten Replicator™ package (`.rpm`), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell-staging> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Change to the Tungsten Replicator staging directory:

```
shell-staging> cd vmware-continuent-replication-oracle-source-5.0.1-136
```

3. Locate the Vertica JDBC driver. This can be downloaded separately from the Vertica website. The driver will need to be copied into the Tungsten Replication `lib` directory.

```
shell> cp vertica-jdbc-7.1.2-0.jar tungsten-replicator-5.0.1-136/tungsten-replicator/lib/
```

4. On the Oracle source primary host1, create `/etc/tungsten/tungsten.ini` with the following configuration:

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

[oracle2vertical]
datasource-type=oracle
datasource-oracle-service=ORCL
enable-heterogeneous-service=true
install-vmware-redo-reader=true
```

```

master=host1
members=host1,host2
oracle-redo-tablespace=TUNGSTEN_TS
oracle-redo-replicate-tables=DB1,DB2,DB3
oracle-sys-user-password=secret
oracle-system-user-password=secret
oracle-extractor-method=redo

```

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 \[347\]](#)

Directory where Tungsten Replication will be installed.

- [profile-script=~/.bash_profile \[361\]](#)

Tells **tpm** to add PATH information to the script to initialize the environment.

- [replication-user=tungsten \[363\]](#)

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

- [replication-password=secret \[363\]](#)

The password that will be used for the replication service when connecting to the datasource.

- [start-and-report=true \[365\]](#)

Once the service has been installed, start it automatically, and the report on the status.

- [user=tungsten \[370\]](#)

The operating system user that the service will be executed using.

- [master=host1 \[351\]](#)

Specifies which host will be the primary (master).

- [members=host1,host2 \[351\]](#)

The members of the service, in this case, the primary (master) ([host1](#)) and the secondary (slave) ([host2](#)).

- [enable-heterogeneous-service=true \[344\]](#)

When configuring heterogeneous services it can be easier to configure them without security controls, as the certificates will need to be created centrally and applied on both servers during installation. For more information, see [Section 7.5, "Deployment Security"](#).

- [datasource-type=oracle \[340\]](#)

Specifies the source database type for the primary (master), in this case, Oracle.

- [datasource-oracle-service=ORCL \[338\]](#)

The Oracle service name. This must match the name of the Oracle service as defined in [oratab](#).

- [install-vmware-redo-reader=true \[348\]](#)

Enables automatic installation of the VMware Redo Reader to extract information from the Oracle Redo logs.

- [oracle-redo-tablespace=TUNGSTEN_TS \[357\]](#)

Defines the tablespace name where the required user will be created for extracting data.

- [oracle-redo-replicate-tables=SCHEMA\[.TABLE\]\[,...\] \[357\]](#)

Specify the schemas and optionally the tables that will be extracted from the source database.

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the `TUNGSTEN` schema, and so it does NOT need to be added to the `oracle-redo-replicate-tables` [357] configuration option.

The `TUNGSTEN` schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- `oracle-sys-user-password` [358]

The password of the Oracle `sys` user. This is required so that the required table configuration can take place.

- `oracle-system-user-password` [358]

The password of the Oracle `SYSTEM` user. This is required so that the required table configuration can take place.

- `oracle-extractor-method=redo` [356]

Specifies the extraction method to be used with Oracle. The `redo` method uses the VMware Redo Reader component.

5. On the Vertica target secondary host2, create `/etc/tungsten/tungsten.ini` with the following configuration:

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

[oracle2vertica]
master=host1
members=host1,host2
replication-user=dbadmin
replication-password=secret
batch-load-language=jis
batch-load-template=vertica
datasource-type=vertica
vertica-dbname=bigdata
replication-host=host2
replication-port=5433
skip-validation-check=InstallerMasterSlaveCheck
svc-applier-block-commit-size=25000
svc-applier-block-commit-interval=30s
```

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` [347]

Directory where Tungsten Replication will be installed.

- `profile-script=~/bash_profile` [361]

Tells `tpm` to add PATH information to the script to initialize the environment.

- `replication-user=dbadmin` [363]

Set the user for connecting to the Vertica database service.

- `replication-password=secret` [363]

Set the password used to connect to the Vertica database service.

- `start-and-report=true` [365]

Once the service has been installed, start it automatically, and the report on the status.

- `user=tungsten` [370]

The operating system user that the service will be executed using.

- `master=host1` [351]

Specifies which host will be the primary (master).

- `members=host1,host2` [351]

The members of the service, in this case, the primary (master) (`host1`) and the secondary (slave) (`host2`).

- `batch-enabled=true` [331]

The Vertica applier uses the Tungsten Replication batch loading system to generate the load data imported.

- `batch-load-language=js` [331]

The batch load language should be JavaScript, which supports the required scripts and loading processes.

- `batch-load-template=vertica` [331]

The batch load templates configure how the batch load operation operates. These templates perform the necessary steps to load the generated CSV file, and execute the SQL statement that migrate the data from the seed tables.

- `datasource-type=vertica` [340]

Specifies the datasource type, in this case Vertica. This ensures that the correct applier is being used to apply transactions in the target database.

- `vertica-dbname=bigdata` [370]

Set the database name to be used when applying data to the Vertica database.

- `replication-port=5433` [363]

Set the port number to use when connecting to the Vertica database service.

- `replication-host=host2` [363]

Specify the host name of the secondary, in this case the Vertica database server, `host2`.

- `skip-validation-check=InstallerMasterSlaveCheck`

Skip the test for the master/slave check; this does not apply in a heterogeneous deployment.

- `svc-applier-block-commit-size=25000` [366]

Set the block commit size to 25,000 events. Because Vertica uses the batch applier, the commit process can submit a larger number of transactions simultaneously to Vertica.

For more information, see [Section 12.1, “Block Commit”](#).

- `svc-applier-block-commit-interval=30s` [366]

Set the maximum interval between commits, regardless of the transaction count, to 3s.

For more information, see [Section 12.1, “Block Commit”](#).

6. Configure security. Use the procedure in section [Section 7.5, “Deployment Security”](#) to prepare and deploy the needed certificates, then edit the `/etc/tungsten/tungsten.ini` accordingly.

To proceed without using the secure installation mode, add the following to `/etc/tungsten/tungsten.ini` on all member hosts:

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

7. Run `tpm` from the extracted software staging directory on all member hosts to install the software using the INI-based 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 `start-and-report` [365] is set and the service starts correctly, you should see the configuration and current status of the service.

Check replication status on host1

```
shell> host1> ./opt/continuent/share/env.sh
```

```
shell> trepctl status
shell> vmrdd_oracle2vertica status
```

Check replication status on host2

```
shell> . /opt/continuent/share/env.sh
shell> trepctl status
```

5.6.2.2.2. Installing Oracle to Vertica Replication - Advanced Procedure using INI Files

This procedure will install Oracle to Vertica replication on a per-host basis using INI files. Repeat these steps on all member hosts.

The VMware Redo Reader will be installed separately on a per-host basis so that Oracle system passwords are not included in the `tpm` configuration for security reasons.

The `tpm` install command will:

- Validate the Replicator and VMware Redo Reader
- Configure the Replicator

To install Tungsten Replication follow these steps:

1. Install the Tungsten Replicator™ package (`.rpm`), or download the compressed tarball and unpack it:

```
shell> cd /opt/continuent/software
shell> tar zxf vmware-continuent-replication-oracle-source-5.0.1-136.tar.gz
```

2. Change to the Tungsten Replicator staging directory:

```
shell> cd vmware-continuent-replication-oracle-source-5.0.1-136
```

3. Locate the Vertica JDBC driver. This can be downloaded separately from the Vertica website. The driver will need to be copied into the Tungsten Replication `lib` directory.

```
shell> cp vertica-jdbc-7.1.2-0.jar tungsten-replicator-5.0.1-136/tungsten-replicator/lib/
```

4. On the Oracle source primary host1, create `/etc/tungsten/tungsten.ini` with the following configuration:

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

[oracle2vertica]
datasource-type=oracle
datasource-oracle-service=ORCL
enable-heterogeneous-service=true
install-vmware-redo-reader=false
master=host1
members=host1,host2
oracle-extractor-method=redo
```

On the Vertica target secondary host2, create `/etc/tungsten/tungsten.ini` with the following configuration:

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

[oracle2vertica]
master=host1
members=host1,host2
replication-user=dbadmin
replication-password=secret
batch-load-language=js
batch-load-template=vertica
datasource-type=vertica
vertica-database=bigdata
replication-host=host2
replication-port=5433
skip-validation-check=InstallerMasterSlaveCheck
svc-applier-block-commit-size=25000
svc-applier-block-commit-interval=30s
```

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` [347]

Directory where Tungsten Replication will be installed.

- `profile-script=~/bash_profile` [361]

Tells `tpm` to add PATH information to the script to initialize the environment.

- `replication-user=tungsten` [363]

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

- `replication-password=secret` [363]

The password that will be used for the replication service when connecting to the datasource.

- `start-and-report=true` [365]

Once the service has been installed, start it automatically, and the report on the status.

- `user=tungsten` [370]

The operating system user that the service will be executed using.

- `master=host1` [351]

Specifies which host will be the primary (master).

- `members=host1,host2` [351]

The members of the service, in this case, the primary (master) (`host1`) and the secondary (slave) (`host2`).

- `enable-heterogeneous-service=true` [344]

When configuring heterogeneous services it can be easier to configure them without security controls, as the certificates will need to be created centrally and applied on both servers during installation. For more information, see [Section 7.5, “Deployment Security”](#).

- `datasource-type=vertica` [340]

Specifies the source database type for the secondary (slave), in this case, Vertica.

- `datasource-type=oracle` [340]

Specifies the source database type for the primary (master), in this case, Oracle.

- `datasource-oracle-service=ORCL` [338]

The Oracle service name. This must match the name of the Oracle service as defined in `oratab`.

- `install-vmware-redo-reader=false` [348]

Disables automatic installation of the VMware Redo Reader which extracts information from the Oracle Redo logs.

- `oracle-extractor-method=redo` [356]

Specifies the extraction method to be used with Oracle. The `redo` method uses the VMware Redo Reader component.

5. Configure security. Use the procedure in section [Section 7.5, “Deployment Security”](#) to prepare and deploy the needed certificates, then edit the `/etc/tungsten/tungsten.ini` accordingly.

To proceed without using the secure installation mode, add the following to `/etc/tungsten/tungsten.ini` on all member hosts:

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

6. Run `tpm` from the extracted software staging directory on all member hosts to install the software using the INI-based configuration:

```
shell> cd vmware-continuent-replication-oracle-source-5.0.1-136
```

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

During the startup and installation, `tpm` will notify you of any problems that need to be fixed before the service can be correctly installed and started. If `start-and-report` [365] is set and the service starts correctly, you should see the configuration and current status of the service.

7. Install the VMware Redo Reader on host1 by running the installation from the installed Tungsten Replication directory:

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> tpm install-vmware-redo-reader oracle2vertica \
--oracle-redo-tablespace=TUNGSTEN_TS \
--oracle-sys-user-password=password \
--oracle-system-user-password=password \
--oracle-redo-replicate-tables=DB1,DB2,DB3
```

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.

- `oracle-redo-tablespace=TUNGSTEN_TS` [357]

Defines the tablespace name where the required user will be created for extracting data.

- `oracle-redo-replicate-tables=SCHEMA[.TABLE][,...]` [357]

Specify the schemas and optionally the tables that will be extracted from the source database.

Important

The VMware Redo Reader must be configured to extract specific tablespaces from the source database. The specified schemas must already have been created on the target secondary.

The VMware Redo Reader auto-replicates the `TUNGSTEN` schema, and so it does NOT need to be added to the `oracle-redo-replicate-tables` [357] configuration option.

The `TUNGSTEN` schema is designed to be used only by the replicator, and as such, should not be user-modified in any way.

- `oracle-sys-user-password` [358]

The password of the Oracle `sys` user. This is required so that the required table configuration can take place.

- `oracle-system-user-password` [358]

The password of the Oracle `SYSTEM` user. This is required so that the required table configuration can take place.

8. Start all replication services on host1:

```
shell-host1> startall
```

Instead of providing the password on the command line, you may put the password into a file and provide the path of that file as the argument value. If you use this method, the password should be the only content in the file.

If the Oracle values are not correct, the `tpm` command may fail with an error like "There was an error while processing the VMware Redo Reader response file". Look at the `/tmp/tungsten-configure.log` file for more information about the root cause.

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> trepctl status
shell-host1> vmrdd_oracle2vertica status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> trepctl status
```

5.6.3. Best Practices: Oracle to Vertica Replication

Follow the guidelines in Section 2.2, "Best Practices".

5.6.3.1. Management and Monitoring of Oracle to Vertica Deployments

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

Check replication status on host1

```
shell-host1> . /opt/continuent/share/env.sh
shell-host1> trepctl status
shell-host1> vmrdd_oracle2vertica status
```

Check replication status on host2

```
shell-host2> . /opt/continuent/share/env.sh
shell-host2> trepctl status
```

The `appliedLastSeqno` should match as normal. Because of the batching of transactions the `appliedLatency` may be much higher than normal homogenous replication.

5.6.3.2. Troubleshooting Vertica Installations

- Remember that changes to the DDL within the source database are not automatically replicated to Vertica. Changes to the table definitions, additional tables, or additional databases, must all be updated manually within Vertica.
- If you get errors similar to:

```
stage_xxx_access_log does not exist
```

When loading into Vertica, it means that the staging tables have not created correctly. Check the steps for creating the staging tables using `ddlscan` in [Section 6.4.1, "Preparing Hosts for Vertica Deployments"](#).

- Replication may fail if date types contain zero values, which are legal in MySQL. For example, the timestamp `0000-00-00 00:00:00` is valid in MySQL. An error reporting a mismatch in the values will be reported when applying the data into Vertica, for example:

```
ERROR 2631: Column "time" is of type timestamp but expression is of type int
HINT: You will need to rewrite or cast the expression
```

Or:

```
ERROR 2992: Date/time field value out of range: "0"
HINT: Perhaps you need a different "datestyle" setting
```

To address this error, use the `zerodate2null` filter, which translates zero-value dates into a valid NULL value. This can be enabled by adding the `zerodate2null` filter to the applier stage when configuring the service using `tpm`:

```
shell> ./tools/tpm update oracle2vertica --repl-svc-applier-filters=zerodate2null
```

5.7. Deploying Oracle Replication using CDC

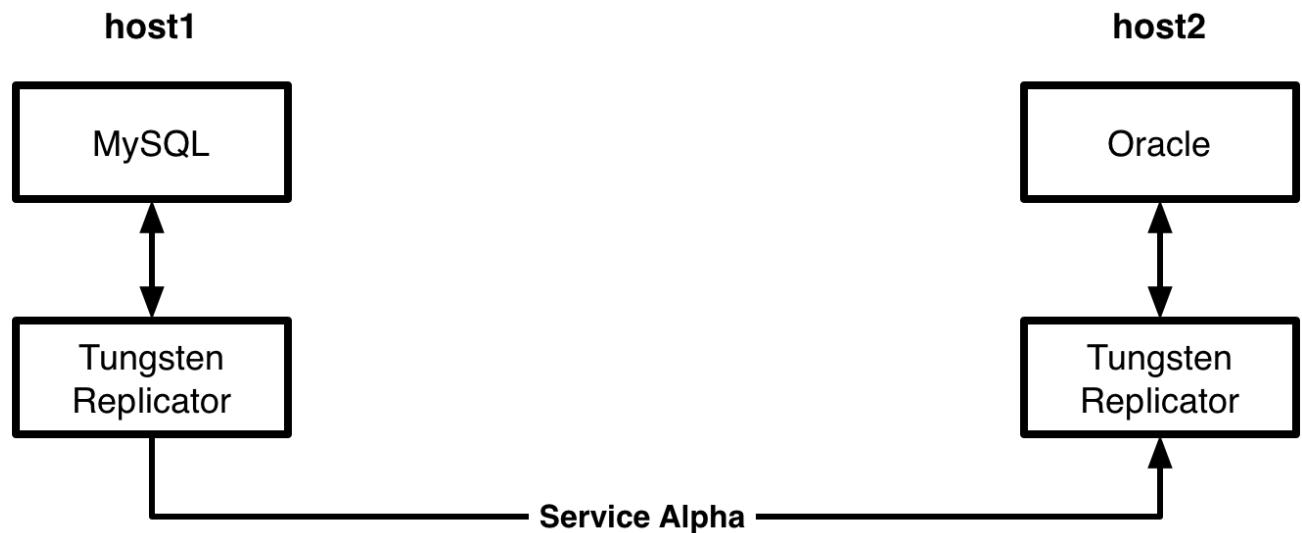
Replication Operation Support	
Statements Replicated	No
Rows Replicated	Yes
Schema Replicated	No
<code>ddlscan</code> Supported	Yes, for mixed Oracle/MySQL

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

Three variations of Oracle-based replication are officially supported:

- MySQL to Oracle

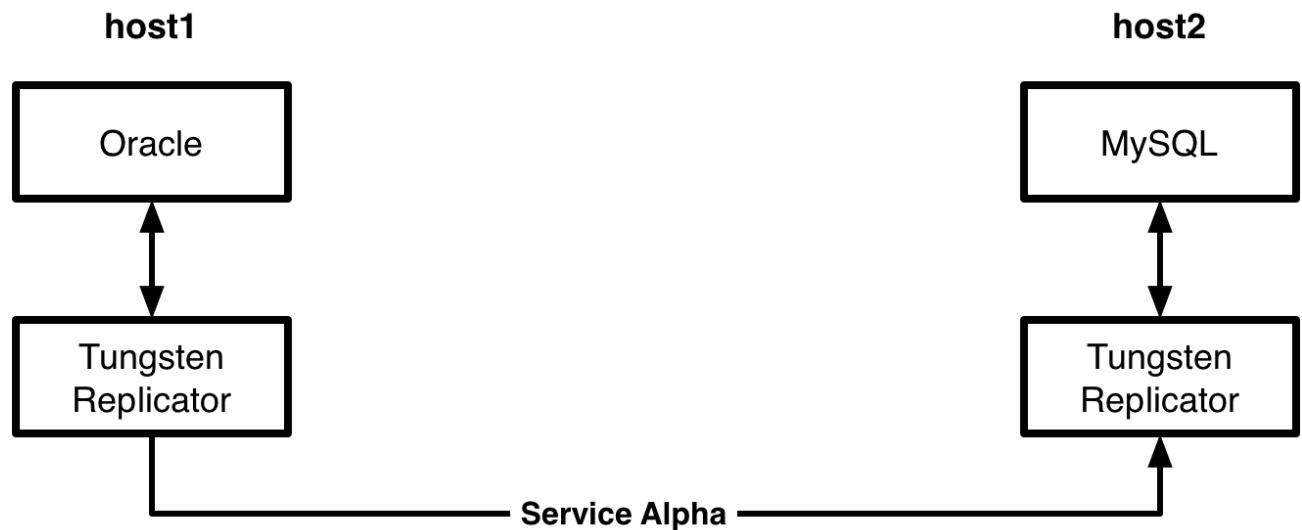
Figure 5.6. Topologies: MySQL to Oracle



For configuration, see [Section 6.1, “Deploying MySQL to Oracle Replication”](#)

- Oracle to MySQL

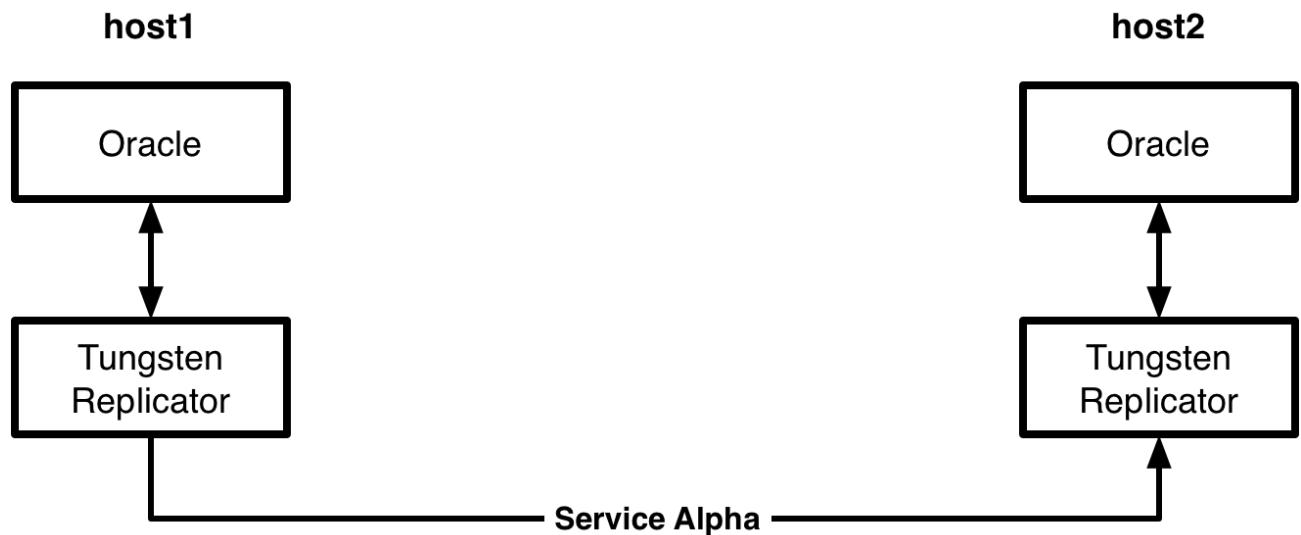
Figure 5.7. Topologies: Oracle to MySQL



For configuration, see [Section 5.7.3, “Creating an Oracle to MySQL Deployment”](#)

- Oracle to MySQL

Figure 5.8. Topologies: Oracle to Oracle



For configuration, see [Section 5.7.4, “Creating an Oracle to Oracle Deployment”](#)

Replication in these configurations operates using two separate replicators:

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

5.7.1. How Oracle Extraction Works

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

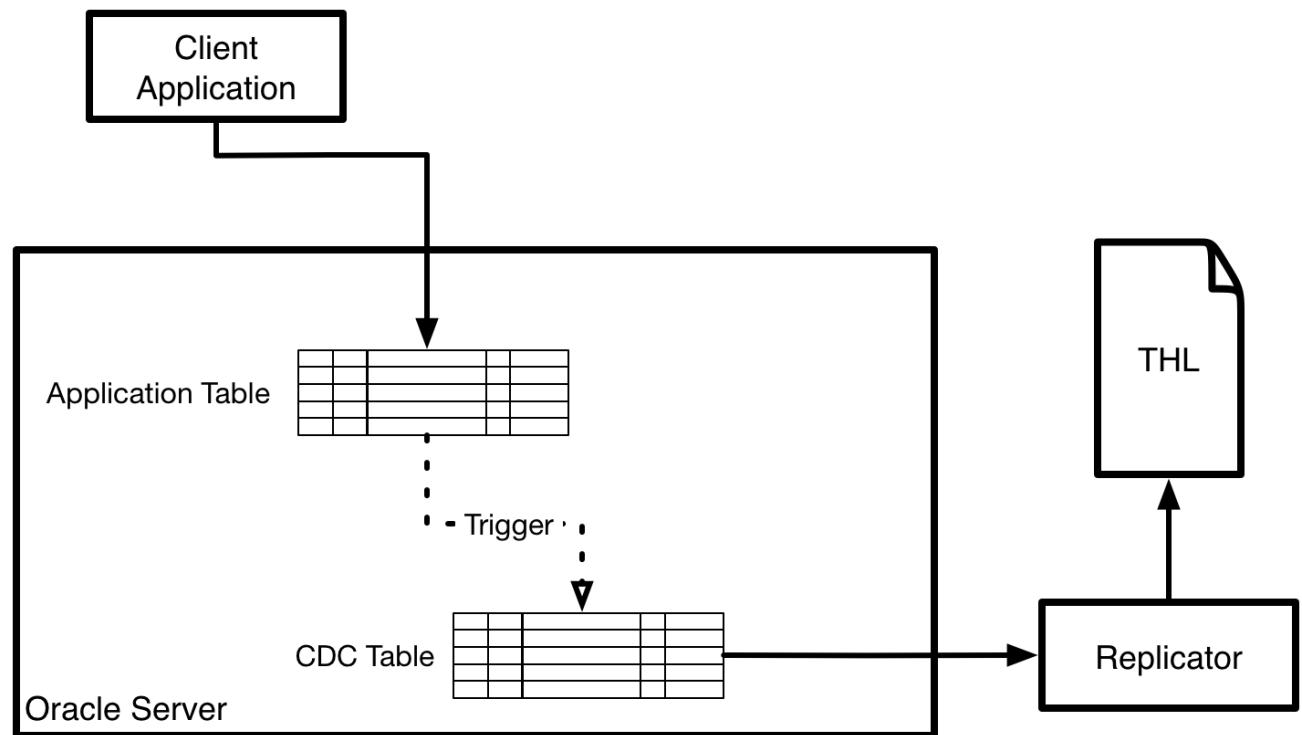
When replicating from Oracle, changes to the database are extracted using the Oracle Change Data Capture (CDC) system. Support is available for using Synchronous and Asynchronous CDC according to the version of Oracle that is being used:

Edition	Synchronous CDC	Asynchronous CDC
Standard Edition (SE)	Yes	No
Enterprise Edition (EE)	Yes	Yes
Standard Edition 1 (SE1)	Yes	No
Express Edition (XE)	No	No

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

- Synchronous CDC

Figure 5.9. Oracle Extraction with Synchronous CDC

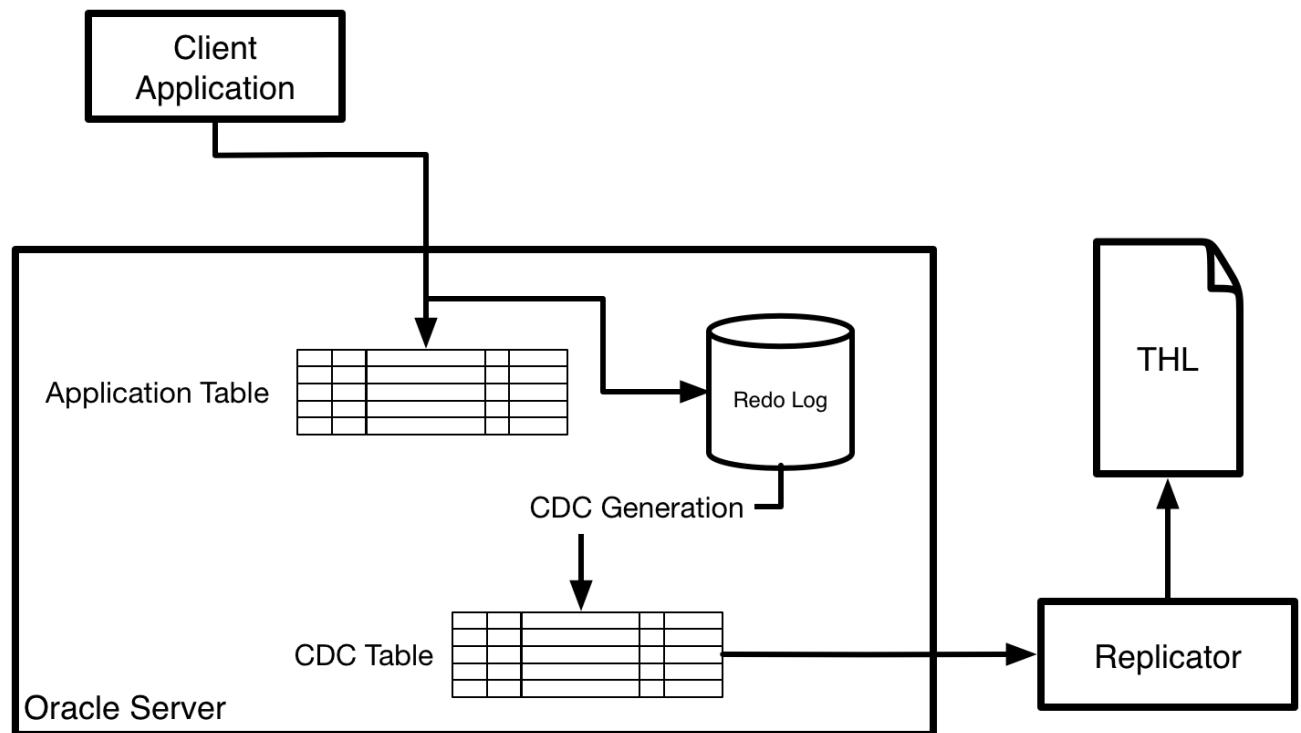


Within Synchronous CDC, triggers are created on the source tables which are configured to record the change information into the change tables. Subscribers to the change tables then read the information. With Tungsten Replicator, the replicator acts as the subscriber, reads the change information and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

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

- Asynchronous CDC

Figure 5.10. Oracle Extraction with Asynchronous CDC



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

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

Note

More information on Oracle CDC can be found within the [Oracle documentation](#).

5.7.2. Data Type Differences and Limitations

When replicating from MySQL to Oracle there are a number of datatype differences that should be accommodated to ensure reliable replication of the information. The core differences are described in [Table 5.1, "Data Type differences when replicating data from MySQL to Oracle"](#).

Table 5.1. Data Type differences when replicating data from MySQL to Oracle

MySQL Datatype	Oracle Datatype	Notes
INT	NUMBER(10, 0)	
BIGINT	NUMBER(19, 0)	
TINYINT	NUMBER(3, 0)	
SMALLINT	NUMBER(5, 0)	
MEDIUMINT	NUMBER(7, 0)	
DECIMAL(x,y)	NUMBER(x, y)	
FLOAT	FLOAT	
CHAR(n)	CHAR(n)	

MySQL Datatype	Oracle Datatype	Notes
VARCHAR(n)	VARCHAR2(n)	For sizes less than 2000 bytes data can be replicated. For lengths larger than 2000 bytes, the data will be truncated when written into Oracle
DATE	DATE	
DATETIME	DATE	
TIMESTAMP	DATE	
TEXT	CLOB	Replicator can transform TEXT into CLOB or VARCHAR(N). If you choose VARCHAR(N) on Oracle, the length of the data accepted by Oracle will be limited to 4000. This is limitation of Oracle. The size of CLOB columns within Oracle is calculated in terabytes. If TEXT fields on MySQL are known to be less than 4000 bytes (not characters) long, then VARCHAR(4000) can be used on Oracle. This may be faster than using CLOB.
BLOB	BLOB	
ENUM(...)	VARCHAR(255)	Use the enumtostring filter
SET(...)	VARCHAR(255)	Use the settostring filter

When replicating between Oracle and other database types, the `ddlscan` command can be used to generate DDL appropriate for the supported data types in the target database. For example, in MySQL to Oracle deployments the DDL can be read from the MySQL server and generated for the Oracle server so that replication can begin without manually creating the Oracle specific DDL.

When replicating *from* Oracle to MySQL or Oracle, there are limitations on the data types that can be replicated due to the nature of the CDC, whether you are using Asynchronous or Synchronous CDC for replication. The details of data types not supported by each mechanism are detailed in [Table 5.2, “Data Type Differences when Replicating from Oracle to MySQL or Oracle”](#).

Table 5.2. Data Type Differences when Replicating from Oracle to MySQL or Oracle

Data Type	Asynchronous CDC (Oracle EE Only)	Synchronous CDC (Oracle SE and EE)
BFILE	Not Supported	Not Supported
LONG	Not Supported	Not Supported
ROWID	Not Supported	Not Supported
UROWID	Not Supported	Not Supported
BLOB		Not Supported
CLOB		Not Supported
NCLOB		Not Supported
All Object Types	Not Supported	Not Supported

Note

More information on Oracle CDC can be found within the [Oracle documentation](#).

In addition, the following DDL differences and requirements exist:

- Column orders on MySQL and Oracle must match, but column names do not have to match.

Using the `dropcolumn` filter, columns can be dropped and ignored if required.

- Each table within MySQL should have a Primary Key. Without a primary key, full-row based lookups are performed on the data when performing `UPDATE` or `DELETE` operations. With a primary key, the `pkey` filter can add metadata to the `UPDATE/DELETE` event, enabling faster application of events within Oracle.
- Indexes on MySQL and Oracle do not have to match. This allows for different index types and tuning between the two systems according to application and dataserver performance requirements.
- Keywords that are restricted on Oracle should not be used within MySQL as table, column or database names. For example, the keyword `SESSION` is not allowed within Oracle. Tungsten Replication determines the column name from the target database metadata by position (column reference), not name, so replication will not fail, but applications may need to be adapted. For compatibility, try to avoid Oracle keywords.

For more information on differences between MySQL and Oracle, see [Oracle and MySQL Compared](#).

To make the process of migration from MySQL to Oracle easier, Tungsten Replication includes a tool called `ddlscan` which will read table definitions from MySQL and create appropriate Oracle table definitions to use during replication. For more information on using this tool in a MySQL to Oracle deployment, see [Section 6.1, "Deploying MySQL to Oracle Replication"](#).

For reference information on the `ddlscan` tool, see [Section 9.5, "The ddlscan Command"](#).

5.7.3. Creating an Oracle to MySQL Deployment

Replication Operation Support	
Statements Replicated	No
Rows Replicated	Yes
Schema Replicated	No
<code>ddlscan</code> Supported	Yes

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

Replication from Oracle has the following parameters:

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

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

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

5.7.3.1. Configuring the Oracle Environment

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

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

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

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

Then restart the database for the change to take effect:

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

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

Once these steps have been completed, a configuration file must be created that defines the CDC configuration. For more information on the options for `setupCDC.conf`, see [Section 9.12, "The setupCDC.sh Command"](#).

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

To configure the CDC configuration:

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

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

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

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

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

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

```
shell> cd tungsten-replicator-5.0.1-136/tungsten-replicator/extractors/oracle-cdc
shell> ./setupCDC.sh custom.conf
Using configuration custom.conf
Configuring CDC for service 'SALES' for Oracle 11. Change Set is 'TUNGSTEN_CS_SALES'
Removing old CDC installation if any (SYSDBA)
Done.
Setup tungsten_load (SYSDBA)
Done.
Creating publisher/subscriber and preparing table instantiation (SYSDBA)
Done.
Setting up HOTLOG_SOURCE (SALES_PUB)
Oracle version : 11.2.0.2.0
Setting Up Asynchronous Data Capture TUNGSTEN_CS_SALES
Processing SALES_SAMPLE -> 'CT_SAMPLE' : OK
Enabling change set : TUNGSTEN_CS_SALES
Dropping view TUNGSTEN_PUBLISHED_COLUMNS
Dropping view TUNGSTEN_SOURCE_TABLES

PL/SQL procedure successfully completed.

Done.
adding synonym if needed (tungsten)
Done.
Cleaning up (SYSDBA)
Done.
Capture started at position 16610205
```

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

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

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

5.7.3.2. Creating the MySQL Environment

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

In particular:

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

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

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

- Write row-columns in order — the default mode, columns are written to MySQL in the same order in which they are extracted from the Oracle database. This allows for differences in the table and column names in the target database, while still replicating the same information. This can be useful if there are reserved words or other differences between the two environments.
- Write using column-names — this enables the column orders to be different, but the column names to be used to apply data. This can be particularly useful if only a selection of columns are being extracted from Oracle and these selected columns are being written into MySQL. To enable this option, the following setting must be applied to the `tpm` installation command used:

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

5.7.3.3. Creating the Destination Schema

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

```
shell> cd tungsten-replicator-5.0.1-136
shell> ./bin/ddlsync -user tungsten -url 'jdbc:oracle:thin:@//host1:1521/ORCL' \
    -pass password -db access_log -template ddl-oracle-mysql.vm -db access_log
```

The output should be captured and checked before applying it to your Oracle instance:

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

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

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

The generated DDL includes statements to drop existing tables if they exist. This will fail in a new installation, but the output can be ignored.

Once the process has been completed for this database, it must be repeated for each database that you plan on replicating from MySQL to Oracle.

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

5.7.3.4. Creating the Master Replicator

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

- Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-5.0.tar.gz
```

- Change into the staging directory:

```
shell> cd tungsten-replicator-5.0
```

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

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

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

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

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

- **`tpm install SALES`**

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

- **`--datasource-oracle-service=ORCL [338]`**

Specify the Oracle service name, as configured for the database to which you want to read data. For older Oracle installations that use the SID format, use the `--datasource-oracle-sid=ORCL [339]` option to `tpm`.

- **`--datasource-type=oracle [340]`**

Defines the datasource type that will be read from, in this case, Oracle.

- **`--install-directory=/opt/continuent [347]`**

The installation directory for Tungsten Replication.

- **`--master=host1 [351]`**

The hostname of the master.

- **`--members=host1 [351]`**

The list of members for this service.

- **`--property=replicator.extractor.dbms.transaction_frag_size=10`**

Define the fragment size, or number of transactions that will be queued before extraction.

- **`--property=replicator.global.extract.db.password=password`**

The password of the subscriber user configured within `setupCDC.sh`.

- **`--property=replicator.global.extract.db.user=tungsten`**

The username of the subscriber user configured within `setupCDC.sh`.

- **`--replication-host=host1 [363]`**

The hostname of the replicator.

- **`--replication-password=password [363]`**

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

- **`--replication-port=1521 [363]`**

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

- **`--replication-user=SALES_PUB [363]`**

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

- **`--role=master [364]`**

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

- **`--start-and-report=true [365]`**

Start the replicator and report the status.

- **`--svc-table-engine=CDCSYNC [367]`**

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

<code>setupCDC.conf</code> Setting	124	<code>--svc-table-engine [367]</code> Setting
<code>SYNC_SOURCE</code>		<code>CDCSYNC</code>

<code>setupCDC.conf</code> Setting	<code>--svc-table-engine [367]</code> Setting
HOTLOG_SOURCE	CDCASYNC

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

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

```
shell> trepctl status
Processing status command...
NAME          VALUE
-----
appliedLastEventId    : ora:16626156
appliedLastSegno     : 67
appliedLatency       : 37.51
autoRecoveryEnabled  : false
autoRecoveryTotal    : 0
channels          : 1
clusterName        : SALES
currentEventId      : NONE
currentTimeMillis   : 1410430937700
dataServerHost      : tr-fromoracle1
extensions         :
host              : tr-fromoracle1
latestEpochNumber  : 67
masterConnectUri   : thl://localhost:/
masterListenUri    : thl://tr-fromoracle1:2112/
maximumStoredSeqNo : 67
minimumStoredSeqNo : 67
offlineRequests    : NONE
pendingError        : NONE
pendingErrorCode    : NONE
pendingErrorEventId: NONE
pendingErrorSeqno  : -1
pendingExceptionMessage: NONE
pipelineSource     : UNKNOWN
relativeLatency    : 38.699
resourcePrecedence: 99
rmiPort           : 10000
role              : master
seqnoType         : java.lang.Long
serviceName        : SALES
serviceType        : local
simpleServiceName  : SALES
siteName          : default
sourceId          : tr-fromoracle1
state              : ONLINE
timeInStateSeconds: 37.782
transitioningTo   :
uptimeSeconds      : 102.545
useSSLConnection  : false
version           : Tungsten Replicator 5.0.1 build 136
Finished status command...
```

5.7.3.5. Creating the Slave Replicator

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

1. Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-5.0.tar.gz
```

2. Change into the staging directory:

```
shell> cd tungsten-replicator-5.0
```

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

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

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

```
shell> ./tools/tpm install alpha \
--install-directory=/opt/continuent \
--master=host1 \
```

```
--members=host2 \
--datasource-password=password \
--datasource-user=tungsten \
--svc-applier-filters=CDC,dbtransform,optimizeupdates \
--property=replicator.filter.CDC.from=SALES_PUB.HEARTBEAT \
--property=replicator.filter.CDC.to=tungsten_alpha.heartbeat \
--property=replicator.filter.dbtransform.from_regex1=DEMO \
--property=replicator.filter.dbtransform.to_regex1=demo \
--skip-validation-check=InstallerMasterSlaveCheck \
--start-and-report
```

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

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

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

- `--members=host2 [351]`

Specifies the members of the cluster. In this case, the only member is the host into which we are deploying the slave replicator service.

- `--master=host1 [351]`

Specify the name of the master replicator that will provide the THL data to be replicated, in this case, the Oracle server configured with the CDC service.

- `--datasource-user=tungsten_alpha [363]`

The name of the user created within Oracle to be used for writing data into the Oracle tables.

- `--datasource-password=password [363]`

The password to be used by the Oracle user when writing data.

- `--install-directory=/opt/continuent [347]`

The directory where Tungsten Replicator will be installed.

- `--svc-applier-filters=dropstatementdata [366]`

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

- `--skip-validation-check=InstallerMasterSlaveCheck`

Skip validation for the MySQL master/slave operation, since that it is irrelevant in a MySQL/Oracle deployment.

- `--start-and-report [365]`

Start the service and report the status.

- `--svc-applier-filters=CDC,dbtransform,optimizeupdates [366]`

Enable a number of filters to improve the replication:

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

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

- `--property=replicator.filter.CDC.to=tungsten_alpha.heartbeat`

Defines the target name for the CDC metadata rename.

- `--property=replicator.filter.dbtransform.from_regex1=DEMO`

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

- `--property=replicator.filter.dbtransform.to_regex1=demo`

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

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

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

```
shell> trepctl status
Processing status command...
NAME          VALUE
----          -----
appliedLastEventId    : ora:16626156
appliedLastSeqno      : 67
appliedLatency        : 314.359
channels           : 1
clusterName         : alpha
currentEventId       : NONE
currentTimeMillis    : 1410431215649
dataServerHost       : tr-fromoracle2
extensions          :
host               : tr-fromoracle2
latestEpochNumber   : 67
masterConnectUri    : thl://tr-fromoracle1:2112/
masterListenUri     : thl://tr-fromoracle2:2112/
maximumStoredSeqNo  : 67
minimumStoredSeqNo  : 67
offlineRequests     : NONE
pendingError         : NONE
pendingErrorCode     : NONE
pendingErrorEventId  : NONE
pendingErrorSeqno    : -1
pendingExceptionMessage: NONE
pipelineSource       : thl://tr-fromoracle1:2112/
relativeLatency     : 316.649
resourcePrecedence   : 99
rmiPort             : 10000
role                : slave
seqnoType           : java.lang.Long
serviceName          : alpha
serviceType          : local
simpleServiceName    : alpha
siteName             : default
sourceId            : tr-fromoracle2
state               : ONLINE
timeInStateSeconds  : 2.343
transitioningTo     :
uptimeSeconds        : 74327.712
useSSLConnection    : false
version              : Tungsten Replicator 5.0.1 build 136
Finished status command...
```

5.7.4. Creating an Oracle to Oracle Deployment

Replication Operation Support	
Statements Replicated	No
Rows Replicated	Yes
Schema Replicated	No
<code>ddlsync</code> Supported	No

An Oracle to Oracle deployment replicates data between two Oracle servers, either for scaling or Data Recovery (DR) support. Enabling the Oracle to Oracle replication consists of the following parameters during replication:

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

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

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

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

5.7.4.1. Setting up the Source Oracle Environment

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

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

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

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

Then restart the database for the change to take effect:

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

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

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

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

Table 5.3. `setupCDC.conf` Configuration File Parameters

Variable	Sample Value	Description
<code>service</code>		The name of the service that will be used to process these events. It should match the name of the schema from which data is being read. The name should also match the name of the service that will be created using Tungsten Replicator to extract events from Oracle.
<code>sys_user [258]</code>	SYSDBA	The name of the SYSDBA user configured. The default (if not specified) is <code>SYSDBA</code> .
<code>sys_pass</code>		The password of the SYSDBA user; you will be prompted for this information if it has not been added.
<code>enable_archivelog</code>	0	If set to true, the Oracle instance will be configured to enable archive logging (required by Oracle CDC), and then the Oracle instance will be restarted.
<code>source_user</code>		The name of the source schema user that will be used to identify the tables used to build the publish tables. This user is created by the <code>setupCDC.sh</code> script.
<code>pub_tablespace</code>	0	By default, the system tablespace is used for holding the publisher tables. Using the system tablespace should only be used during testing, as the tablespace is typically not large enough to hold the required change data. If set to 1, use the created tablespace (matching the value of <code>pub_user</code>) which is assumed to be large enough to hold the change information.
<code>pub_user</code>		The publisher user that will be created to publish the CDC views.
<code>pub_password</code>		The publisher password that will be used when the publisher user is created.

Variable	Sample Value	Description
tungsten_user	tungsten	The subscriber user that will be created to access the CDC views. This will be used as the datasource username within the Tungsten Replicator configuration.
tungsten_pwd	password	The subscriber password that will be created to access the CDC. This will be used as the datasource username within the Tungsten Replicator configuration.
delete_publisher		If set to 1, the publisher user will be deleted before being recreated.
delete_subscriber		If set to 1, the subscriber user will be deleted before being recreated.
cdc_type [256]	SYNC_SOURCE	Specifies the CDC extraction type to be deployed. Using <code>SYNC_SOURCE</code> uses synchronous capture; <code>HOTLOG_SOURCE</code> uses asynchronous capture.
specific_tables		If set to 1, limits the replication to only use the tables listed in a <code>tungsten.tables</code> file. If set to 0, no file is used and all tables are included.
specific_path		The path of the <code>tungsten.tables</code> file. When using Oracle RAC, the location of the <code>tungsten.tables</code> file must be in a shared location accessible by Oracle RAC. If not specified, the current directory is used.

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

To configure the CDC configuration:

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

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

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

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

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

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

```
shell> cd tungsten-replicator-5.0.1-136/tungsten-replicator/scripts
shell> ./setupCDC.sh custom.conf
Using configuration custom.conf
Configuring CDC for service 'SALES' for Oracle 11. Change Set is 'TUNGSTEN_CS_SALES'
Removing old CDC installation if any (SYSDBA)
Done.
Setup tungsten_load (SYSDBA)
Done.
Creating publisher/subscriber and preparing table instantiation (SYSDBA)
```

```

Done.
Setting up HOTLOG_SOURCE (SALES_PUB)
Oracle version : 11.2.0.2.0
Setting Up Asynchronous Data Capture TUNGSTEN_CS_SALES
Processing SALES.SAMPLE -> 'CT_SAMPLE' : OK
Enabling change set : TUNGSTEN_CS_SALES
Dropping view TUNGSTEN_PUBLISHED_COLUMNS
Dropping view TUNGSTEN_SOURCE_TABLES

PL/SQL procedure successfully completed.

Done.
adding synonym if needed (tungsten)
Done.
Cleaning up (SYSDBA)
Done.
Capture started at position 16610205

```

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

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

5.7.4.2. Setting up the Target Oracle Environment

Before starting replication, the Oracle target database must be configured:

- A user and schema must exist for each database from MySQL that you want to replicate. In addition, the schema used by the services within Tungsten Replication must have an associated schema and user name.

For example, if you are replicating the database `sales` to Oracle, the following statements must be executed to create a suitable user. This can be performed through any connection, including `sqlplus`:

```

shell> sqlplus sys/oracle as sysdba
SQL> CREATE USER sales IDENTIFIED BY password DEFAULT TABLESPACE DEMO QUOTA UNLIMITED ON DEMO;

```

The above assumes a suitable tablespace has been created (`DEMO` in this case).

- A schema must also be created for each service replicating into Oracle. For example, if the source schema is called `alpha`, then the `tungsten_alpha` schema/user must be created. The same command can be used:

```

SQL> CREATE USER tungsten_alpha IDENTIFIED BY password DEFAULT TABLESPACE DEMO QUOTA UNLIMITED ON DEMO;

```

- One of the users used above must be configured so that it has the rights to connect to Oracle and has all rights so that it can execute statements on any schema:

```

SQL> GRANT CONNECT TO tungsten_alpha;
SQL> GRANT ALL PRIVILEGES TO tungsten_alpha;

```

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

5.7.4.3. Creating the Destination Schema

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

Tables should be created on the slave with the following caveats:

- Drop triggers from all tables. Triggers are not automatically disabled by Tungsten Replicator, and leaving them enabled may create data drift, duplicate keys and other errors. Triggers can be disabled on a table using:

```

SQL> ALTER TABLE sales DISABLE ALL TRIGGERS

```

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

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

```

SQL> desc SALES.sample;
Name          Null?    Type
-----        -----
ID           NOT NULL NUMBER(38)
MSG          CHAR(80)

```

Or by extracting the information from the database metadata:

```
SQL> select dbms_metadata.get_ddl( 'TABLE','SAMPLE','SALES') from dual;
DBMS_METADATA.GET_DDL('TABLE','SAMPLE','SALES')
-----
CREATE TABLE "SALES"."SAMPLE"
( "ID" NUMBER(*,0),
"MSG" CHAR(80),
PRIMARY KEY ("ID")
USING INDEX PCTFREE 10 INITTRANS 2 MAXTRANS 255 COMPUTE STATISTICS NOLOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DE
FAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "SALES_PUB" ENABLE,
SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS,
DBMS_METADATA.GET_DDL('TABLE','SAMPLE','SALES')
-----
SUPPLEMENTAL LOG DATA (UNIQUE INDEX) COLUMNS,
SUPPLEMENTAL LOG DATA (FOREIGN KEY) COLUMNS,
SUPPLEMENTAL LOG DATA (ALL) COLUMNS
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITTRANS 1 MAXTRANS 255 NOCOMPRESS NOLOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DE
FAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "SALES"
```

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

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

Will increase the displayed length to 10,000 characters.

5.7.4.4. Installing the Master Replicator

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

1. Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-5.0.tar.gz
```

2. Change into the staging directory:

```
shell> cd tungsten-replicator-5.0
```

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

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

4.

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

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

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

- `tpm install SALES`

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

- `--datasource-oracle-service=ORCL [338]`

Specify the Oracle service name, as configured for the database to which you want to read data. For older Oracle installations that use the SID format, use the `--datasource-oracle-sid=ORCL [339]` option to `tmp`.

- `--datasource-type=oracle [340]`

Defines the datasource type that will be read from, in this case, Oracle.

- `--install-directory=/opt/continuent [347]`

The installation directory for Tungsten Replication.

- `--master=host1 [351]`

The hostname of the master.

- `--members=host1 [351]`

The list of members for this service.

- `--property=replicator.extractor.dbms.transaction_frag_size=10`

Define the fragment size, or number of transactions that will be queued before extraction.

- `--property=replicator.global.extract.db.password=password`

The password of the subscriber user configured within `setupCDC.sh`.

- `--property=replicator.global.extract.db.user=tungsten`

The username of the subscriber user configured within `setupCDC.sh`.

- `--replication-host=host1 [363]`

The hostname of the replicator.

- `--replication-password=password [363]`

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

- `--replication-port=1521 [363]`

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

- `--replication-user=SALES_PUB [363]`

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

- `--role=master [364]`

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

- `--start-and-report=true [365]`

Start the replicator and report the status.

- `--svc-table-engine=CDCSYNC [367]`

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

<code>setupCDC.conf</code> Setting	<code>--svc-table-engine [367]</code> Setting
SYNC_SOURCE	CDCSYNC
HOTLOG_SOURCE	CDCASYNC

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

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

```
shell> treptctl status
```

```

Processing status command...
NAME          VALUE
----          -----
appliedLastEventId    : ora:16626156
appliedLastSeqno      : 67
appliedLatency        : 37.51
autoRecoveryEnabled   : false
autoRecoveryTotal     : 0
channels            : 1
clusterName          : SALES
currentEventId       : NONE
currentTimeMillis    : 1410430937700
dataServerHost        : tr-fromoracle1
extensions           :
host                : tr-fromoracle1
latestEpochNumber    : 67
masterConnectUri     : thl://localhost:/
masterListenUri      : thl://tr-fromoracle1:2112/
maximumStoredSeqNo   : 67
minimumStoredSeqNo   : 67
offlineRequests      : NONE
pendingError          : NONE
pendingErrorCode      : NONE
pendingErrorEventId   : NONE
pendingErrorSeqno     : -1
pendingExceptionMessage: NONE
pipelineSource        : UNKNOWN
relativeLatency       : 38.699
resourcePrecedence   : 99
rmiPort              :
role                : master
seqnoType            : java.lang.Long
serviceName          : SALES
serviceType          : local
simpleServiceName    : SALES
siteName             : default
sourceId             : tr-fromoracle1
state                : ONLINE
timeInStateSeconds   : 37.782
transitioningTo      :
uptimeSeconds         : 102.545
useSSLConnection     : false
version              : Tungsten Replicator 5.0.1 build 136
Finished status command...

```

5.7.4.5. Installing the Slave Replicator

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

1. Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-5.0.tar.gz
```

2. Change into the staging directory:

```
shell> cd tungsten-replicator-5.0
```

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

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

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

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

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

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

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

- `--members=host2 [351]`
Specifies the members of the cluster. In this case, the only member is the host into which we are deploying the slave replicator service.
- `--master=host1 [351]`
Specify the name of the master replicator that will provide the THL data to be replicated.
- `--datasource-type=oracle [340]`
Specify the datasource type, in this case Oracle. This configures the replicator to use the Oracle JDBC driver, semantics, and connect to the Oracle database to manager the replication service.
- `--datasource-oracle-service=ORCL [338]`
The name of the Oracle service within the Oracle database that the replicator will be writing data to. For older Oracle installations, where there is an explicit Oracle SID, use the `--datasource-oracle-sid [339]` command-line option to `tmp`.
- `--datasource-user=tungsten_alpha [363]`
The name of the user created within Oracle to be used for writing data into the Oracle tables.
- `--datasource-password=password [363]`
The password to be used by the Oracle user when writing data.
- `--install-directory=/opt/continuent [347]`
The directory where Tungsten Replicator will be installed.
- `--svc-applier-filters=dropstatementdata [366]`
Enables a filter that will ensure that statement information is dropped. When executing statement data that was written from MySQL, those statements cannot be executed on Oracle, so the statements are filtered out using the `dropstatementdata` filter.
- `--skip-validation-check=InstallerMasterSlaveCheck`
Skip validation for the MySQL master/slave operation, since that it is irrelevant in a MySQL/Oracle deployment.
- `--start-and-report [365]`
Start the service and report the status.

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

Once the installation has completed, the status of the service should be reported. The service should be online and reading events from the master replicator.

5.7.5. Deployment with Provisioning

You can setup the extractor from Oracle to automatically read and provision the slave database by using the [Section 8.10, “Using the Parallel Extractor”](#). The parallel extractor reads information from the source database schema in chunks and then feeds this information into the THL data stream as row-based INSERT operations. When the slave connects, these are applied to the slave database as with a normal INSERT operations. The parallel extractor is particularly useful in heterogeneous environments such as Oracle to MySQL where the slave data does not already exist on the slave.

The basic provisioning process operates in two stages:

1. Provisioning data is extracted and inserted into the THL. One event is used to contain all of the data from a single table. If the table is too large to be contained in a single event, the data will be distributed over multiple events.
2. Once provisioning has finished, data is extracted from the CDC as normal and added to the THL

Important

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

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

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

To setup provisioning with parallel extractor:

1. Run `setupCDC.sh` to create the Oracle CDC infrastructure.
2. Install master Tungsten Replicator using `tpm`, but do not enable automatic starting (i.e. do not use the `--start` [365] or `--start-and-report` [365] options).
3. On the slave database, create the destination tables for the schemas being replicated. This can be achieved either manually or by using `ddlscan` to create the required table definitions.
4. Install the slave replicator as normal; this can be a MySQL or Oracle destination.
5. On the master:
 - a. Start the replicator in `OFFLINE` [207] mode using `replicator start offline`:

```
shell> replicator start offline
```

- b. Put the replicator into the `ONLINE` [207] state, using the `-provision` option:

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

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

```
Capture started at position 40748375
```

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

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

During the provisioning process, the replicator will show the status `GOING-ONLINE: PROVISIONING` [207] until all of the data has been read from the existing database.

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

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

Important

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

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

For more information on tuning the parallel extractor, see [Section 8.10.1, “Advanced Configuration Parameters”](#).

5.7.6. Updating CDC after Schema Changes

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

Schema changes should therefore be performed as follows:

1. Stop the replicator using `trepctl offline`:

```
shell> trepctl offline
```

2. Change the schema definition within Oracle.

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

4. To update multiple tables, the entire setup process must be started again; run the `setupCDC.sh` command, using the original configuration file, but with the `delete_publisher` set to 1:

```
shell> setupCDC.sh setupCDC.conf
```

- If only one table has changed, or you are adding only a single table, this can be specified on the command-line:

```
shell> updateCDC.sh setupCDC.conf samptable
```

5. Put the replicator back online with `trepctl online`:

```
shell> repctl online
```

To add a new table to an existing configuration:

1. Stop the replicator using `trepctl offline`:

```
shell> repctl offline
```

2. Update the configuration using `updateCDC.sh`, supplying the new table name.

```
shell> updateCDC.sh setupCDC.conf newtable
```

If you have used a specific tables file (i.e. with `specific_tables=1` in the configuration file), make sure that you add the table to the table file.

3. Put the replicator back online with `trepctl online`:

```
shell> repctl online
```

5.7.7. CDC Cleanup and Correction

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

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

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

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

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

Note

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

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

Once completed, `setupCDC.sh` can be executed again. See [Section 6.1.2.2, “Configure the Oracle database”](#) for more information.

5.7.8. Tuning CDC Extraction

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

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

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

```
shell> tpm update alpha --property=replicator_extractor.dim, maxSleepTime=32 ...
```

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

Sleep	Data
16s	No data

Sleep	Data
32s	No data
	Data found
16s	No data
32s	No data
32s	No data
32s	No data

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

Sleep	Data
16s	No data
24s	No data
	Data found
16s	No data
24s	No data
32s	No data
32s	No data

5.7.9. Troubleshooting Oracle CDC Deployments

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

- Extractor Slow-down on Single Service

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

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

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

Last Updated: 2016-04-20

Condition or Error

It is possible for the Oracle server to get into a state where Tungsten Replication 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 Replication.

Rectifications

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

```
shell> sqlplus sys/oracle as sysdba
SQL> ALTER SYSTEM SET db_recovery_file_dest_size = 80G;
```

Chapter 6. Heterogeneous MySQL Deployments

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

- MySQL to Oracle, Oracle to MySQL and Oracle to Oracle, using either the [Redo Reader](#) method or [CDC](#) method or
- [MySQL or Oracle to Hadoop](#)
- [MySQL or Oracle to Amazon Redshift](#)
- [MySQL to Vertica](#)

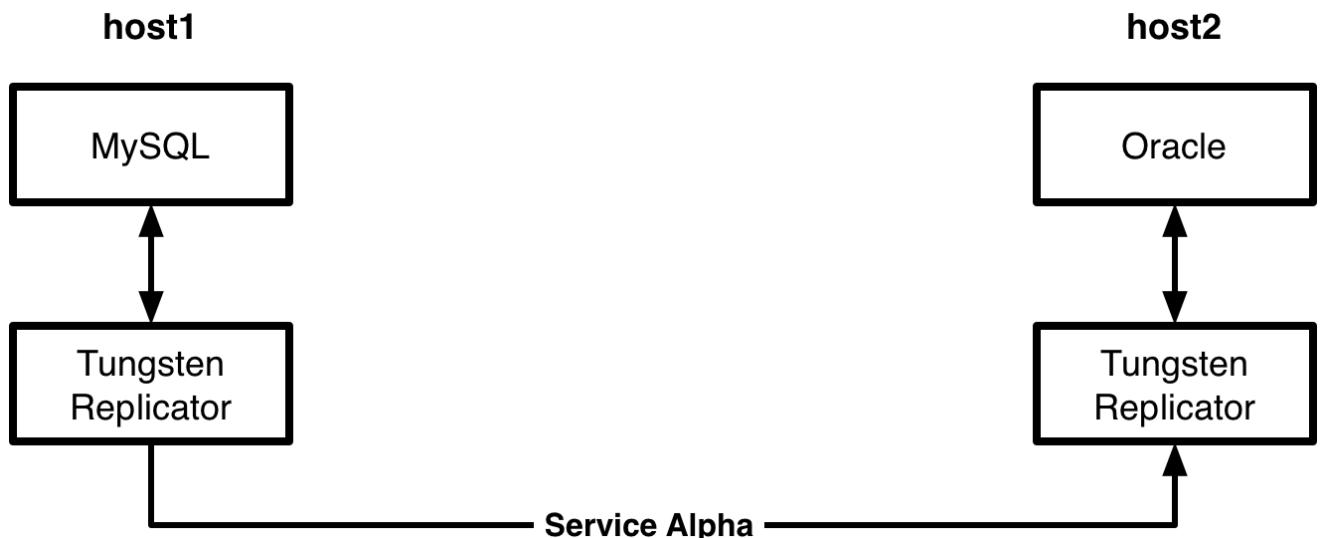
The following sections provide more detail and information on the setup and configuration of these different solutions.

6.1. Deploying MySQL to Oracle Replication

Replication Operation Support	
Statements Replicated	No
Rows Replicated	Yes
Schema Replicated	No
ddlsupport Supported	Yes

Tungsten Replication supports replication to Oracle as a datasource. This allows replication of data from MySQL to Oracle. See the [Database Support](#) prerequisites for more details.

Figure 6.1. Topologies: MySQL to Oracle



Replication in these configurations operates using two separate replicators:

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

6.1.1. Prepare: MySQL to Oracle Replication

When replicating from MySQL to Oracle there are a number of datatype differences that should be accommodated to ensure reliable replication of the information. The core differences are described in [Table 5.1, "Data Type differences when replicating data from MySQL to Oracle"](#).

Table 6.1. Data Type differences when replicating data from MySQL to Oracle

MySQL Datatype	Oracle Datatype	Notes
INT	NUMBER(10, 0)	
BIGINT	NUMBER(19, 0)	
TINYINT	NUMBER(3, 0)	
SMALLINT	NUMBER(5, 0)	
MEDIUMINT	NUMBER(7, 0)	
DECIMAL(x,y)	NUMBER(x, y)	
FLOAT	FLOAT	
CHAR(n)	CHAR(n)	
VARCHAR(n)	VARCHAR2(n)	For sizes less than 2000 bytes data can be replicated. For lengths larger than 2000 bytes, the data will be truncated when written into Oracle
DATE	DATE	
DATETIME	DATE	
TIMESTAMP	DATE	
TEXT	CLOB	Replicator can transform TEXT into CLOB or VARCHAR(N). If you choose VARCHAR(N) on Oracle, the length of the data accepted by Oracle will be limited to 4000. This is limitation of Oracle. The size of CLOB columns within Oracle is calculated in terabytes. If TEXT fields on MySQL are known to be less than 4000 bytes (not characters) long, then VARCHAR(4000) can be used on Oracle. This may be faster than using CLOB.
BLOB	BLOB	
ENUM(...)	VARCHAR(255)	Use the EnumToString filter
SET(...)	VARCHAR(255)	Use the SetToString filter

When replicating from MySQL to Oracle, the [ddlscan](#) command can be used to generate DDL appropriate for the supported data types in the target database. In MySQL to Oracle deployments the DDL can be read from the MySQL server and generated for the Oracle server so that replication can begin without manually creating the Oracle specific DDL.

In addition, the following DDL differences and requirements exist:

- Column orders on MySQL and Oracle must match, but column names do not have to match.
Using the [dropcolumn](#) filter, columns can be dropped and ignored if required.
- Each table within MySQL should have a Primary Key. Without a primary key, full-row based lookups are performed on the data when performing [UPDATE](#) or [DELETE](#) operations. With a primary key, the [pkey](#) filter can add metadata to the [UPDATE/DELETE](#) event, enabling faster application of events within Oracle.
- Indexes on MySQL and Oracle do not have to match. This allows for different index types and tuning between the two systems according to application and dataserver performance requirements.
- Keywords that are restricted on Oracle should not be used within MySQL as table, column or database names. For example, the keyword [SESSION](#) is not allowed within Oracle. Tungsten Replication determines the column name from the target database metadata by position (column reference), not name, so replication will not fail, but applications may need to be adapted. For compatibility, try to avoid Oracle keywords.

For more information on differences between MySQL and Oracle, see [Oracle and MySQL Compared](#).

To make the process of migration from MySQL to Oracle easier, Tungsten Replication includes a tool called [ddlscan](#) which will read table definitions from MySQL and create appropriate Oracle table definitions to use during replication.

For reference information on the [ddlscan](#) tool, see [Section 9.5, “The ddlscan Command”](#).

6.1.2. Install: MySQL to Oracle Replication

When replicating from MySQL to Oracle there are a number of key steps that must be performed. The primary process is the preparation of the Oracle database and DDL for the database schema that are being replicated. Although DDL statements will be replicated to Oracle, they will often fail because of SQL language differences. Because of this, tables within Oracle must be created before replication starts.

A brief list of the major steps involved are listed below:

1. Configure the MySQL database
2. Configure the Oracle database
3. Install the Master replicator to extract information from the Oracle database using the information generated by the CDC
4. Extract the schema from Oracle and apply it to MySQL
5. Install the Slave replicator to read data from the master replicator and apply it to MySQL

Each of these steps has particular steps and commands that must be executed. A detailed sequence of steps is provided below:

6.1.2.1. Configure the MySQL database

MySQL must be operating in ROW format for the binary log. Statement-based replication is not supported. In addition, for compatibility reasons, MySQL should be configured to use UTF8 and a neutral timezone.

- MySQL must be using Row-based replication for information to be replicated to Oracle. For the best results, you should change the global binary log format, ideally in the configuration file (`my.cnf`):

```
binlog-format = row
```

Alternatively, the global binlog format can be changed by executing the following statement:

```
mysql> SET GLOBAL binlog-format = ROW;
```

For MySQL 5.6.2 and later, you must enable full row log images:

```
binlog-row-image = full
```

This information will be forgotten when the MySQL server is restarted; placing the configuration in the `my.cnf` file will ensure this option is permanently enabled.

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

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

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

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

6.1.2.2. Configure the Oracle database

Before starting replication, the Oracle target database must be configured:

- A user and schema must exist for each database from MySQL that you want to replicate. In addition, the schema used by the services within Tungsten Replication must have an associated schema and user name.

For example, if you are replicating the database `sales` to Oracle, the following statements must be executed to create a suitable user. This can be performed through any connection, including `sqlplus`:

```
shell> sqlplus sys/oracle as sysdba
SQL> CREATE USER sales IDENTIFIED BY password DEFAULT TABLESPACE DEMO QUOTA UNLIMITED ON DEMO;
```

The above assumes a suitable tablespace has been created (`DEMO` in this case).

- A schema must also be created for each service replicating into Oracle. For example, if the service is called `alpha`, then the `tungsten_alpha` schema/user must be created. The same command can be used:

```
SQL> CREATE USER tungsten_alpha IDENTIFIED BY password DEFAULT TABLESPACE DEMO QUOTA UNLIMITED ON DEMO;
```

- One of the users used above must be configured so that it has the rights to connect to Oracle and has all rights so that it can execute statements on any schema:

```
SQL> GRANT CONNECT TO tungsten_alpha;
SQL> GRANT ALL PRIVILEGES TO tungsten_alpha;
```

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

6.1.2.3. Create the Destination Schema

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

```
shell> cd tungsten-replicator-5.0.1-136
shell> ./bin/ddlsync -user tungsten -url 'jdbc:mysql:thin://host1:13306/access_log' \
    -pass password -template ddl-mysql-oracle.vm -db access_log
```

The output should be captured and checked before applying it to your Oracle instance:

```
shell> ./bin/ddlsync -user tungsten -url 'jdbc:mysql:thin://host1:13306/access_log' \
    -pass password -template ddl-mysql-oracle.vm -db access_log > access_log.ddl
```

If you are happy with the output, it can be executed against your target Oracle database:

```
shell> cat access_log.ddl | sqlplus sys/oracle as sysdba
```

The generated DDL includes statements to drop existing tables if they exist. This will fail in a new installation, but the output can be ignored.

Once the process has been completed for this database, it must be repeated for each database that you plan on replicating from Oracle to MySQL.

6.1.2.4. Install the Master Replicator Service

The master replicator is responsible for reading information from the MySQL binary log, converting that data into the THL format, and then exposing that data to the slave that will apply the data into Oracle.

To configure the master replicator, use `tpm` to create a simple replicator installation, in this case enabling heterogeneous operations, and the required user, password and installation directory.

```
shell> ./tools/tpm install alpha \
    --master=host1 \
    --install-directory=/opt/continuent \
    --replication-user=tungsten \
    --replication-password=password \
    --enable-heterogeneous-master=true \
    --start
```

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

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

- [tpm install](#)

Executes `tpm` in `install` mode to create the service `alpha`.

- [--master=host1 \[351\]](#)

Specifies which host will be the master.

- [--replication-user=tungsten \[363\]](#)

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

- [--install-directory=/opt/continuent \[347\]](#)

Directory where Tungsten Replication will be installed.

- [--replication-password=password \[363\]](#)

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

- [--enable-heterogeneous-master=true \[344\]](#)

This enables a number of filters and settings that ensure heterogeneous support is enabled, including enabling the correct filtering, Java character, string handling and time settings.

- [--start \[365\]](#)

This starts the replicator service once the replicator has been configured and installed.

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

Once the master replicator has been installed and started, the contents of the binary log will be read and written into THL.

6.1.2.5. Install Slave Replicator

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

1. Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-5.0.tar.gz
```

2. Change into the staging directory:

```
shell> cd tungsten-replicator-5.0
```

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

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

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

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

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

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

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

- `--members=host2` [351]

Specifies the members of the cluster. In this case, the only member is the host into which we are deploying the slave replicator service.

- `--master=host1` [351]

Specify the name of the master replicator that will provide the THL data to be replicated.

- `--datasource-type=oracle` [340]

Specify the datasource type, in this case Oracle. This configures the replicator to use the Oracle JDBC driver, semantics, and connect to the Oracle database to manager the replication service.

- `--datasource-oracle-service=ORCL` [338]

The name of the Oracle service within the Oracle database that the replicator will be writing data to. For older Oracle installations, where there is an explicit Oracle SID, use the `--datasource-oracle-sid` [339] command-line option to `tpm`.

- `--datasource-user=tungsten_alpha` [363]

The name of the user created within Oracle to be used for writing data into the Oracle tables.

- `--datasource-password=password` [363]

The password to be used by the Oracle user when writing data.

- `--install-directory=/opt/continuent` [347]

The directory where Tungsten Replicator will be installed.

- `--svc-applier-filters=dropstatementdata` [366]

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

- `--skip-validation-check=InstallerMasterSlaveCheck`

Skip validation for the MySQL master/slave operation, since that is irrelevant in a MySQL/Oracle deployment.

- `--start-and-report [365]`

Start the service and report the status.

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

Once the installation has completed, the status of the service should be reported. The service should be online and reading events from the master replicator.

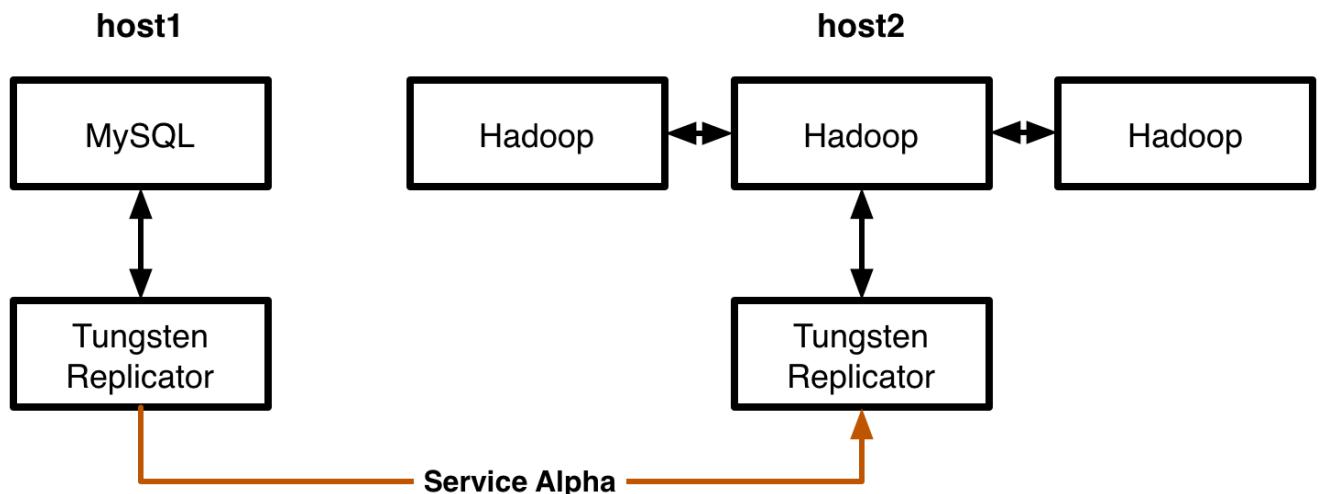
6.2. Deploying MySQL to Hadoop Replication

Replicating data into Hadoop is achieved by generating character-separated values from ROW-based information that is applied directly to the Hadoop HDFS using a `batch loading` process. Files are written directly to the HDFS using the Hadoop client libraries. A separate process is then used to merge existing data, and the changed information extracted from the master database.

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

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

Figure 6.2. Topologies: MySQL to Hadoop



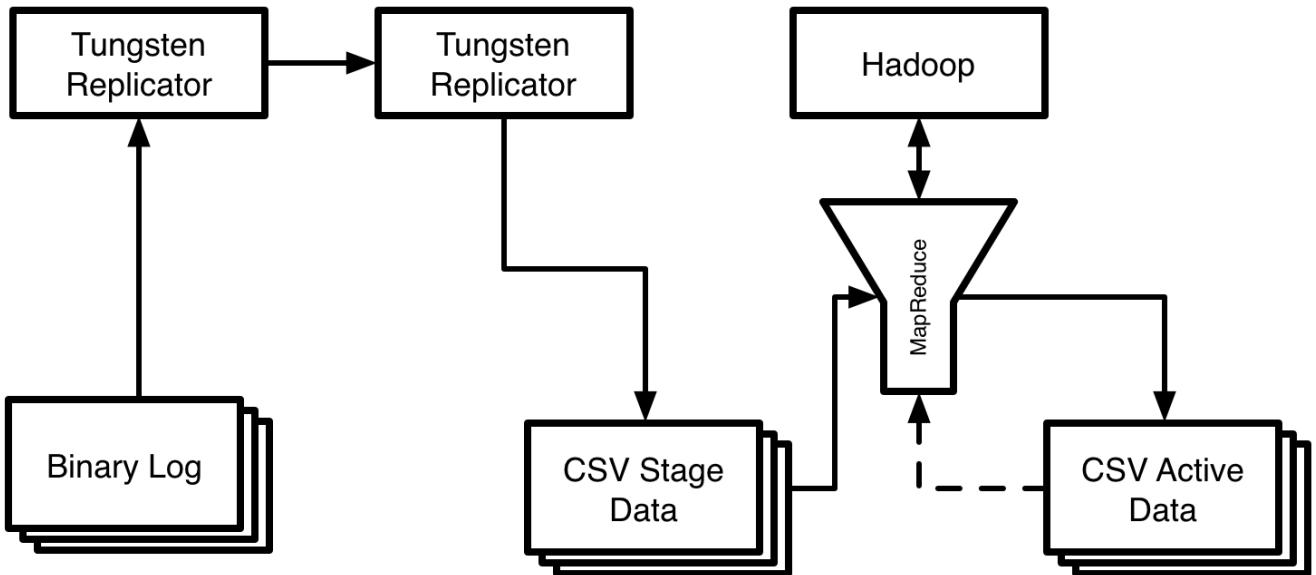
Basic requirements for replication into Hadoop:

- Hadoop Replication is supported on the following Hadoop distributions and releases:
 - Cloudera Enterprise 4.4, Cloudera Enterprise 5.0 (Certified)
 - HortonWorks DataPlatform 2.0
 - Amazon Elastic MapReduce
 - IBM InfoSphere BigInsights 2.1 and 3.0
 - MapR 3.0, 3.1, and 5.x
 - Pivotal HD 2.0
 - Apache Hadoop 2.1.0, 2.2.0
- Source tables must have primary keys. Without a primary key, Tungsten Replicator is unable to determine the row to be updated when the data reaches Hadoop.

6.2.1. Hadoop Replication Operation

The Hadoop replicator makes use of the JavaScript based batch loading system (see [Section 7.2.4, “JavaScript Batchloader Scripts”](#)). This constructs change data from the source-database, and uses this information in combination with any existing data to construct, using Hive, a materialized view. A summary of this basic structure can be seen in [Figure 6.3, “Topologies: MySQL to Hadoop Replication Operation”](#).

Figure 6.3. Topologies: MySQL to Hadoop Replication Operation



The full replication of information operates as follows:

1. Data is extracted from the source database using the standard extractor, for example by reading the row change data from the binlog in MySQL.
2. The `colnames` filter is used to extract column name information from the database. This enables the row-change information to be tagged with the corresponding column information. The data changes, and corresponding row names, are stored in the THL.
The `pkey` filter is used to extract primary key data from the source tables.
3. On the slave replicator, the THL data is read and written into batch-files in the character-separated value format.

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

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

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

- The format of the files is a character separated values file, with each row separated by a newline, and individual fields separated by the character `0x01`. This is supported by Hive as a native value separator.
- The content of the file consists of the full row data extracted from the master, plus metadata describing the operation for each row, the sequence number, and then the full row information.

Operation	Sequence No	Table-specific primary key	Table-column
I (Insert) or D (Delete)	<code>SEQNO</code> [459] that generated this row		

For example, the MySQL row:

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

Is represented within the staging files generated as:

```
I^A1318^A3^A3^A#1 Single^A2006^ACats and Dogs (#1.4)
```

The character separator, and whether to use quoting, are configurable within the replicator when it is deployed. The default is to use a newline character for records, and the `0x01` character for fields.

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

Table 6.2. Hadoop Replication Directory Locations

Directory/File	Description
<code>/user/tungsten</code>	Top-level directory for Tungsten Replicator information
<code>/user/tungsten/metadata</code>	Location for metadata related to the replication operation
<code>/user/tungsten/metadata/alpha</code>	The directory (named after the servicename of the replicator service) that holds service-specific metadata
<code>/user/tungsten/staging</code>	Directory of the data transferred
<code>/user/tungsten/staging/servicename</code>	Directory of the data transferred from a specific servicename.
<code>/user/tungsten/staging/servicename/databasename</code>	Directory of the data transferred specific to a database.
<code>/user/tungsten/staging/servicename/databasename tablename</code>	Directory of the data transferred specific to a table.
<code>/user/tungsten/staging/servicename/databasename tablename tablename-##.csv</code>	Filename of a single file of the data transferred for a specific table and database.

Files are automatically created, named according to the parent table name, and the starting Tungsten Replicator sequence number for each file that is transferred. The size of the files is determined by the batch and commit parameters. For example, in the truncated list of files below displayed using the `hadoop fs` command,

```
shell> hadoop fs -ls /user/tungsten/staging/hadoop/chicago
Found 66 items
-rw-r--r-- 3 cloudera cloudera 1270236 2014-01-13 06:58 /user/tungsten/staging/alpha/hadoop/chicago/chicago-10.csv
-rw-r--r-- 3 cloudera cloudera 10274189 2014-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-103.csv
-rw-r--r-- 3 cloudera cloudera 1275832 2014-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-104.csv
-rw-r--r-- 3 cloudera cloudera 1275411 2014-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-105.csv
-rw-r--r-- 3 cloudera cloudera 10370471 2014-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-113.csv
-rw-r--r-- 3 cloudera cloudera 1279435 2014-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-114.csv
-rw-r--r-- 3 cloudera cloudera 2544062 2014-01-13 06:58 /user/tungsten/staging/alpha/hadoop/chicago/chicago-12.csv
-rw-r--r-- 3 cloudera cloudera 11694202 2014-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-123.csv
-rw-r--r-- 3 cloudera cloudera 1279072 2014-01-13 08:34 /user/tungsten/staging/alpha/hadoop/chicago/chicago-124.csv
-rw-r--r-- 3 cloudera cloudera 2570481 2014-01-13 08:34 /user/tungsten/staging/alpha/hadoop/chicago/chicago-126.csv
-rw-r--r-- 3 cloudera cloudera 9073627 2014-01-13 08:34 /user/tungsten/staging/alpha/hadoop/chicago/chicago-133.csv
-rw-r--r-- 3 cloudera cloudera 1279708 2014-01-13 08:34 /user/tungsten/staging/alpha/hadoop/chicago/chicago-134.csv
...
```

The individual file numbers will not be sequential, as they will depend on the sequence number, batch size and range of tables transferred.

6.2.2. Preparing Hosts for Hadoop Replication

During the replication process, data is exchanged from the MySQL database/table/row structure into corresponding Hadoop directory and files, as shown in the table below:

MySQL	Hadoop
Database	Directory
Table	Hive-compatible Character-Separated Text file
Row	Line in the text file, fields terminated by character <code>0x01</code>

6.2.2.1. MySQL Host

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

The following are *required* for replication to Hadoop:

- MySQL must be using Row-based replication for information to be replicated to Hadoop. For the best results, you should change the global binary log format, ideally in the configuration file (`my.cnf`):

```
binlog-format = row
```

Alternatively, the global binlog format can be changed by executing the following statement:

```
mysql> SET GLOBAL binlog-format = ROW;
```

For MySQL 5.6.2 and later, you must enable full row log images:

```
binlog-row-image = full
```

This information will be forgotten when the MySQL server is restarted; placing the configuration in the `my.cnf` file will ensure this option is permanently enabled.

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

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

Tables must also be configured as UTF8 tables, and existing tables should be updated to UTF8 support before they are replicated to prevent character set corruption issues.

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

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

- Each table that is being replicated *must* have a primary key. Failure to provide a primary key in the table definition will cause replication to stop.
- All tables to be replicated should include a primary key. Failure to include a primary key on a table will cause replication to fail.

6.2.2.2. Hadoop Host

The Hadoop environment should have the following features and parameters for the most efficient operation:

- Disk storage

There must be enough disk storage for the change data, data being actively merged, and the live data for the replicated information. Depending on the configuration and rate of changes in the master, the required data space will fluctuate.

For example, replicating a 10GB dataset, and 5GB of change data during replication, will require at least 30GB of storage. 10GB for the original dataset, 5GB of change data, and 10-25GB of merged data. The exact size is dependent on the quantity of inserts/updates/deletes.

- Pre-requisites

Currently, deployment of the slave to a relay host is not supported. One host within the Hadoop cluster must be chosen to act as the slave.

The prerequisites for a standard Tungsten Replication should be followed, including:

- [Section B.3.1, “Creating the User Environment”](#)
- [Section B.3.2.1, “Network Ports”](#)
- [Section B.3.3, “Directory Locations and Configuration”](#)
- [Section B.3.4, “Configure Software”](#)

This will provide the base environment into which Tungsten Replicator can be installed.

- HDFS Location

The `/user/tungsten` directory must be writable by the replicator user within HDFS:

```
shell> hadoop fs -mkdir /user/tungsten
shell> hadoop fs -chmod 700 /user/tungsten
shell> hadoop fs -chown tungsten /user/tungsten
```

These commands should be executed by a user with HDFS administration rights (e.g. the `hdfs` user).

- Replicator User Group Membership

The user that will be executing the replicator (typically `tungsten`, as recommended in the [Appendix B, Prerequisites](#)) must be a member of the `hive` group on the Hadoop host where the replicator will be installed. Without this membership, the user will be unable to execute Hive queries.

6.2.2.3. Schema Generation

In order to access the generated tables, both staging and the final tables, it is necessary to create a schema definition. The `ddlscan` tool can be used to read the existing definition of the tables from the source server and generate suitable Hive schema definitions to access the table data.

To create the staging table definition, use the `ddl-mysql-hive-0.10.vm` template; you must specify the JDBC connection string, user, password and database names. For example:

```
shell> ddlsan -user tungsten -url 'jdbc:mysql:thin://host1:13306/test' -pass password \
  -template ddl-mysql-hive-0.10.vm -db test
--
-- SQL generated on Wed Jan 29 16:17:05 GMT 2014 by Tungsten ddlsan utility
--
-- url = jdbc:mysql:thin://host1:13306/test
-- user = tungsten
-- db_name = test
--
CREATE DATABASE test;

DROP TABLE IF EXISTS test.movies_large;

CREATE TABLE test.movies_large
(
    id INT ,
    title STRING ,
    year INT ,
    episodetitle STRING )
;
```

The output from this command should be applied to your Hive installation within the Hadoop cluster. For example, by capturing the output, transferring that file and then running:

```
shell> cat schema.sql | hive
```

To create Hive tables that read the staging files loaded by the replicator, use the `ddl-mysql-hive-0.10-staging.vm`:

```
shell> ddlsan -user tungsten -url 'jdbc:mysql:thin://host1:13306/test' -pass password \
  -template ddl-mysql-hive-0.10-staging.vm -db test
```

The process creates the schema and tables which match the schema and table names on the source database.

Transfer this file to your Hadoop environment and then create the generated schema:

```
shell> cat schema-staging.sql |hive
```

The process creates matching schema names, but table names are modified to include the prefix `stage_xxx_`. For example, for the table `movies_large` a staging table named `stage_xxx_movies_large` is created. The Hive table definition is created pointing to the external file-based tables, using the default `0x01` field separator and `0x0A` (newline) record separator. If different values were used for these in the configuration, the schema definition in the captured file from `ddlsan` should be updated by hand.

The tables should now be available within Hive. For more information on accessing and using the tables, see [Section 6.2.5, “Accessing Generated Tables in Hive”](#).

6.2.3. Installing Hadoop Replication

Installation of the Hadoop replication consists of multiple stages:

1. Install the Master replicator extract information from your source database. Separate instructions are available for:
 - MySQL Master, see [Section 6.2.3.1, “MySQL to Hadoop Master Replicator Service”](#)
 - Oracle Master, see [Section 6.2.3.2, “Oracle to Hadoop Master Replicator Service”](#)
2. Install the Slave replicator which will apply information to the target Hadoop environment. See [Section 6.2.3.3, “Slave Replicator Service”](#). Separate instructions are available for installation to different Hadoop platforms:
 - Native (Apache) Slave, see [Section 6.2.3.3.1, “Slave Replicator Service \(Native Hadoop\)”](#)
 - Amazon Elastic Map Reduce (EMR), see [Section 6.2.3.3.3, “Slave Replicator Service \(Amazon EMR\)”](#)
 - Cloudera Enterprise (CDH) 4.x or 5.x, see [Section 6.2.3.3.2, “Slave Replicator Service \(Cloudera\)”](#)
 - HortonWorks Data Platform 2.0 or 2.1, see [Section 6.2.3.3.4, “Slave Replicator Service \(HortonWorks\)”](#)
 - IBM InfoSphere BigInsights, see [Section 6.2.3.3.5, “Slave Replicator Service \(IBM InfoSphere BigInsights\)”](#)

- MapR, M3, M5 or M7, see [Section 6.2.3.3.6, "Slave Replicator Service \(MapR\)"](#)
3. Once the installation of the Master and Slave components have been completed, materialization of tables and views on

6.2.3.1. MySQL to Hadoop Master Replicator Service

To configure the master replicator, which extracts information from MySQL into THL. The instructions below assume you are using the staging directory installation method. For more information on comparing staging and INI based installation methods, see [Section 10.1, "Comparing Staging and INI tpm Methods"](#).

1. Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-5.0.tar.gz
```

2. Change into the staging directory:

```
shell> cd tungsten-replicator-5.0
```

3. Perform the installation using **tpm**:

```
shell> ./tools/tpm install alpha \
--install-directory=/opt/continuent \
--master=host1 \
--members=host1 \
--enable-heterogeneous-master=true \
--enable-batch-service=true \
--replication-password=password \
--replication-user=tungsten \
--skip-validation-check=HostsFileCheck \
--skip-validation-check=ReplicationServicePipelines \
--start-and-report=true
```

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

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

- **tpm install**

Executes **tpm** in **install** mode to create the service **alpha**.

- **--install-directory=/opt/continuent [347]**

Directory where Tungsten Replication will be installed.

- **--master=host1 [351]**

Specifies which host will be the master.

- **--enable-heterogeneous-master=true [344],**

This option is an alias for a number of options that ensure that data is extracted from the master binary log, setting the correct column names, primary key information, and setting string, rather than **SET** or **ENUM** references, and also sets the correct string encoding to make the exchange of data more compatible with heterogeneous targets.

- **--enable-batch-service=true [343]**

Ensures that column and primary key information is added to the THL during extraction, including adding primary key and data information to ROW data. Within replication to Hadoop this is required to ensure that the extracted row data contains all the information needed to perform the transfer of correct ROW data.

- **--replication-user=tungsten [363]**

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

- **--replication-password=password [363]**

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

- **--skip-validation-check=HostsFileCheck, --skip-validation-check=ReplicationServicePipelines**

Disables certain checks normally performed during installation. These checks would fail the installation process.

- **--start [365]**

This starts the replicator service once the replicator has been configured and installed.

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

Once the replicator service has started, the status of the service can be checked using `trepctl`. See [Section 6.2.6, “Management and Monitoring of Hadoop Deployments”](#) for more information.

6.2.3.2. Oracle to Hadoop Master Replicator Service

6.2.3.3. Slave Replicator Service

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

Once the service has been installed it can be monitored using the `trepctl` command. See [Section 6.2.6, “Management and Monitoring of Hadoop Deployments”](#) for more information. If there are problems during installation, see [Section 6.2.6.1, “Troubleshooting Hadoop Replication”](#).

6.2.3.3.1. Slave Replicator Service (Native Hadoop)

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

Note

Installation must currently take place on a node within the Hadoop cluster. Writing to a remote HDFS filesystem is not currently supported.

The `tpm` required to install the slave replicator:

1. Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-5.0.tar.gz
```

2. Change into the staging directory:

```
shell> cd tungsten-replicator-5.0
```

3. Execute the `tpm` to perform the installation. The `tpm` command shown below configures a loading mechanism using files that are copied into HDFS by the replicator.

```
shell> ./tools/tpm install alpha \
--batch-enabled=true \
--batch-load-language=js \
--batch-load-template=hadoop \
--datasource-type=file \
--install-directory=/opt/continuent \
--master=host1 \
--members=host2 \
--property=replicator.datasource.global.csvType=hive \
--property=replicator.stage.q-to-dbms.blockCommitInterval=1s \
--property=replicator.stage.q-to-dbms.blockCommitRowCount=1000 \
--replication-password=secret \
--replication-user=tungsten \
--skip-validation-check=DatasourceDBPort \
--skip-validation-check=DirectDatasourceDBPort \
--skip-validation-check=HostsFileCheck \
--skip-validation-check=InstallerMasterSlaveCheck \
--skip-validation-check=ReplicationServicePipelines \
--rmi-port=25550 \
--start-and-report=true
```

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

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

- `tpm install`

Executes `tpm` in `install` mode to create the service `alpha`.

- `--batch-enabled=true [331], --batch-load-language=js [331], --batch-load-template=hdfs [331]`

Enables batch-loading, using JavaScript as the batch loading language, writing to Hadoop or directly in to HDFS. The `--batch-load-template [331]` supports two options, `hadoop` and `hdfs`. In `hadoop` mode, files are written to disk and then copied to HDFS using the `hadoop dfs` command.

When using the hdfs mode, files are written directly into the HDFS filesystem by connecting to the HDFS service. When using the hdfs method the `--java-external-lib-dir` [348] option must also be specified, pointing to a valid Hadoop client library directory where the HDFS and Hadoop JAR files are located.

- `--datasource-type=file` [340]

Defines the format of the information to be generated when used through the batch loader. When using the hadoop batch load template (`--batch-load-template` [331], this option should be set to file .

When using the hdfs template, the `--datasource-type` [340] option should be set to hdfs .

- `--install-directory=/opt/continuent` [347]

Directory where Tungsten Replication will be installed.

- `--java-external-lib-dir=/usr/lib/hadoop/client` [348]

Location of additional libraries to be added to the installation. This should point to the location of the Hadoop client libraries. In a default Cloudera installation this is located in `/usr/lib/hadoop/client`.

- `--master=host1` [351]

Specifies the master host where THL data will be read from, and should point to the server configured in [Section 6.2.3.3, “Slave Replicator Service”](#).

- `--members=host2` [351]

Specifies the hostname of the slave (current) host where data will be applied.

- `--replication-user=tungsten` [363]

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

- `--replication-password=password` [363]

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

- `--property=replicator.datasource.global.csvType=hive`

Defines the default format for the CSV files that are generated. This setting configures the field, line and quoting used to format the CSV file.

- `--skip-validation-check=HostsFileCheck, --skip-validation-check=InstallerMasterSlaveCheck, --skip-validation-check=Direct-DatasourceDBPort, --skip-validation-check=DatasourceDBPort`

Configures certain validation checks not to be executed. These would otherwise cause installation to fail, as they check parameters specific to other database implementations.

- `--rmi-port=25550` [363]

Within Hadoop, Hive normally listens on port 10000. To prevent issues with starting the replicator on the Hadoop side, the port for RMI management should be changed to another free port number.

- `--start=true` [365]

Start the replicator service once it has been configured and installed.

To configure a replicator that writes directly into HDFS, the following options should be changed to use the value hdfs :

```
--batch-load-template=hdfs \
--datasource-type=hdfs \
```

And the following option, pointing to the location of the Hadoop client libraries added to the configuration:

```
--java-external-lib-dir=/usr/share/cmf/lib \
```

There are optional parameters that can be added to this configuration to set alternative settings and options.

Click the icon to show the detailed list of optional parameters:

For a list of all the optional parameters available, click the icon:

- `--property=replicator.datasource.applier.csv.fieldSeparator=\u0001`

This `replicator.datasource.applier.csv.fieldSeparator` configures the character that will be used to separate fields. On the command, this must be properly escape to ensure that the character given is used correctly. For example:

- `--property=replicator.datasource.applier.csv.fieldSeparator=\u0001`
- `--property=replicator.datasource.applier.csv.recordSeparator=\u0001`

This parameter configures the character that will be used to separate records in the generated staging files.

- `--property=replicator.datasource.applier.csv.useQuotes=false`

This parameter configures whether individual fields are quoted to contain their value. Use of quotes is optional.

- `--property=replicator.stage.q-to-dbms.blockCommitInterval=1s`

Information is committed in blocks, either according to the elapsed time, or the number of rows committed. This parameter configures the commit internal in seconds. For more information, see [Section 12.1, “Block Commit”](#).

- `--property=replicator.stage.q-to-dbms.blockCommitRowCount=1000`

This parameter configures the number of rows on which data will be written. For more information, see [Section 12.1, “Block Commit”](#).

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

Once the service has been installed it can be monitored using the `trepctl` command. See [Section 6.2.6, “Management and Monitoring of Hadoop Deployments”](#) for more information. If there are problems during installation, see [Section 6.2.6.1, “Troubleshooting Hadoop Replication”](#).

6.2.3.3.2. Slave Replicator Service (Cloudera)

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

Note

Installation must currently take place on a node within the Hadoop cluster. Writing to a remote HDFS filesystem is not currently supported.

The `tpm` required to install the slave replicator:

1. Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-5.0.tar.gz
```

2. Change into the staging directory:

```
shell> cd tungsten-replicator-5.0
```

3. Execute the `tpm` to perform the installation. The `tpm` command shown below configures a loading mechanism using files that are copied into HDFS by the replicator.

```
shell> ./tools/tpm install alpha \
--batch-enabled=true \
--batch-load-language=js \
--batch-load-template=hadoop \
--datasource-type=file \
--install-directory=/opt/continuent \
--master=host1 \
--members=host2 \
--property=replicator.datasource.applier.csvType=hive \
--property=replicator.stage.q-to-dbms.blockCommitInterval=1s \
--property=replicator.stage.q-to-dbms.blockCommitRowCount=1000 \
--replication-password=secret \
--replication-user=tungsten \
--skip-validation-check=DatasourceDBPort \
--skip-validation-check=DirectDatasourceDBPort \
--skip-validation-check=HostsFileCheck \
--skip-validation-check=InstallerMasterSlaveCheck \
--skip-validation-check=ReplicationServicePipelines \
--rmi-port=25550 \
--start-and-report=true
```

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

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

- [tpm install](#)

Executes `tpm` in `install` mode to create the service `alpha`.

- `--batch-enabled=true [331], --batch-load-language=js [331], --batch-load-template=hdfs [331]`

Enables batch-loading, using JavaScript as the batch loading language, writing to Hadoop or directly in to HDFS. The `--batch-load-template [331]` supports two options, `hadoop` and `hdfs`. In `hadoop` mode, files are written to disk and then copied to HDFS using the `hadoop dfs` command.

When using the `hdfs` mode, files are written directly into the HDFS filesystem by connecting to the HDFS service. When using the `hdfs` method the `--java-external-lib-dir [348]` option must also be specified, pointing to a valid Hadoop client library directory where the HDFS and Hadoop JAR files are located.

- `--datasource-type=file [340]`

Defines the format of the information to be generated when used through the batch loader. When using the `hadoop` batch load template (`--batch-load-template [331]`, this option should be set to `file`).

When using the `hdfs` template, the `--datasource-type [340]` option should be set to `hdfs`.

- `--install-directory=/opt/continuent [347]`

Directory where Tungsten Replication will be installed.

- `--java-external-lib-dir=/usr/lib/hadoop/client [348]`

Location of additional libraries to be added to the installation. This should point to the location of the Hadoop client libraries. In a default Cloudera installation this is located in `/usr/lib/hadoop/client`.

- `--master=host1 [351]`

Specifies the master host where THL data will be read from, and should point to the server configured in [Section 6.2.3.3, “Slave Replicator Service”](#).

- `--members=host2 [351]`

Specifies the hostname of the slave (current) host where data will be applied.

- `--replication-user=tungsten [363]`

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

- `--replication-password=password [363]`

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

- `--property=replicator.datasource.applier.csvType=hive`

Defines the default format for the CSV files that are generated. This setting configures the field, line and quoting used to format the CSV file.

- `--skip-validation-check=HostsFileCheck, --skip-validation-check=InstallerMasterSlaveCheck, --skip-validation-check=Direct-DatasourceDBPort, --skip-validation-check=DatasourceDBPort`

Configures certain validation checks not to be executed. These would otherwise cause installation to fail, as they check parameters specific to other database implementations.

- `--rmi-port=25550 [363]`

Within Hadoop, Hive normally listens on port 10000. To prevent issues with starting the replicator on the Hadoop side, the port for RMI management should be changed to another free port number.

- `--start=true [365]`

Start the replicator service once it has been configured and installed.

To configure a replicator that writes directly into HDFS, the following options should be changed to use the value `hdfs`:

```
--batch-load-template=hdfs \
--datasource-type=hdfs \
```

And the following option, pointing to the location of the Hadoop client libraries added to the configuration:

```
--java-external-lib-dir=/usr/share/cmf/lib \
```

There are optional parameters that can be added to this configuration to set alternative settings and options.

Click the icon to show the detailed list of optional parameters:

For a list of all the optional parameters available, click the icon:

- `--property=replicator.datasource.applier.csv.fieldSeparator=\u0001`

This `replicator.datasource.applier.csv.fieldSeparator` configures the character that will be used to separate fields. On the command, this must be properly escape to ensure that the character given is used correctly. For example:

```
--property=replicator.datasource.applier.csv.fieldSeparator=\u0001'
```

- `--property=replicator.datasource.applier.csv.recordSeparator=\u0001`

This parameter configures the character that will be used to separate records in the generated staging files.

- `--property=replicator.datasource.applier.csv.useQuotes=false`

This parameter configures whether individual fields are quoted to contain their value. Use of quotes is optional.

- `--property=replicator.stage.q-to-dbms.blockCommitInterval=1s`

Information is committed in blocks, either according to the elapsed time, or the number of rows committed. This parameter configures the commit internal in seconds. For more information, see [Section 12.1, “Block Commit”](#).

- `--property=replicator.stage.q-to-dbms.blockCommitRowCount=1000`

This parameter configures the number of rows on which data will be written. For more information, see [Section 12.1, “Block Commit”](#).

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

Once the service has been installed it can be monitored using the `trepctl` command. See [Section 6.2.6, “Management and Monitoring of Hadoop Deployments”](#) for more information. If there are problems during installation, see [Section 6.2.6.1, “Troubleshooting Hadoop Replication”](#).

6.2.3.3. Slave Replicator Service (Amazon EMR)

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

Note

Installation must currently take place on a node within the Hadoop cluster. Writing to a remote HDFS filesystem is not currently supported.

The `tpm` required to install the slave replicator:

- Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-5.0.tar.gz
```

- Change into the staging directory:

```
shell> cd tungsten-replicator-5.0
```

- Execute the `tpm` to perform the installation. The `tpm` command shown below configures a loading mechanism using files that are copied into HDFS by the replicator.

```
shell> ./tools/tpm install alpha \
--batch-enabled=true \
--batch-load-language=js \
--batch-load-template=hadoop \
--datasource-type=file \
--install-directory=/opt/continuent \
--master=host1 \
--members=host2 \
--property=replicator.datasource.applier.csvType=hive \
--property=replicator.stage.q-to-dbms.blockCommitInterval=1s \
--property=replicator.stage.q-to-dbms.blockCommitRowCount=1000 \
--replication-password=secret \
--replication-user=tungsten \
--skip-validation-check=DatasourceDBPort \
--skip-validation-check=DirectDatasourceDBPort \
--skip-validation-check=HostsFileCheck \
--skip-validation-check=InstallerMasterSlaveCheck \
```

```
--skip-validation-check=ReplicationServicePipelines \
--rmi-port=25550 \
--start-and-report=true
```

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

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

- **[tpm install](#)**

Executes **tpm** in **install** mode to create the service **alpha**.

- [--batch-enabled=true \[331\]](#), [--batch-load-language=js \[331\]](#), [--batch-load-template=hdfs \[331\]](#)

Enables batch-loading, using JavaScript as the batch loading language, writing to Hadoop or directly in to HDFS. The [--batch-load-template \[331\]](#) supports two options, hadoop and hdfs . In hadoop mode, files are written to disk and then copied to HDFS using the **hadoop dfs** command.

When using the hdfs mode, files are written directly into the HDFS filesystem by connecting to the HDFS service. When using the hdfs method the [--java-external-lib-dir \[348\]](#) option must also be specified, pointing to a valid Hadoop client library directory where the HDFS and Hadoop JAR files are located.

- [--datasource-type=file \[340\]](#)

Defines the format of the information to be generated when used through the batch loader. When using the hadoop batch load template ([--batch-load-template \[331\]](#), this option should be set to file .

When using the hdfs template, the [--datasource-type \[340\]](#) option should be set to hdfs .

- [--install-directory=/opt/continuent \[347\]](#)

Directory where Tungsten Replication will be installed.

- [--java-external-lib-dir=/usr/lib/hadoop/client \[348\]](#)

Location of additional libraries to be added to the installation. This should point to the location of the Hadoop client libraries. In a default Cloudera installation this is located in [/usr/lib/hadoop/client](#).

- [--master=host1 \[351\]](#)

Specifies the master host where THL data will be read from, and should point to the server configured in [Section 6.2.3.3, "Slave Replicator Service"](#).

- [--members=host2 \[351\]](#)

Specifies the hostname of the slave (current) host where data will be applied.

- [--replication-user=tungsten \[363\]](#)

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

- [--replication-password=password \[363\]](#)

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

- [--property=replicator.datasource.applier.csvType=hive](#)

Defines the default format for the CSV files that are generated. This setting configures the field, line and quoting used to format the CSV file.

- [--skip-validation-check=HostsFileCheck](#), [--skip-validation-check=InstallerMasterSlaveCheck](#), [--skip-validation-check=Direct-DatasourceDBPort](#), [--skip-validation-check=DatasourceDBPort](#)

Configures certain validation checks not to be executed. These would otherwise cause installation to fail, as they check parameters specific to other database implementations.

- [--rmi-port=25550 \[363\]](#)

Within Hadoop, Hive normally listens on port 10000. To prevent issues with starting the replicator on the Hadoop side, the port for RMI management should be changed to another free port number.

- [--start=true \[365\]](#)

Start the replicator service once it has been configured and installed.

To configure a replicator that writes directly into HDFS, the following options should be changed to use the value hdfs :

```
--batch-load-template=hdfs \
--datasource-type=hdfs \
```

And the following option, pointing to the location of the Hadoop client libraries added to the configuration:

```
--java-external-lib-dir=/usr/share/cmf/lib \
```

There are optional parameters that can be added to this configuration to set alternative settings and options.

Click the icon to show the detailed list of optional parameters:

For a list of all the optional parameters available, click the icon:

- `--property=replicator.datasource.applier.csv.fieldSeparator=\u0001`

This `replicator.datasource.applier.csv.fieldSeparator` configures the character that will be used to separate fields. On the command, this must be properly escape to ensure that the character given is used correctly. For example:

```
--property=replicator.datasource.applier.csv.fieldSeparator=\u0001
```

- `--property=replicator.datasource.applier.csv.recordSeparator=\u0001`

This parameter configures the character that will be used to separate records in the generated staging files.

- `--property=replicator.datasource.applier.csv.useQuotes=false`

This parameter configures whether individual fields are quoted to contain their value. Use of quotes is optional.

- `--property=replicator.stage.q-to-dbms.blockCommitInterval=1s`

Information is committed in blocks, either according to the elapsed time, or the number of rows committed. This parameter configures the commit internal in seconds. For more information, see [Section 12.1, “Block Commit”](#).

- `--property=replicator.stage.q-to-dbms.blockCommitRowCount=1000`

This parameter configures the number of rows on which data will be written. For more information, see [Section 12.1, “Block Commit”](#).

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

Once the service has been installed it can be monitored using the `trepctl` command. See [Section 6.2.6, “Management and Monitoring of Hadoop Deployments”](#) for more information. If there are problems during installation, see [Section 6.2.6.1, “Troubleshooting Hadoop Replication”](#).

6.2.3.3.4. Slave Replicator Service (HortonWorks)

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

Note

Installation must currently take place on a node within the Hadoop cluster. Writing to a remote HDFS filesystem is not currently supported.

The `tpm` required to install the slave replicator:

1. Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-5.0.tar.gz
```

2. Change into the staging directory:

```
shell> cd tungsten-replicator-5.0
```

3. Execute the `tpm` to perform the installation. The `tpm` command shown below configures a loading mechanism using files that are copied into HDFS by the replicator.

```
shell> ./tools/tpm install alpha \
--batch-enabled=true \
--batch-load-language=js \
--batch-load-template=hadoop \
--datasource-type=file \
--install-directory=/opt/continuent \
--master=host1 \
```

```
--members=host2 \
--property=replicator.datasource.applier.csvType=hive \
--property=replicator.stage.q-to-dbms.blockCommitInterval=1s \
--property=replicator.stage.q-to-dbms.blockCommitRowCount=1000 \
--replication-password=secret \
--replication-user=tungsten \
--skip-validation-check=DatasourceDBPort \
--skip-validation-check=DirectDatasourceDBPort \
--skip-validation-check=HostsFileCheck \
--skip-validation-check=InstallerMasterSlaveCheck \
--skip-validation-check=ReplicationServicePipelines \
--rmi-port=25550 \
--start-and-report=true
```

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

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

- **[tpm install](#)**

Executes [tpm](#) in [install](#) mode to create the service [alpha](#).

- **[--batch-enabled=true \[331\]](#), [--batch-load-language=js \[331\]](#), [--batch-load-template=hdfs \[331\]](#)**

Enables batch-loading, using JavaScript as the batch loading language, writing to Hadoop or directly in to HDFS. The [--batch-load-template \[331\]](#) supports two options, hadoop and hdfs . In hadoop mode, files are written to disk and then copied to HDFS using the [hadoop dfs](#) command.

When using the hdfs mode, files are written directly into the HDFS filesystem by connecting to the HDFS service. When using the hdfs method the [--java-external-lib-dir \[348\]](#) option must also be specified, pointing to a valid Hadoop client library directory where the HDFS and Hadoop JAR files are located.

- **[--datasource-type=file \[340\]](#)**

Defines the format of the information to be generated when used through the batch loader. When using the hadoop batch load template ([--batch-load-template \[331\]](#), this option should be set to file .

When using the hdfs template, the [--datasource-type \[340\]](#) option should be set to hdfs .

- **[--install-directory=/opt/continuent \[347\]](#)**

Directory where Tungsten Replication will be installed.

- **[--java-external-lib-dir=/usr/lib/hadoop/client \[348\]](#)**

Location of additional libraries to be added to the installation. This should point to the location of the Hadoop client libraries. In a default Cloudera installation this is located in [/usr/lib/hadoop/client](#).

- **[--master=host1 \[351\]](#)**

Specifies the master host where THL data will be read from, and should point to the server configured in [Section 6.2.3.3, “Slave Replicator Service”](#).

- **[--members=host2 \[351\]](#)**

Specifies the hostname of the slave (current) host where data will be applied.

- **[--replication-user=tungsten \[363\]](#)**

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

- **[--replication-password=password \[363\]](#)**

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

- **[--property=replicator.datasource.applier.csvType=hive](#)**

Defines the default format for the CSV files that are generated. This setting configures the field, line and quoting used to format the CSV file.

- **[--skip-validation-check=HostsFileCheck, --skip-validation-check=InstallerMasterSlaveCheck, --skip-validation-check=Direct-DatasourceDBPort, --skip-validation-check=DatasourceDBPort](#)**

Configures certain validation checks not to be executed. These would otherwise cause installation to fail, as they check parameters specific to other database implementations.

- `--rmi-port=25550` [363]

Within Hadoop, Hive normally listens on port 10000. To prevent issues with starting the replicator on the Hadoop side, the port for RMI management should be changed to another free port number.

- `--start=true` [365]

Start the replicator service once it has been configured and installed.

To configure a replicator that writes directly into HDFS, the following options should be changed to use the value `hdfs`:

```
--batch-load-template=hdfs \
--datasource-type=hdfs \
```

And the following option, pointing to the location of the Hadoop client libraries added to the configuration:

```
--java-external-lib-dir=/usr/share/cmflib \
```

There are optional parameters that can be added to this configuration to set alternative settings and options.

Click the icon to show the detailed list of optional parameters:

For a list of all the optional parameters available, click the icon:

- `--property=replicator.datasource.applier.csv.fieldSeparator=\u0001`

This `replicator.datasource.applier.csv.fieldSeparator` configures the character that will be used to separate fields. On the command, this must be properly escape to ensure that the character given is used correctly. For example:

```
--property=replicator.datasource.applier.csv.fieldSeparator=\u0001*
```

- `--property=replicator.datasource.applier.csv.recordSeparator=\u0001`

This parameter configures the character that will be used to separate records in the generated staging files.

- `--property=replicator.datasource.applier.csv.useQuotes=false`

This parameter configures whether individual fields are quoted to contain their value. Use of quotes is optional.

- `--property=replicator.stage.q-to-dbms.blockCommitInterval=1s`

Information is committed in blocks, either according to the elapsed time, or the number of rows committed. This parameter configures the commit internal in seconds. For more information, see [Section 12.1, “Block Commit”](#).

- `--property=replicator.stage.q-to-dbms.blockCommitRowCount=1000`

This parameter configures the number of rows on which data will be written. For more information, see [Section 12.1, “Block Commit”](#).

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

Once the service has been installed it can be monitored using the `trepctl` command. See [Section 6.2.6, “Management and Monitoring of Hadoop Deployments”](#) for more information. If there are problems during installation, see [Section 6.2.6.1, “Troubleshooting Hadoop Replication”](#).

6.2.3.3.5. Slave Replicator Service (IBM InfoSphere BigInsights)

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

Note

Installation must currently take place on a node within the Hadoop cluster. Writing to a remote HDFS filesystem is not currently supported.

The `tpm` required to install the slave replicator:

1. Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-5.0.tar.gz
```

2. Change into the staging directory:

```
shell> cd tungsten-replicator-5.0
```

3. Execute the `tpm` to perform the installation. The `tpm` command shown below configures a loading mechanism using files that are copied into HDFS by the replicator.

```
shell> ./tools/tpm install alpha \
--batch-enabled=true \
--batch-load-language=js \
--batch-load-template=hadoop \
--datasource-type=file \
--install-directory=/opt/continuent \
--master=host1 \
--members=host2 \
--property=replicator.datasource.applier.csvType=hive \
--property=replicator.stage.q-to-dbms.blockCommitInterval=1s \
--property=replicator.stage.q-to-dbms.blockCommitRowCount=1000 \
--replication-password=secret \
--replication-user=tungsten \
--skip-validation-check=DatasourceDBPort \
--skip-validation-check=DirectDatasourceDBPort \
--skip-validation-check=HostsFileCheck \
--skip-validation-check=InstallerMasterSlaveCheck \
--skip-validation-check=ReplicationServicePipelines \
--rmi-port=25550 \
--start-and-report=true
```

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

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

- **`tpm install`**

Executes `tpm` in `install` mode to create the service `alpha`.

- **`--batch-enabled=true [331]`, `--batch-load-language=js [331]`, `--batch-load-template=hdfs [331]`**

Enables batch-loading, using JavaScript as the batch loading language, writing to Hadoop or directly in to HDFS. The `--batch-load-template [331]` supports two options, `hadoop` and `hdfs`. In `hadoop` mode, files are written to disk and then copied to HDFS using the `hadoop dfs` command.

When using the `hdfs` mode, files are written directly into the HDFS filesystem by connecting to the HDFS service. When using the `hdfs` method the `--java-external-lib-dir [348]` option must also be specified, pointing to a valid Hadoop client library directory where the HDFS and Hadoop JAR files are located.

- **`--datasource-type=file [340]`**

Defines the format of the information to be generated when used through the batch loader. When using the hadoop batch load template (`--batch-load-template [331]`, this option should be set to `file`.

When using the `hdfs` template, the `--datasource-type [340]` option should be set to `hdfs`.

- **`--install-directory=/opt/continuent [347]`**

Directory where Tungsten Replication will be installed.

- **`--java-external-lib-dir=/usr/lib/hadoop/client [348]`**

Location of additional libraries to be added to the installation. This should point to the location of the Hadoop client libraries. In a default Cloudera installation this is located in `/usr/lib/hadoop/client`.

- **`--master=host1 [351]`**

Specifies the master host where THL data will be read from, and should point to the server configured in [Section 6.2.3.3, “Slave Replicator Service”](#).

- **`--members=host2 [351]`**

Specifies the hostname of the slave (current) host where data will be applied.

- **`--replication-user=tungsten [363]`**

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

- **`--replication-password=password [363]`**

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

- `--property=replicator.datasource.applier.csvType=hive`

Defines the default format for the CSV files that are generated. This setting configures the field, line and quoting used to format the CSV file.

- `--skip-validation-check=HostsFileCheck, --skip-validation-check=InstallerMasterSlaveCheck, --skip-validation-check=Direct-DatasourceDBPort, --skip-validation-check=DatasourceDBPort`

Configures certain validation checks not to be executed. These would otherwise cause installation to fail, as they check parameters specific to other database implementations.

- `--rmi-port=25550 [363]`

Within Hadoop, Hive normally listens on port 10000. To prevent issues with starting the replicator on the Hadoop side, the port for RMI management should be changed to another free port number.

- `--start=true [365]`

Start the replicator service once it has been configured and installed.

To configure a replicator that writes directly into HDFS, the following options should be changed to use the value hdfs :

```
--batch-load-template=hdfs \
--datasource-type=hdfs \
```

And the following option, pointing to the location of the Hadoop client libraries added to the configuration:

```
--java-external-lib-dir=/usr/share/cmf/lib \
```

There are optional parameters that can be added to this configuration to set alternative settings and options.

Click the icon to show the detailed list of optional parameters:

For a list of all the optional parameters available, click the icon:

- `--property=replicator.datasource.applier.csv.fieldSeparator=\u0001`

This `replicator.datasource.applier.csv.fieldSeparator` configures the character that will be used to separate fields. On the command, this must be properly escape to ensure that the character given is used correctly. For example:

```
--property=replicator.datasource.applier.csv.fieldSeparator=\u0001'
```

- `--property=replicator.datasource.applier.csv.recordSeparator=\u0001`

This parameter configures the character that will be used to separate records in the generated staging files.

- `--property=replicator.datasource.applier.csv.useQuotes=false`

This parameter configures whether individual fields are quoted to contain their value. Use of quotes is optional.

- `--property=replicator.stage.q-to-dbms.blockCommitInterval=1s`

Information is committed in blocks, either according to the elapsed time, or the number of rows committed. This parameter configures the commit internal in seconds. For more information, see [Section 12.1, “Block Commit”](#).

- `--property=replicator.stage.q-to-dbms.blockCommitRowCount=1000`

This parameter configures the number of rows on which data will be written. For more information, see [Section 12.1, “Block Commit”](#).

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

Once the service has been installed it can be monitored using the `trepctl` command. See [Section 6.2.6, “Management and Monitoring of Hadoop Deployments”](#) for more information. If there are problems during installation, see [Section 6.2.6.1, “Troubleshooting Hadoop Replication”](#).

6.2.3.3.6. Slave Replicator Service (MapR)

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

Note

Installation must currently take place on a node within the Hadoop cluster. Writing to a remote HDFS filesystem is not currently supported.

The **tpm** required to install the slave replicator:

1. Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-5.0.tar.gz
```

2. Change into the staging directory:

```
shell> cd tungsten-replicator-5.0
```

3. Execute the **tpm** to perform the installation. The **tpm** command shown below configures a loading mechanism using files that are copied into HDFS by the replicator.

```
shell> ./tools/tpm install alpha \
--batch-enabled=true \
--batch-load-language=js \
--batch-load-template=hadoop \
--datasource-type=file \
--install-directory=/opt/continuent \
--master=host1 \
--members=host2 \
--property=replicator.datasource.global.csvType=hive \
--property=replicator.stage.q-to-dbms.blockCommitInterval=1s \
--property=replicator.stage.q-to-dbms.blockCommitRowCount=1000 \
--replication-password=secret \
--replication-user=tungsten \
--skip-validation-check=DatasourceDBPort \
--skip-validation-check=DirectDatasourceDBPort \
--skip-validation-check=HostsFileCheck \
--skip-validation-check=InstallerMasterSlaveCheck \
--skip-validation-check=ReplicationServicePipelines \
--rmi-port=25550 \
--start-and-report=true
```

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

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

- **tpm install**

Executes **tpm** in **install** mode to create the service **alpha**.

- **--batch-enabled=true [331]**, **--batch-load-language=js [331]**, **--batch-load-template=hdfs [331]**

Enables batch-loading, using JavaScript as the batch loading language, writing to Hadoop or directly in to HDFS. The **--batch-load-template [331]** supports two options, **hadoop** and **hdfs**. In **hadoop** mode, files are written to disk and then copied to HDFS using the **hadoop dfs** command.

When using the **hdfs** mode, files are written directly into the HDFS filesystem by connecting to the HDFS service. When using the **hdfs** method the **--java-external-lib-dir [348]** option must also be specified, pointing to a valid Hadoop client library directory where the HDFS and Hadoop JAR files are located.

- **--datasource-type=file [340]**

Defines the format of the information to be generated when used through the batch loader. When using the hadoop batch load template (**--batch-load-template [331]**), this option should be set to file .

When using the **hdfs** template, the **--datasource-type [340]** option should be set to **hdfs**.

- **--install-directory=/opt/continuent [347]**

Directory where Tungsten Replication will be installed.

- **--java-external-lib-dir=/usr/lib/hadoop/client [348]**

Location of additional libraries to be added to the installation. This should point to the location of the Hadoop client libraries. In a default Cloudera installation this is located in **/usr/lib/hadoop/client**.

- **--master=host1 [351]**

Specifies the master host where THL data will be read from, and should point to the server configured in [Section 6.2.3.3, “Slave Replicator Service”](#).

- `--members=host2 [351]`

Specifies the hostname of the slave (current) host where data will be applied.

- `--replication-user=tungsten [363]`

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

- `--replication-password=password [363]`

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

- `--property=replicator.datasource.global.csvType=hive`

Defines the default format for the CSV files that are generated. This setting configures the field, line and quoting used to format the CSV file.

- `--skip-validation-check=HostsFileCheck, --skip-validation-check=InstallerMasterSlaveCheck, --skip-validation-check=Direct-DatasourceDBPort, --skip-validation-check=DatasourceDBPort`

Configures certain validation checks not to be executed. These would otherwise cause installation to fail, as they check parameters specific to other database implementations.

- `--rmi-port=25550 [363]`

Within Hadoop, Hive normally listens on port 10000. To prevent issues with starting the replicator on the Hadoop side, the port for RMI management should be changed to another free port number.

- `--start=true [365]`

Start the replicator service once it has been configured and installed.

To configure a replicator that writes directly into HDFS, the following options should be changed to use the value hdfs:

```
--batch-load-template=hdfs \
--datasource-type=hdfs \
```

And the following option, pointing to the location of the Hadoop client libraries added to the configuration:

```
--java-external-lib-dir=/usr/share/cmf/lib \
```

There are optional parameters that can be added to this configuration to set alternative settings and options.

Click the icon to show the detailed list of optional parameters:

For a list of all the optional parameters available, click the icon:

- `--property=replicator.datasource.global.csv.fieldSeparator=\u0001`

This `replicator.datasource.global.csv.fieldSeparator` configures the character that will be used to separate fields. On the command, this must be properly escape to ensure that the character given is used correctly. For example:

```
--property=replicator.datasource.global.csv.fieldSeparator=\u0001'
```

- `--property=replicator.datasource.global.csv.recordSeparator=\u0001`

This parameter configures the character that will be used to separate records in the generated staging files.

- `--property=replicator.datasource.global.csv.useQuotes=false`

This parameter configures whether individual fields are quoted to contain their value. Use of quotes is optional.

- `--property=replicator.stage.q-to-dbms.blockCommitInterval=1s`

Information is committed in blocks, either according to the elapsed time, or the number of rows committed. This parameter configures the commit internal in seconds. For more information, see [Section 12.1, “Block Commit”](#).

- `--property=replicator.stage.q-to-dbms.blockCommitRowCount=1000`

This parameter configures the number of rows on which data will be written. For more information, see [Section 12.1, “Block Commit”](#).

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

Once the service has been installed it can be monitored using the `trepctl` command. See [Section 6.2.6, “Management and Monitoring of Hadoop Deployments”](#) for more information. If there are problems during installation, see [Section 6.2.6.1, “Troubleshooting Hadoop Replication”](#).

6.2.4. Generating Materialized Views

The `continuent-tools-hadoop` repository contains a set of tools that allow for the convenient creation of DDL, materialized views, and data comparison on the tables that have been replicated from MySQL.

To obtain the tools, use `git`

```
shell> ./bin/load-reduce-check -s test -Ujdbc:mysql:thin://tr-hadoop2:13306 -udbload -ppassword
```

The `load-reduce-check` command performs four distinct steps:

1. Reads the schema from the MySQL server and creates the staging table DDL within Hive
2. Reads the schema from the MySQL server and creates the base table DDL within Hive
3. Executes the materialized view process on each selected staging table data to build the base table content.
4. Performs a data comparison

6.2.5. Accessing Generated Tables in Hive

If not already completed, the schema generation process described in [Section 6.2.2.3, “Schema Generation”](#) should have been followed. This creates the necessary Hive schema and staging schema definitions.

Once the tables have been created through `ddlscan` you can query the stage tables:

```
hive> select * from stage_xxx_movies_large limit 10;
OK
I 10 1 57475 All in the Family 1971 Archie Feels Left Out (#4.17)
I 10 2 57476 All in the Family 1971 Archie Finds a Friend (#6.18)
I 10 3 57477 All in the Family 1971 Archie Gets the Business: Part 1 (#8.1)
I 10 4 57478 All in the Family 1971 Archie Gets the Business: Part 2 (#8.2)
I 10 5 57479 All in the Family 1971 Archie Gives Blood (#1.4)
I 10 6 57480 All in the Family 1971 Archie Goes Too Far (#3.17)
I 10 7 57481 All in the Family 1971 Archie in the Cellar (#4.10)
I 10 8 57482 All in the Family 1971 Archie in the Hospital (#3.15)
I 10 9 57483 All in the Family 1971 Archie in the Lock-Up (#2.3)
I 10 10 57484 All in the Family 1971 Archie Is Branded (#3.20)
```

6.2.6. Management and Monitoring of Hadoop Deployments

Once the two services — extractor and applier — have been installed, the services can be monitored using `trepctl`. To monitor the master (extractor) service:

```
shell> trepctl status
Processing status command...
NAME          VALUE
-----
appliedLastEventId : mysql-bin.000023:000000505545003;0
appliedLastSeqno : 10992
appliedLatency  : 42.764
channels       : 1
clusterName    : alpha
currentEventId : mysql-bin.000023:000000505545003
currentTimeMillis : 1389871897922
dataServerHost : host1
extensions     :
host           : host1
latestEpochNumber : 0
masterConnectUri : thl://localhost:/
masterListenUri : thl://host1:2112/
maximumStoredSeqNo : 10992
minimumStoredSeqNo : 0
offlineRequests : NONE
pendingError    : NONE
pendingErrorCode: NONE
pendingErrorEventId : NONE
pendingErrorSeqno : -1
pendingExceptionMessage: NONE
pipelineSource  : jdbc:mysql:thin://host1:13306/
relativeLatency : 158296.922
resourcePrecedence : 99
```

```
rmiPort : 10000
role : master
seqnoType : java.lang.Long
serviceName : alpha
serviceType : local
simpleServiceName : alpha
siteName : default
sourceId : host1
state : ONLINE
timeInStateSeconds : 165845.474
transitioningTo :
uptimeSeconds : 165850.047
useSSLConnection : false
version : Tungsten Replicator 5.0.1 build 136
Finished status command...
```

On the slave, `trepctl status` shows the currently applied into Hadoop:

```
shell> trepctl status
Processing status command...
NAME          VALUE
----          -----
appliedLastEventId : mysql-bin.000010:000000349816178;0
appliedLastSeqno : 1102
appliedLatency : 57.109
channels : 1
clusterName : alpha
currentEventId : NONE
currentTimeMillis : 1389629684476
dataServerHost : 192.168.1.252
extensions :
host : 192.168.1.252
latestEpochNumber : 236
masterConnectUri : th1://host1:2112/
masterListenUri : null
maximumStoredSeqNo : 1102
minimumStoredSeqNo : 0
offlineRequests : NONE
pendingError : NONE
pendingErrorCode : NONE
pendingErrorEventId : NONE
pendingErrorSeqno : -1
pendingExceptionMessage: NONE
pipelineSource : th1://host1:2112/
relativeLatency : 121.476
resourcePrecedence : 99
rmiPort : 10002
role : slave
seqnoType : java.lang.Long
serviceName : alpha
serviceType : local
simpleServiceName : alpha
siteName : default
sourceId : 192.168.1.252
state : ONLINE
timeInStateSeconds : 9690.134
transitioningTo :
uptimeSeconds : 10734.015
useSSLConnection : false
version : Tungsten Replicator 5.0.1 build 136
```

6.2.6.1. Troubleshooting Hadoop Replication

Replicating to Hadoop involves a number of discrete, specific steps. Due to the batch and multi-stage nature of the extract and apply process, replication can stall or stop due to a variety of issues.

6.2.6.1.1. Errors Reading/Writing `commitseqno.0` File

During initial installation, or when starting up replication, the replicator may report that the `commitseqno.0` can not be created or written properly, or during startup, that the file cannot be read.

The following checks and recovery procedures can be tried:

- Check the permissions of the directory to the `commitseqno.0` file, the file itself, and the ownership:

```
shell> hadoop fs -ls -R /user/tungsten/metadata
drwxr-xr-x - cloudera cloudera 0 2014-01-14 10:40 /user/tungsten/metadata/alpha
-rw-r--r-- 3 cloudera cloudera 251 2014-01-14 10:40 /user/tungsten/metadata/alpha/commitseqno.0
```

- Check that the file is writable and is not empty. An empty file may indicate a problem updating the content with the new sequence number.

- Check the content of the file is correct. The content should be a JSON structure containing the replicator state and position information. For example:

```
shell> hadoop fs -cat /user/tungsten/metadata/alpha/commitseqno.0
{
  "appliedLatency" : "0",
  "epochNumber" : "0",
  "fragno" : "0",
  "shardId" : "dna",
  "seqno" : "8",
  "eventId" : "mysql-bin.000015:000000000103156:0",
  "extractedTstamp" : "1389706078000",
  "lastFrag" : "true",
  "sourceId" : "host1"
}
```

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

```
shell> hadoop fs -rm /user/tungsten/metadata/alpha/commitseqno.0
shell> trepctl online
```

6.2.6.1.2. Recovering from Replication Failure

If the replication fails, is manually stopped, or the host needs to be restarted, replication should continue from the last point. When replication was stopped. Files that were being written when replication was last running will be overwritten and the information recreated.

Unlike other Heterogeneous replication implementations, the Hadoop applier stores the current replication state and restart position in a file within the HDFS of the target Hadoop environment. To recover from failed replication, this file must be deleted, so that the THL can be re-read from the master and CSV files will be recreated and applied into HDFS.

- On the Slave, put the replicator offline:

```
shell> trepctl offline
```

- Remove the THL files from the slave:

```
shell> trepctl reset -thl
```

- Remove the staging CSV files replicated into Hadoop:

```
shell> hadoop fs -rm -r /user/tungsten/staging
```

- Reset the restart position:

```
shell> rm /opt/continuent/tungsten/tungsten-replicator/data/alpha/commitseqno.0
```

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

- Restart replication on the slave; this will start to recreate the THL files from the MySQL binary log:

```
shell> trepctl online
```

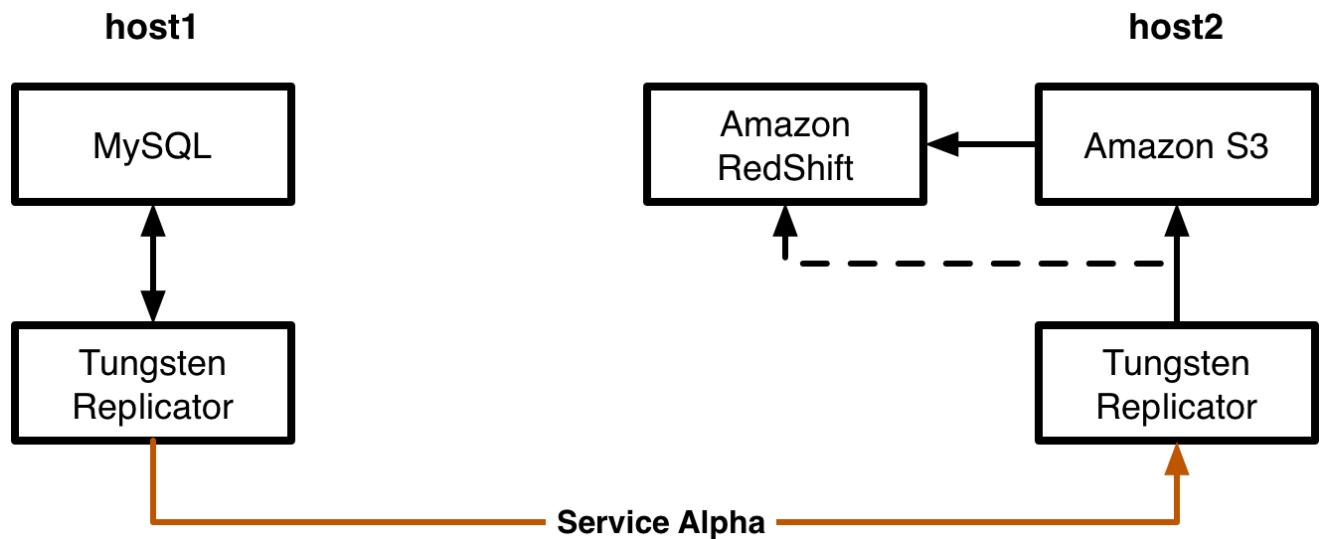
6.3. Deploying MySQL to Amazon Redshift Replication

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

Replication to Amazon Redshift operates as follows:

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

Figure 6.4. Topologies: MySQL to Amazon Redshift

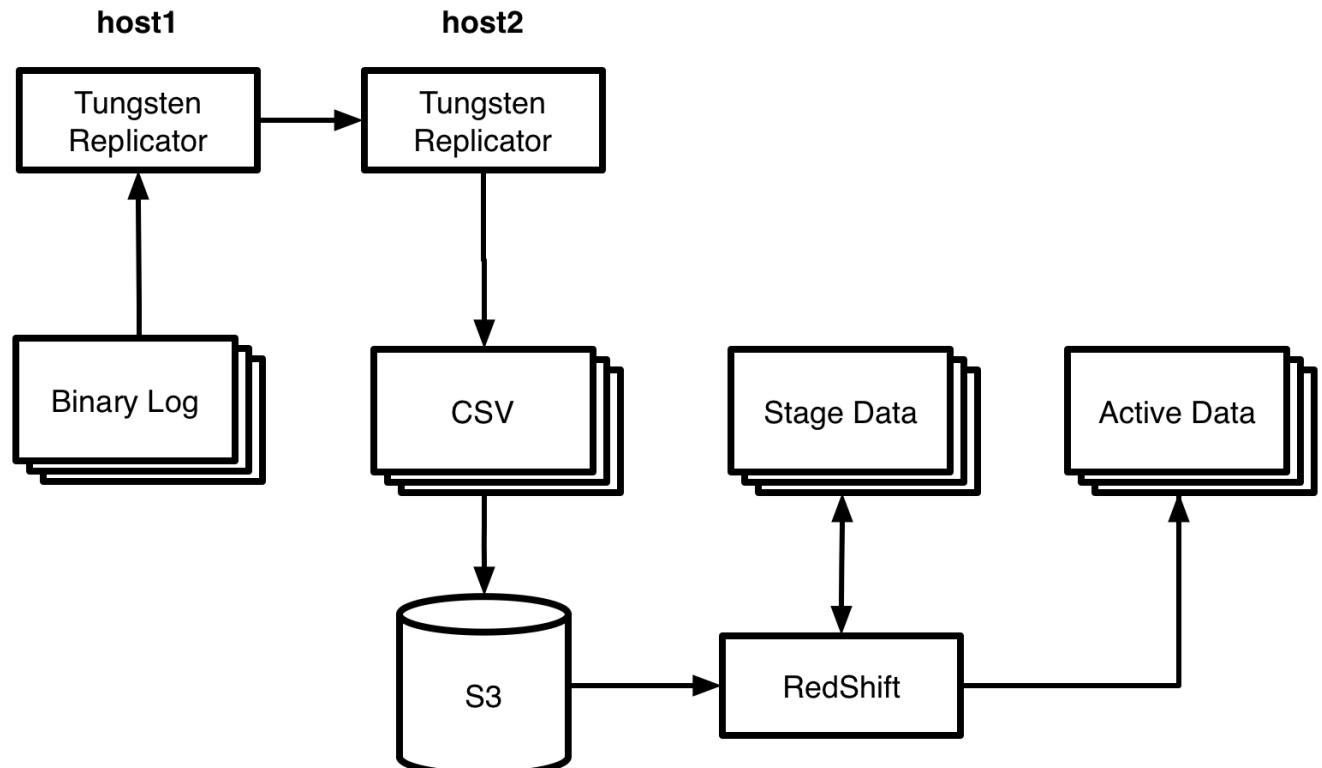


Setting up replication requires setting up both the master and slave components as two different configurations, one for MySQL and the other for Amazon Redshift. Replication also requires some additional steps to ensure that the Amazon Redshift host is ready to accept the replicated data that has been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

6.3.1. Redshift Replication Operation

The Redshift applier makes use of the JavaScript based batch loading system (see [Section 7.2.4, “JavaScript Batchloader Scripts”](#)). This constructs change data from the source-database, and uses this information in combination with any existing data to construct, using Hive, a materialized view. A summary of this basic structure can be seen in [Figure 6.5, “Topologies: MySQL to Redshift Replication Operation”](#).

Figure 6.5. Topologies: MySQL to Redshift Replication Operation



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

MySQL	Redshift
Instance	Database
Database	Schema
Table	Table

The full replication of information operates as follows:

1. Data is extracted from the source database using the standard extractor, for example by reading the row change data from the binlog in MySQL.
2. The [Section 11.4.8, “ColumnName Filter”](#) filter is used to extract column name information from the database. This enables the row-change information to be tagged with the corresponding column information. The data changes, and corresponding row names, are stored in the THL.

The [Section 11.4.29, “PrimaryKey Filter”](#) filter is used to extract primary key data from the source tables.

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

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

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

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

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

- The format of the files is a character separated values file, with each row separated by a newline, and individual fields separated by the character `0x01`. This is supported by Hive as a native value separator.
- The content of the file consists of the full row data extracted from the master, plus metadata describing the operation for each row, the sequence number, and then the full row information.

Operation	Sequence No	Table-specific primary key	DateTime	Table-columns...
OPTYPE	<code>SEQNO</code> [459] that generated this row	PRIMARYKEY	DATETIME of source table commit	

The operation field will match one of the following values

Operation	Description	Notes
I	Row is an <code>INSERT</code> of new data	
D	Row is <code>DELETE</code> of existing data	
UI	Row is an <code>UPDATE</code> which caused <code>INSERT</code> of data	
UD	Row is an <code>UPDATE</code> which caused <code>DELETE</code> of data	

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

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

Is represented within the CSV staging files generated as:

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

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

6.3.2. Preparing Hosts for Amazon Redshift Deployments

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

6.3.2.1. MySQL Preparation for Amazon Redshift Deployments

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

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

- MySQL must be using Row-based replication for information to be replicated to Amazon Redshift. For the best results, you should change the global binary log format, ideally in the configuration file (`my.cnf`):

```
binlog-format = row
```

Alternatively, the global binlog format can be changed by executing the following statement:

```
mysql> SET GLOBAL binlog-format = ROW;
```

For MySQL 5.6.2 and later, you must enable full row log images:

```
binlog-row-image = full
```

This information will be forgotten when the MySQL server is restarted; placing the configuration in the `my.cnf` file will ensure this option is permanently enabled.

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

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

Tables must also be configured as UTF8 tables, and existing tables should be updated to UTF8 support before they are replicated to prevent character set corruption issues.

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

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

6.3.2.2. Redshift Preparation for Amazon Redshift Deployments

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

- An existing Amazon Web Services (AWS) account, and the AWS Access Key and Secret Key required to interact with the account through the API.
- A configured Amazon S3 service. If the S3 service has not already been configured, visit the AWS console and sign up for the Amazon S3 service.
- The `s3cmd` installed and configured. The `s3cmd` can be downloaded from [s3cmd on s3tools.org](http://s3tools.org). To configure the command to automatically connect to the Amazon S3 service without requiring further authentication, the `.s3cfg` in the `tungsten` users home directory should be configured with the corresponding access key and secret key. For example:

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

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

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

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

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

- Create an `s3-config-servicename.json` file based on the sample provided within `cluster-home/samples/conf/s3-config-servicename.json` within the Tungsten Replication staging directory.

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

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

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

The allowed options for this file are as follows:

- `awsS3Path` — the location within your S3 storage where files should be loaded.
- `awsAccessKey` — the S3 access key to access your S3 storage.
- `awsSecretKey` — the S3 secret key to access your S3 storage.
- `cleanUpS3Files` — a boolean value used to identify whether the CSV files loaded into S3 should be deleted after they have been imported and merged. If set to true, the files are automatically deleted once the files have been successfully imported into the Redshift staging tables. If set to false, files are not automatically removed.
- `storeCDCIn` — a definition table that stores the change data from the load, in addition to importing to staging and base tables. The `{schema}` and `{table}` variables will be automatically replaced with the corresponding schema and table name. For more information on keeping CDC information, see [Section 6.3.5, “Keeping CDC Information”](#).

6.3.2.3. Amazon Redshift DDL Generation for Amazon Redshift Deployments

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

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

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

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

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

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

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

```
shell> ddlscan -db test -template ddl-mysql-redshift.vm
DROP TABLE test.msg;
CREATE TABLE test.msg
(
  id INT,
  msg CHAR(80),
  PRIMARY KEY (id)
);
```

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

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

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

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

6.3.3. Installing Amazon Redshift Replication

Replication into Redshift requires two separate replicator installations, one that extracts information from the source database, and a second that generates the CSV files, loads those files into S3 and then executes the statements on the Redshift database to import the CSV data and apply the transformations to build the final tables. These can either be two separate hosts, or configured to work within a single host.

Configure Defaults

The defaults configure the type of the services, topology and hosts in the overall service:

```
shell> ./tools/tpm configure defaults --reset
shell> ./tools/tpm configure alpha \
    --install-directory=/opt/continuent \
    --enable-heterogeneous-service=true \
    --members=host1,host2 \
    --master=host1
```

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` [347]

Directory where Tungsten Replication will be installed.

- `--enable-heterogeneous-service=true` [344]

Configures the service as heterogeneous. This enables certain filters and options designed to ensure that information is formatted and replicated properly between databases of different types, for example MySQL and Redshift.

- `--members=host1,host2` [351]

Defines the number of members in this cluster.

- `--master=host1` [351]

Specifies the host that will be the master in this cluster.

- `--java-user-timezone=GMT` [350]

Sets the timezone for the Java environment to GMT. Setting this, and setting the master and host timezones to the same value ensures that there are no time differences between hosts.

- `--java-file-encoding=UTF8` [348]

Enable UTF8 based file encoding. This is required to ensure that data is correctly stored in the THL in preparation for extraction by the Amazon Redshift replicator.

Configure Master Replicator Service

To configure the master replicator, which will extract information from MySQL into THL:

```
shell> ./tools/tpm configure alpha --hosts=host1 \
    --replication-user=tungsten \
    --replication-password=password \
    --property=replicator.filter.pkey.addColumnsToDelete=true \
    --property=replicator.filter.pkey.addKeyToInserts=true
```

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`

Executes `tpm` in `install` mode to create the service `alpha`. The use of the `--hosts` [347] option configures the parameters only for that host.

- `--replication-user=tungsten` [363]

The user that will be used to connect to the source database to record status information.

- `--replication-password=password` [363]

The password that will be used to connect to the source database to record status information.

- `--property=replicator.filter.pkey.addColumnstoDeletes=true`

Configure a property for the `pkey` filter that will add all columns from a row into the delete operation. This is required to ensure that the CSV row contains all the information during a delete operation.

- `--property=replicator.filter.pkey.addPkeyToInserts=true`

Configure a property for the `pkey` filter that will add the primary key information to all insert rows. This ensures that during an `INSERT` operation, the primary key information is included.

Configure the Amazon Redshift Replicator Service

Creating the the Amazon Redshift side of the process requires creating a slave to the master service created in the previous step, and configuring the correct applier and user/password combination.

1. Use `tpm` to configure the applier side of the installation:

```
shell> ./tools/tpm configure alpha --hosts=host2 \
    --replication-host=redshift.us-east-1.redshift.amazonaws.com \
    --replication-user=awsRedshiftUser \
    --replication-password=awsRedshiftPass \
    --datasource-type=redshift \
    --batch-enabled=true \
    --batch-load-template=redshift \
    --redshift-database=dev \
    --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 the service `alpha`, because the `--hosts` [347] option has been used, we will only update the configuration for the specified host.

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

Defines the hostname of the Redshift service into which the Redshift data will be loaded. This should be the fully-qualified hostname of the Redshift instance.

- `--replication-user=awsRedshiftUser`

Defines the username with access rights to login to the configured Redshift service.

- `--replication-password=awsRedshiftPass`

Defines the username with access rights to login to the configured Redshift service.

- `--batch-enabled=true` [331]

The Amazon Redshift applier uses the Tungsten Replication batch loading system to generate the load data imported.

- `--batch-load-template=redshift` [331]

The batch load templates configure how the batch load operation operates. These templates perform the necessary steps to load the generated CSV file, and execute the SQL statement that migrate the data from the seed tables.

- `--datasource-type=redshift` [340]

Specifies the datasource type, in this case Amazon Redshift. This ensures that the correct applier is being used to apply transactions in the target database.

- `--redshift-database=default` [362]

Set the database name to be used when applying data to the Amazon Redshift database.

- `--svc-applier-filters=dropstatementdata [366]`

Adds the `dropstatementdata` filter to the applier to ensure DDL is not executed on the Redshift database.

- `--svc-applier-block-commit-interval=10 [366]`

Sets the block commit interval in seconds. For more information, see [Section 12.1, “Block Commit”](#).

- `--svc-applier-block-commit-size=5 [366]`

Sets the block commit interval in transactions. For more information, see [Section 12.1, “Block Commit”](#).

With the configuration in place, the replicators can be installed by running:

```
shell> tpm install alpha
```

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

On the host that is loading data into Redshift, create the `s3-config-servicename.json` file and then copy that file into the `share` directory within the installed directory on that host. For example:

```
shell> cp s3-config-servicename.json /opt/continuent/share/
```

Now the services can be started:

```
shell> tpm start alpha
```

Once the service is configured and running, the service can be monitored as normal using the `trepctl` command. See [Section 6.3.6, “Management and Monitoring of Amazon Redshift Deployments”](#) for more information.

6.3.4. Verifying your Redshift Installation

1. Create a database within your source MySQL instance:

```
mysql> CREATE DATABASE redtest;
```

2. Create a table within your source MySQL instance:

```
mysql> CREATE TABLE redtest.msg (id INT PRIMARY KEY AUTO_INCREMENT,msg CHAR(80));
```

3. Create a schema for the tables:

```
redshift> CREATE SCHEMA redtest;
```

4. Create a staging table within your Redshift instance:

```
redshift> CREATE TABLE redtest.stage_xxx_msg (tungsten_opcode CHAR(1), \
    tungsten_seqno INT, tungsten_row_id INT,tungsten_date CHAR(30),id INT,msg CHAR(80));
```

5. Create the target table:

```
redshift> CREATE TABLE redtest.msg (id INT,msg CHAR(80));
```

6. Insert some data within your MySQL source instance:

```
mysql> INSERT INTO redtest.msg VALUES (0,'First');
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO redtest.msg VALUES (0,'Second');
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO redtest.msg VALUES (0,'Third');
Query OK, 1 row affected (0.04 sec)

mysql> UPDATE redtest.msg SET msg = 'This is the first update of the second row' WHERE ID = 2;
```

7. Check the replicator status on the applier (`host2`):

```
shell> trepctl status
```

There should be 5 transactions replicated.

8. Check the table within Redshift:

```
redshift> SELECT * FROM redtest.msg;
```

```

1 First
3 Third
2 This is the first update of the second row

```

6.3.5. Keeping CDC Information

The Redshift applier can keep the CDC data, that is, the raw CDC CSV data that is recorded and replicated during the loading process, rather than simply cleaning up the CDC files and deleting them. The CDC data can be useful if you want to be able to monitor data changes over time.

The process works as follows:

1. Batch applier generates CSV files.
2. Batch applier loads the CSV data into the staging tables.
3. Batch applier loads the CSV data into the CDC tables.
4. Staging data is merged with the base table data.
5. Staging data is deleted.

Unlike the staging and base table information, the data in the CDC tables is kept forever, without removing any of the processed information. Using this data you can report on change information over time for different data sets, or even recreate datasets at a specific time by using the change information.

To enable this feature:

1. When creating the DDL for the staging and base tables, also create the table information for the CDC data for each table. The actual format of the information is the same as the staging table data, and can be created using `ddlscan`:

```

shell> ddlscan -service my_red -db test \
    -template ddl-mysql-redshift-staging.vm \
    -opt renameSchema cdc_{schema} -opt renameTable {table}_cdc

```

2. In the configuration file, `s3-config-svc.json` for each service, specify the name of the table to be used when storing the CDC information using the `storeCDCIn` field. This should specify the table template to be used, with the schema and table name being automatically replaced by the load script. The structure should match the structure used by `ddlscan` to define the CDC tables:

```

{
  "awsS3Path" : "s3://your-bucket-for-redshift/redshift-test",
  "awsAccessKey" : "access-key-id",
  "awsSecretKey" : "secret-access-key",
  "storeCDCIn" : "cdc_{schema}.{table}_cdc"
}

```

3. Restart the replicator using `replicator restart` to update the configuration.

6.3.6. Management and Monitoring of Amazon Redshift Deployments

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

```

shell> trepctl status
Processing status command...
NAME          VALUE
-----
appliedLastEventId : mysql-bin.000006:0000000000002857;-1
appliedLastSeqno : 15
appliedLatency  : 1.918
autoRecoveryEnabled : false
autoRecoveryTotal : 0
channels       : 1
clusterName    : alpha
currentEventId : mysql-bin.000006:0000000000002857
currentTimeMillis : 1407336195165
dataServerHost : redshift1
extensions     :
host           : redshift1
latestEpochNumber : 8
masterConnectUri : thl://localhost:/
masterListenUri  : thl://redshift1:2112/
maximumStoredSeqNo : 15
minimumStoredSeqNo : 0
offlineRequests : NONE
pendingError    : NONE
pendingErrorCode : NONE

```

```

pendingErrorEventId   : NONE
pendingErrorSeqno    : -1
pendingExceptionMessage: NONE
pipelineSource        : jdbc:mysql:thin://redshift1:3306/tungsten_alpha
relativeLatency       : 35.164
resourcePrecedence   : 99
rmiPort               : 10000
role                  : master
seqnoType             : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName     : alpha
siteName               : default
sourceId              : redshift1
state                 : ONLINE
timeInStateSeconds    : 34.807
transitioningTo       :
uptimeSeconds          : 36.493
useSSLConnection      : false
version               : Tungsten Replicator 5.0.1 build 136
Finished status command...

```

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

```

shell> trepctl status
Processing status command...
NAME           VALUE
----           -----
appliedLastEventId : mysql-bin.000006:0000000000002857;-1
appliedLastSeqno : 15
appliedLatency  : 154.748
autoRecoveryEnabled : false
autoRecoveryTotal : 0
channels        : 1
clusterName     : alpha
currentEventId  : NONE
currentTimeMillis: 1407336316454
dataServerHost   : redshift.us-east-1.redshift.amazonaws.com
extensions      :
host            : redshift.us-east-1.redshift.amazonaws.com
latestEpochNumber: 8
masterConnectUri: thl://redshift1:2112/
masterListenUri : null
maximumStoredSeqNo: 15
minimumStoredSeqNo: 0
offlineRequests  : NONE
pendingError     : NONE
pendingErrorCode : NONE
pendingErrorEventId: NONE
pendingErrorSeqno: -1
pendingExceptionMessage: NONE
pipelineSource   : thl://redshift1:2112/
relativeLatency : 156.454
resourcePrecedence: 99
rmiPort          : 10000
role             : slave
seqnoType        : java.lang.Long
serviceName       : alpha
serviceType       : local
simpleServiceName: alpha
siteName          : default
sourceId         : redshift.us-east-1.redshift.amazonaws.com
state            : ONLINE
timeInStateSeconds: 2.28
transitioningTo  :
uptimeSeconds    : 524104.751
useSSLConnection : false
version          : Tungsten Replicator 5.0.1 build 136
Finished status command...

```

The `appliedLastSeqno` should match as normal. Because of the batching of transactions the `appliedLatency` may be much higher than a normal MySQL to MySQL replication.

The batch loading parameters controlling the batching of data can be tuned and update by studying the output from the `trepsvc.log` log file. The log will show a line containing the number of rows updated:

```
INFO  scripting.JavascriptExecutor COUNT: 4
```

See [Section 12.1, “Block Commit”](#) for more information on these parameters.

6.3.7. Troubleshooting Amazon Redshift Installations

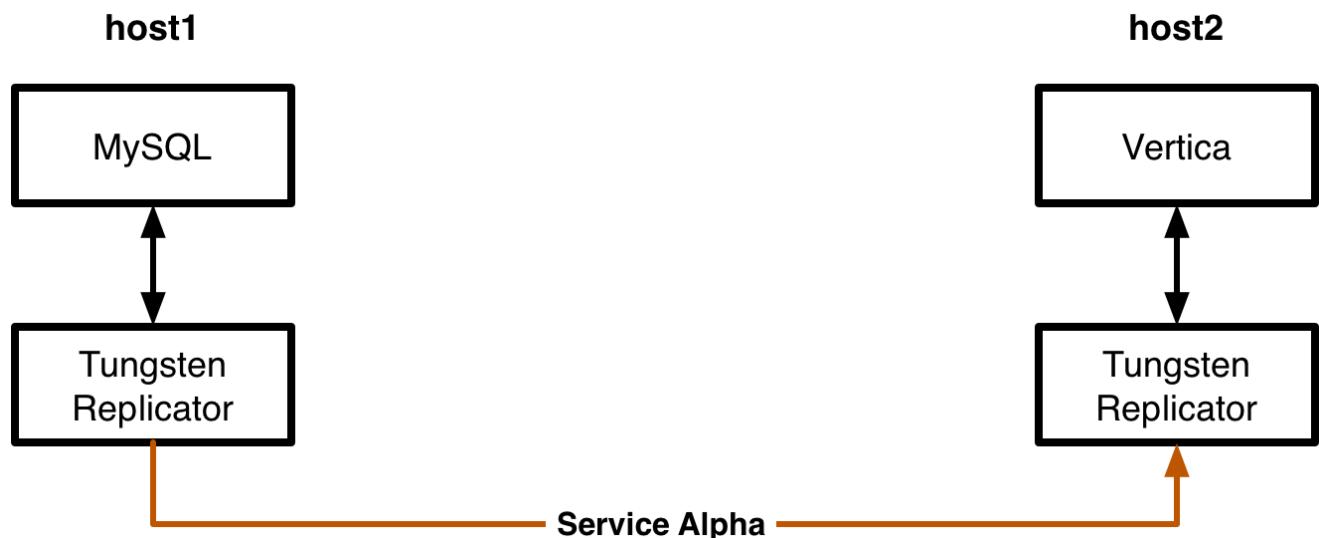
6.4. Deploying MySQL to Vertica Replication

Hewlett-Packard's Vertica provides support for BigData, SQL-based analysis and processing. Integration with MySQL enables data to be replicated live from the MySQL database directly into Vertica without the need to manually export and import the data.

Replication to Vertica operates as follows:

- Data is extracted from the source database into THL.
- When extracting the data from the THL, the Vertica replicator writes the data into CSV files according to the name of the source tables. The files contain all of the row-based data, including the global transaction ID generated by Tungsten Replication during replication, and the operation type (insert, delete, etc) as part of the CSV data.
- The CSV data is then loaded into Vertica into staging tables.
- SQL statements are then executed to perform updates on the live version of the tables, using the CSV, batch loaded, information, deleting old rows, and inserting the new data when performing updates to work effectively within the confines of Vertica operation.

Figure 6.6. Topologies: MySQL to Vertica



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

6.4.1. Preparing Hosts for Vertica Deployments

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

MySQL Host

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

The following are *required* for replication to Vertica:

- MySQL must be using Row-based replication for information to be replicated to Vertica. For the best results, you should change the global binary log format, ideally in the configuration file (`my.cnf`):

```
binlog-format = row
```

Alternatively, the global binlog format can be changed by executing the following statement:

```
mysql> SET GLOBAL binlog-format = ROW;
```

For MySQL 5.6.2 and later, you must enable full row log images:

```
binlog-row-image = full
```

This information will be forgotten when the MySQL server is restarted; placing the configuration in the `my.cnf` file will ensure this option is permanently enabled.

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

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

Tables must also be configured as UTF8 tables, and existing tables should be updated to UTF8 support before they are replicated to prevent character set corruption issues.

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

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

Vertica Host

On the Vertica host, you need to perform some preparation of the destination database, first creating the database, and then creating the tables that are to be replicated.

- Create a database (if you want to use a different one than those already configured), and a schema that will contain the Tungsten data about the current replication position:

```
shell> vsql -Udbadmin -wsecret bigdata
Welcome to vsql, the Vertica Analytic Database v5.1.1-0 interactive terminal.

Type: \h for help with SQL commands
      \? for help with vsql commands
      \g or terminate with semicolon to execute query
      \q to quit

bigdata=> create schema tungsten alpha;
```

The schema will be used only by Tungsten Replication to store metadata about the replication process.

- Locate the Vertica JDBC driver. This can be downloaded separately from the Vertica website. The driver will need to be copied into the Tungsten Replication `lib` directory.

```
shell> cp vertica-jdbc-7.1.2-0.jar tungsten-replicator-5.0.1-136/tungsten-replicator/lib/
```

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

To use `ddlscan`, the template for Vertica must be specified, along with the user/password information to connect to the source database to collect the schema definitions. The tool should be run from the templates directory.

The tool will need to be executed twice, the first time generates the live table definitions:

```
shell> cd tungsten-replicator-5.0.1-136
shell> cd tungsten-replicator/samples/extensions/velocity/
shell> ddlscan -user tungsten -url 'jdbc:mysql:thin://host1:13306/access_log' -pass password \
      -template ddl-mysql-vertica.vm -db access_log
/*
SQL generated on Fri Sep 06 14:37:40 BST 2013 by ./ddlscan utility of Tungsten

url = jdbc:mysql:thin://host1:13306/access_log
user = tungsten
dbName = access_log
*/
CREATE SCHEMA access_log;

DROP TABLE access_log.access_log;

CREATE TABLE access_log.access_log
(
    id INT ,
    userid INT ,
    datetime INT ,
    session CHAR(30) ,
    operation CHAR(80) ,
    opdata CHAR(80) ) ORDER BY id;
...
```

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

```
shell> ddlscan -user tungsten -url 'jdbc:mysql:thin://host1:13306/access_log' -pass password \
```

```
-template ddl-mysql-vertica.vm -db access_log >access_log.ddl
```

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

The file can then be applied to Vertica:

```
shell> cat access_log.ddl | vsql -Udbadmin -wsecret bigdata
```

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

```
shell> ddlscan -user tungsten -url 'jdbc:mysql:thin://host1:13306/access_log' -pass password \
    -template ddl-mysql-vertica-staging.vm -db access_log >access_log.ddl-staging
```

Then applied to Vertica:

```
shell> cat access_log.ddl-staging | vsql -Udbadmin -wsecret bigdata
```

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

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

6.4.2. Installing Vertica Replication

Configuration of the replication deployment to Vertica can be made using a single **tpm** staging-based deployment. However, because the configuration must be different for the master and slave hosts, the configuration must be performed in multiple steps.

1. Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-5.0.tar.gz
```

2. Change into the staging directory:

```
shell> cd tungsten-replicator-5.0
```

3. Locate the Vertica JDBC driver. This can be downloaded separately from the Vertica website. The driver will need to be copied into the Tungsten Replication **lib** directory.

```
shell> cp vertica-jdbc-7.1.2-0.jar tungsten-replicator-5.0.1-136/tungsten-replicator/lib/
```

4. Configure the main parameters for the replicator service:

```
shell> ./tools/tpm configure alpha \
    --master=host1 \
    --members=host1,host3 \
    --install-directory=/opt/continuent \
    --disable-relay-logs=true \
    --skip-validation-check=HostsFileCheck \
    --enable-heterogeneous-service=true \
    --start
```

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**

Executes **tpm** in **configure** mode to create the service **alpha**. This sets up the configuration information without performing an installation, enabling further configuration to be applied.

- **--master=host1 [351]**

Specifies which host will be the master.

- **--members=host1 [351]**

Specifies the members; **host1** is the master, **host3** is the Vertica host.

- **--install-directory=/opt/continuent [347]**

Directory where Tungsten Replication will be installed.

- **--disable-relay-logs=true [342]**

Disable the relay logs.

- `--skip-validation-check=HostsFileCheck`
Disable checking the hosts during deployment
- `--enable-heterogeneous-service=true [344]`
Set certain parameters that ensure that a heterogeneous deployment operates correctly, including setting files, Java file types, and other settings.
- `--start [365]`
This starts the replicator service once the replicator has been configured and installed.

5. Configure the parameters for the master host which will extract the information from MySQL:

```
shell> ./tools/tpm configure alpha \
  --hosts=host1 \
  --replication-user=tungsten \
  --replication-password=password \
  --enable-heterogeneous-master=true
```

This operation sets the user and password information for accessing the MySQL server; this is required by some of the filters which extract information from the running service.

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.

- `--hosts=host1 [347]`
By explicitly stating the list of hosts, only the configuration for the hosts listed will be updated. In this case, the configuration for the master, `host1` is being updated.
- `--replication-user=tungsten [363]`
The user name that will be used to apply replication changes to the database on slaves.
- `--replication-password=password [363]`
The password that will be used to apply replication changes to the database on slaves.
- `--enable-heterogeneous-master=true [344]`
Ensures the correct file encoding and filters are applied to the master configuration for a heterogeneous deployment.

6. Configure the parameters for the slave host that will apply the events to Vertica:

```
shell> ./tools/tpm configure alpha \
  --hosts=host3 \
  --replication-user=dbadmin \
  --replication-password=password \
  --batch-load-template=vertica \
  --batch-load-language=js \
  --datasource-type=vertica \
  --vertica-database=bigdata \
  --replication-host=host3 \
  --replication-port=5433 \
  --skip-validation-check=InstallerMasterSlaveCheck \
  --svc-applier-block-commit-size=25000 \
  --svc-applier-block-commit-interval=30s
```

This configures the Vertica slave to accept replication data from the master.

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

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

- `tpm install`
Executes `tpm` in `install` mode to create the service `alpha`.
- `--hosts=host3 [347]`
By explicitly stating the list of hosts, only the configuration for the hosts listed will be updated. In this case, the configuration for the master, `host3` is being updated.

- `--replication-user=dbadmin [363]`

Set the user for connecting to the Vertica database service.

- `--replication-password=password [363]`

Set the password used to connect to the Vertica database service.

- `--batch-enabled=true [331]`

The Vertica applier uses the Tungsten Replication batch loading system to generate the load data imported.

- `--batch-load-language=js [331]`

The batch load language should be JavaScript, which supports the required scripts and loading processes.

- `--batch-load-template=vertica [331]`

The batch load templates configure how the batch load operation operates. These templates perform the necessary steps to load the generated CSV file, and execute the SQL statement that migrate the data from the seed tables.

- `--datasource-type=vertica [340]`

Specifies the datasource type, in this case Vertica. This ensures that the correct applier is being used to apply transactions in the target database.

- `--vertica-dbname=bigdata [370]`

Set the database name to be used when applying data to the Vertica database.

- `--replication-port=5433 [363]`

Set the port number to use when connecting to the Vertica database service.

- `--replication-host=host3 [363]`

Set the replication host.

- `--skip-validation-check=InstallerMasterSlaveCheck`

Skip the test for the master/slave check; this does not apply in a heterogeneous deployment.

- `--svc-applier-block-commit-size=25000 [366]`

Set the block commit size to 25,000 events. Because Vertica uses the batch applier, the commit process can submit a larger number of transactions simultaneously to Vertica.

For more information, see [Section 12.1, “Block Commit”](#).

- `--svc-applier-block-commit-interval=30s [366]`

Set the maximum interval between commits, regardless of the transaction count, to 3s.

For more information, see [Section 12.1, “Block Commit”](#).

7. Install the services:

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

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

Once the service is configured and running, the service can be monitored as normal using the `treptctl` command. See [Section 6.4.3, “Management and Monitoring of Vertica Deployments”](#) for more information.

6.4.3. Management and Monitoring of Vertica Deployments

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

```
shell> treptctl status
Processing status command...
NAME          VALUE
```

```
-----
appliedLastEventId      : mysql-bin.000012:0000000128889042;0
appliedLastSeqno        : 1070
appliedLatency          : 22.537
channels                : 1
clusterName             : alpha
currentEventId          : mysql-bin.000012:0000000128889042
currentTimeMillis       : 1378489888477
dataServerHost           : host1
extensions              :
latestEpochNumber       : 897
masterConnectUri         : th1://localhost:/
masterListenUri          : th1://host1:2112/
maximumStoredSeqNo      : 1070
minimumStoredSeqNo      : 0
offlineRequests          : NONE
pendingError             : NONE
pendingErrorCode         : NONE
pendingErrorEventId     : NONE
pendingErrorSeqno        : -1
pendingExceptionMessage  : NONE
pipelineSource           : jdbc:mysql:thin://host1:13306/
relativeLatency          : 691980.477
resourcePrecedence       : 99
rmiPort                 : 10000
role                    : master
seqnoType               : java.lang.Long
serviceName              : alpha
serviceType              : local
simpleServiceName        : alpha
siteName                : default
sourceId                : host1
state                   : ONLINE
timeInStateSeconds      : 694039.058
transitioningTo          :
uptimeSeconds            : 694041.81
useSSLConnection         : false
version                 : Tungsten Replicator 5.0.1 build 136
Finished status command...
```

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

```
shell> trepctl status
Processing status command...
NAME          VALUE
-----
appliedLastEventId      : mysql-bin.000012:0000000128889042;0
appliedLastSeqno        : 1070
appliedLatency          : 78.302
channels                : 1
clusterName             : default
currentEventId          : NONE
currentTimeMillis       : 1378479271609
dataServerHost           : host3
extensions              :
latestEpochNumber       : 897
masterConnectUri         : th1://host1:2112/
masterListenUri          : null
maximumStoredSeqNo      : 1070
minimumStoredSeqNo      : 0
offlineRequests          : NONE
pendingError             : NONE
pendingErrorCode         : NONE
pendingErrorEventId     : NONE
pendingErrorSeqno        : -1
pendingExceptionMessage  : NONE
pipelineSource           : th1://host1:2112/
relativeLatency          : 681363.609
resourcePrecedence       : 99
rmiPort                 : 10000
role                    : slave
seqnoType               : java.lang.Long
serviceName              : alpha
serviceType              : local
simpleServiceName        : alpha
siteName                : default
sourceId                : host3
state                   : ONLINE
timeInStateSeconds      : 681486.806
transitioningTo          :
uptimeSeconds            : 689922.693
useSSLConnection         : false
version                 : Tungsten Replicator 5.0.1 build 136
Finished status command...
```

The `appliedLastSeqno` should match as normal. Because of the batching of transactions the `appliedLatency` may be much higher than a normal MySQL to MySQL replication.

6.4.4. Troubleshooting Vertica Installations

- Remember that changes to the DDL within the source database are not automatically replicated to Vertica. Changes to the table definitions, additional tables, or additional databases, must all be updated manually within Vertica.
- If you get errors similar to:

```
stage_xxx_access_log does not exist
```

When loading into Vertica, it means that the staging tables have not created correctly. Check the steps for creating the staging tables using `ddlscan` in [Section 6.4.1, "Preparing Hosts for Vertica Deployments"](#).

- Replication may fail if date types contain zero values, which are legal in MySQL. For example, the timestamp `0000-00-00 00:00:00` is valid in MySQL. An error reporting a mismatch in the values will be reported when applying the data into Vertica, for example:

```
ERROR 2631: Column "time" is of type timestamp but expression is of type int
HINT: You will need to rewrite or cast the expression
```

Or:

```
ERROR 2992: Date/time field value out of range: "0"
HINT: Perhaps you need a different "datestyle" setting
```

To address this error, use the `zerodate2null` filter, which translates zero-value dates into a valid NULL value. This can be enabled by adding the `zerodate2null` filter to the applier stage when configuring the service using `tpm`:

```
shell> ./tools/tpm update alpha --repl-svc-applier-filters=zerodate2null
```

Chapter 7. Advanced Deployments

7.1. 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](#).

7.1.1. 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.

7.1.2. Enabling Parallel Apply

Parallel apply is enabled using the `--svc-parallelization-type` [366] and `--channels` [331] options of `tpm`. The parallelization type defaults to `none` which is to say that parallel apply is disabled. You should set it to `disk` [182]. The `--channels` [331] option sets the the number of channels (i.e., threads) you propose to use for applying data. Here is a code example of master-slave installation with parallel apply enabled. The slave will apply transactions using 30 channels.

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

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` [331] — 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` [355] — Used to allow Tungsten to take over from native MySQL replication and parallelize it. See here for more.

7.1.3. 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` [331] 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.

7.1.4. Disk vs. Memory Parallel Queues

Channels receive transactions through a special type of queue, known as a parallel queue. Tungsten offers two implementations of parallel queues, which vary in their performance as well as the requirements they may place on hosts that operate parallel apply. You choose the type of queue to enable using the `--svc-parallelization-type` [366] 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:

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

You may need to increase the Java VM heap size when you increase the parallel queue maximum size. Use the `--java-mem-size [349]` 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.

7.1.5. Parallel Replication and Offline Operation

7.1.5.1. Clean Offline Operation

When you issue a `trepctl offline` command, Tungsten Replicator will bring all channels to the same point in the log and then go offline. This is known as going offline cleanly. When a slave has been taken offline cleanly the following are true:

- The `trep_commit_seqno` table contains a single row
- The `trep_shard_channel` table is empty

When parallel replication is not enabled, you can take the replicator offline by stopping the replicator process. There is no need to issue a `trepctl offline` command first.

7.1.5.2. Tuning the Time to Go Offline Cleanly

Putting a replicator offline may take a while if the slowest and fastest channels are far apart, i.e., if one channel gets far ahead of another. The separation between channels is controlled by the `maxOfflineInterval` parameter, which defaults to 5 seconds. This sets the allowable distance between commit timestamps processed on different channels. You can adjust this value at installation or later. The following example shows how to change it after installation. This can be done at any time and does not require the replicator to go offline cleanly.

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

The offline interval is only the approximate time that Tungsten Replicator will take to go offline. Up to a point, larger values (say 60 or 120 seconds) allow the replicator to parallelize in spite of a few operations that are relatively slow. However, the down side is that going offline cleanly can become quite slow.

7.1.5.3. Unclean Offline

If you need to take a replicator offline quickly, you can either stop the replicator process or issue the following command:

```
shell> trepctl offline -immediate
```

Both of these result in an unclean shutdown. However, parallel replication is completely crash-safe provided you use transactional table types like InnoDB, so you will be able to restart without causing slave consistency problems.

Warning

You must take the replicator offline cleanly to change the number of channels or when reverting to MySQL native replication. Failing to do so can result in errors when you restart replication.

7.1.6. Adjusting Parallel Replication After Installation

7.1.6.1. How to Change Channels Safely

To change the number of channels you must take the replicator offline cleanly using the following command:

```
shell> trepctl offline
```

This command brings all channels up the same transaction in the log, then goes offline. If you look in the `trep_commit_seqno` table, you will notice only a single row, which shows that updates to the slave have been completely serialized to a single point. At this point you may safely reconfigure the number of channels on the replicator, for example using the following command:

```
shell> tpm update alpha --channels=5
shell> replicator restart
```

You can check the number of active channels on a slave by looking at the "channels" property once the replicator restarts.

If you attempt to reconfigure channels without going offline cleanly, Tungsten Replicator will signal an error when you attempt to go online with the new channel configuration. The cure is to revert to the previous number of channels, go online, and then go offline cleanly. Note that attempting to clean up the `trep_commit_seqno` and `trep_shard_channel` tables manually can result in your slaves becoming inconsistent and requiring full resynchronization. You should only do such cleanup under direction from Continuent support.

Warning

Failing to follow the channel reconfiguration procedure carefully may result in your slaves becoming inconsistent or failing. The cure is usually full resynchronization, so it is best to avoid this if possible.

7.1.6.2. How to Switch Parallel Queue Types Safely

As with channels you should only change the parallel queue type after the replicator has gone offline cleanly. The following example shows how to update the parallel queue type after installation:

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

7.1.7. Monitoring Parallel Replication

Basic monitoring of a parallel deployment can be performed using the techniques in [Chapter 8, Operations Guide](#). Specific operations for parallel replication are provided in the following sections.

7.1.7.1. Useful Commands for Parallel Monitoring Replication

The replicator has several helpful commands for tracking replication performance:

Command	Description
<code>trepctl status</code>	Shows basic variables including overall latency of slave and number of apply channels
<code>trepctl status -name shards</code>	Shows the number of transactions for each shard
<code>trepctl status -name stores</code>	Shows the configuration and internal counters for stores between tasks
<code>trepctl status -name tasks</code>	Shows the number of transactions (events) and latency for each independent task in the replicator pipeline

7.1.7.2. Parallel Replication and Applied Latency On Slaves

The `trepctl status` appliedLastSeqno parameter shows the sequence number of the last transaction committed. Here is an example from a slave with 5 channels enabled.

```
shell> trepctl status
Processing status command...
NAME          VALUE
-----
appliedLastEventId : mysql-bin.000211:000000020094456:0
appliedLastSeqno  : 78021
appliedLatency   : 0.216
channels        : 5
...
Finished status command...
```

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

- Busy channels mark their position in table `trep_commit_seqno` as they commit. These are up-to-date with the traffic on that channel, but channels have latency between those that have a lot of big transactions and those that are more lightly loaded.
- Inactive channels do not get any transactions, hence do not mark their position. Tungsten sends a control event across all channels so that they mark their commit position in `trep_commit_channel`. It is possible to see a delay of many seconds or even minutes in unloaded systems from the true state of the slave because of idle channels not marking their position yet.

For systems with few transactions it is useful to lower the synchronization interval to a smaller number of transactions, for example 500. The following command shows how to adjust the synchronization interval after installation:

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

Note that there is a trade-off between the synchronization interval value and writes on the DBMS server. With the foregoing setting, all channels will write to the `trep_commit_seqno` table every 500 transactions. If there were 50 channels configured, this could lead to an increase in writes of up to 10%—each channel could end up adding an extra write to mark its position every 10 transactions. In busy systems it is therefore better to use a higher synchronization interval for this reason.

You can check the current synchronization interval by running the `trepctl status -name stores` command, as shown in the following example:

```
shell> 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.

7.1.7.3. Relative Latency

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

```
shell> trepctl status
Processing status command...
NAME          VALUE
-----
...
appliedLastEventId   : mysql-bin.000211:000000020094766;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.

7.1.7.4. Serialization Count

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

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

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

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

7.1.7.5. Maximum Offline Interval

The maximum offline interval (`maxOfflineInterval`) parameter controls the "distance" between the fastest and slowest channels when parallel apply is enabled. The replicator measures distance using the seconds between commit times of the last transaction processed on each channel. This time is roughly equivalent to the amount of time a replicator will require to go offline cleanly.

You can change the `maxOfflineInterval` as shown in the following example, the value is defined in seconds.

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

You can view the configured value as well as the estimate current value using the `trepctl status -name stores` command, as shown in yet another example:

```
shell> treptl status -name stores
Processing status command (stores)...
NAME          VALUE
----          -----
...
estimatedOfflineInterval: 1.3
...
maxOfflineInterval      : 15
...
Finished status command (stores)...
```

7.1.7.6. Workload Distribution

Parallel apply works best when transactions are distributed evenly across shards and those shards are distributed evenly across available channels. You can monitor the distribution of transactions over shards using the `treptl status -name shards` command. This command lists transaction counts for all shards, as shown in the following example.

```
shell> treptl 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> treptl status -name shards
Processing status command (shards)...
NAME          VALUE
----          -----
...
appliedLastEventId: mysql-bin.000211:0000000020095529:0
appliedLastSeqno : 78026
appliedLatency   : 0.558
eventCount      : 26
shardId        : #UNKNOWN
stage          : q-to-dbms
...
Finished status command (shards)...
shell> tnl list -seqno 78026
SEQ# = 78026 / FRAG# = 0 (last frag)
- TIME = 2013-01-17 22:29:42.0
- EPOCH# = 1
- EVENTID = mysql-bin.000211:0000000020095529:0
- SOURCEID = logos1
- METADATA = [mysql_server_id=1;service=percona;shard=#UNKNOWN]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [##charset = ISO8859_1, autocommit = 1, sql_auto_is_null = 0, »
    foreign_key_checks = 1, unique_checks = 1, sql_mode = '', character_set_client = 8, »
    collation_connection = 8, collation_server = 33]
- SCHEMA =
- SQL(0) = insert into mats_0.foo values(1) /* ____SERVICE____ = [percona] */
- OPTIONS = [##charset = ISO8859_1, autocommit = 1, sql_auto_is_null = 0, »
    foreign_key_checks = 1, unique_checks = 1, sql_mode = '', character_set_client = 8, »
    collation_connection = 8, collation_server = 33]
- SQL(1) = insert into mats_1.foo values(1)
```

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

```
shell> treptl status -name tasks
Processing status command (tasks)...
...
NAME          VALUE
----          -----
```

```

appliedLastEventId: mysql-bin.000211:0000000020095076;0
appliedLastSeqno : 78023
appliedLatency   : 0.248
applyTime        : 0.003
averageBlockSize: 2.520
cancelled       : false
currentLastEventId: mysql-bin.000211:0000000020095076;0
currentLastFragno: 0
currentLastSeqno : 78023
eventCount      : 5302
extractTime     : 274.907
filterTime      : 0.0
otherTime       : 0.0
stage           : q-to-dbms
state           : extract
taskId          : 0
...
Finished status command (tasks)...

```

If you see one or more channels that have a very high `eventCount`, consider either assigning shards explicitly to channels or redistributing the workload in your application to get better performance.

7.1.8. Controlling Assignment of Shards to Channels

Tungsten Replicator by default assigns channels using a round robin algorithm that assigns each new shard to the next available channel. The current shard assignments are tracked in table `trep_shard_channel` in the Tungsten catalog schema for the replication service.

For example, if you have 2 channels enabled and Tungsten processes three different shards, you might end up with a shard assignment like the following:

```

foo => channel 0
bar => channel 1
foobar => channel 0

```

This algorithm generally gives the best results for most installations and is crash-safe, since the contents of the `trep_shard_channel` table persist if either the DBMS or the replicator fails.

It is possible to override the default assignment by updating the `shard.list` file found in the `tungsten-replicator/conf` directory. This file normally looks like the following:

```

# SHARD MAP FILE.
# This file contains shard handling rules used in the ShardListPartitioner
# class for parallel replication. If unchanged shards will be hashed across
# available partitions.

# You can assign shards explicitly using a shard name match, where the form
# is <db>=<partition>.
#common1=0
#common2=
#db1=1
#db2=2
#db3=3

# Default partition for shards that do not match explicit name.
# Permissible values are either a partition number or -1, in which
# case values are hashed across available partitions. (-1 is the
# default.
#(*)=-1

# Comma-separated list of shards that require critical section to run.
# A "critical section" means that these events are single-threaded to
# ensure that all dependencies are met.
#(critical)=common1,common2

# Method for channel hash assignments. Allowed values are round-robin and
# string-hash.
#(hash-method)=round-robin

```

You can update the `shard.list` file to do three types of custom overrides.

1. Change the hashing method for channel assignments. Round-robin uses the `trep_shard_channel` table. The string-hash method just hashes the shard name.
2. Assign shards to explicit channels. Add lines of the form `shard=channel` to the file as shown by the commented-out entries.
3. Define critical shards. These are shards that must be processed in serial fashion. For example if you have a sharded application that has a single global shard with reference information, you can declare the global shard to be critical. This helps avoid applications seeing out of order information.

Changes to shard.list must be made with care. The same cautions apply here as for changing the number of channels or the parallelization type. For subscription customers we strongly recommend conferring with Continuent Support before making changes.

7.2. Batch Loading for Data Warehouses

Tungsten Replicator normally applies SQL changes to slaves by constructing SQL statements and executing in the exact order that transactions appear in the Tungsten History Log (THL). This works well for OLTP databases like MySQL, Oracle, and MongoDB. However, it is a poor approach for data warehouses.

Data warehouse products like Vertica or GreenPlum load very slowly through JDBC interfaces (50 times slower or even more compared to MySQL). Instead, such databases supply batch loading commands that upload data in parallel. For instance Vertica uses the `COPY` command.

Tungsten Replicator has a batch applier named SimpleBatchApplier that groups transactions and then loads data. This is known as "batch apply." You can configure Tungsten to load 10s of thousands of transactions at once using template that apply the correct commands for your chosen data warehouse.

While we use the term *batch apply* Tungsten is not batch-oriented in the sense of traditional Extract/Transfer/Load tools, which may run only a small number of batches a day. Tungsten builds batches automatically as transactions arrive in the log. The mechanism is designed to be self-adjusting. If small transaction batches cause loading to be slower, Tungsten will automatically tend to adjust the batch size upwards until it no longer lags during loading.

7.2.1. How It Works

The batch applier loads data into the slave DBMS using CSV files and appropriate load commands like `LOAD DATA INFILE` or `COPY`. Here is the basic algorithm.

While executing within a commit block, we write incoming transactions into open CSV files written by the class `CsvWriter`. There is one CSV file per database table. The following sample shows typical contents.

```
"I","84900","1","2016-03-11 20:51:10.000","986","http://www.continent.com/software"
"D","84901","2","2016-03-11 20:51:10.000","143",null
"I","84901","3","2016-03-11 20:51:10.000","143","http://www.microsoft.com"
```

Tungsten adds three extra column values to each line of CSV output.

Column	Description
<code>opcode</code>	A transaction code that has the value "I" for insert and "D" for delete. Other types are available.
<code>seqno</code>	The Tungsten transaction sequence number
<code>row_id</code>	A line number that starts with 1 and increments by 1 for each new row
<code>timestamp</code>	The commit timestamp, i.e. the origin timestamp of the committed statement that generated the row information.

Different update types are handled as follows:

- Each insert generates a single row containing all values in the row with an "I" opcode.
- Each delete generates a single row with the key and a "D" opcode. Non-key fields are null.
- Each update results in a delete with the row key followed by an insert.
- Statements are ignored. If you want DDL you need to put it in yourself.

Tungsten writes each row update into the corresponding CSV file for the SQL. At commit time the following steps occur:

1. Flush and close each CSV file. This ensures that if there is a failure the files are fully visible in storage.
2. For each table execute a merge script to move the data from CSV into the data warehouse. This script varies depending on the data warehouse type or even for specific application. It generally consists of a sequence of operating system commands, load commands like `COPY` or `LOAD DATA INFILE` to load in the CSV data, and ordinary SQL commands to move/massage data.
3. When all tables are loaded, issue a single commit on the SQL connection.

The main requirement of merge scripts is that they must ensure rows load and that delete and insert operations apply in the correct order. Tungsten includes load scripts for MySQL and Vertica that do this automatically.

It is common to use staging tables to help load data. These are described in more detail in a later section.

7.2.2. Important Limitations

Tungsten currently has some important limitations for batch loading, namely:

1. Primary keys must be a single column only. Tungsten does not handle multi-column keys.
 2. Binary data is not certified and may cause problems when converted to CSV as it will be converted to Unicode.
- These limitations will be relaxed in future releases.

7.2.3. Batch Applier Setup

Here is how to set up on MySQL. For more information on specific data warehouse types, refer to [Chapter 2, Deployment](#).

1. Enable row replication on the MySQL master using `set global binlog_format=row` or by updating `my.cnf`.
2. Ensure that you are operating using GMT throughout your source and target database.
3. Install using the `--batch-enabled=true` [331] option. Here's a typical installation command using `tpm`:

```
shell> ./tools/tpm batch
--cluster-hosts=logos1,logos2 \
--master-host=logos1 \
--datasource-user=tungsten \
--datasource-password=secret \
--batch-enabled=true \
--batch-load-template=mysql \
--svc-extractor-filters=colnames,pkey \
--property=replicator.filter.pkey.addPkeyToInserts=true \
--property=replicator.filter.pkey.addColumnstoDeletes=true \
--install-directory=/opt/continuent \
--channels=1 \
--buffer-size=1000 \
--mysql-use-bytes-for-string=false \
--skip-validation-check=MySQLConfigFileCheck \
--skip-validation-check=MySQLExtractorServerIDCheck \
--skip-validation-check=MySQLApplierServerIDCheck \
--svc-parallelization-type=disk
--start-and-report
```

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.

There are a number of important options for batch loading.

- `--batch-enabled=true` [331]

Enables batch loading on the slave.

- `--batch-load-template=name` [331]

Selects a set of connect and merge files. (See below.)

- `--svc-table-engine=name` [367]

For MySQL-based data warehouses, sets the table type for Tungsten catalogs. Must be an engine type valid for the target database.

- `--svc-extractor-filters=colnames,pkey` [366]

Filters that must run on master to fill in column names and the table primary key from the original data.

- `--property=replicator.filter.pkey.addPkeyToInserts=true`

Adds primary key data to inserts.

- `--property=replicator.filter.pkey.addColumnstoDeletes=true`

Adds table columns to deletes.

You may force additional parameter settings using `--property` flags if necessary.

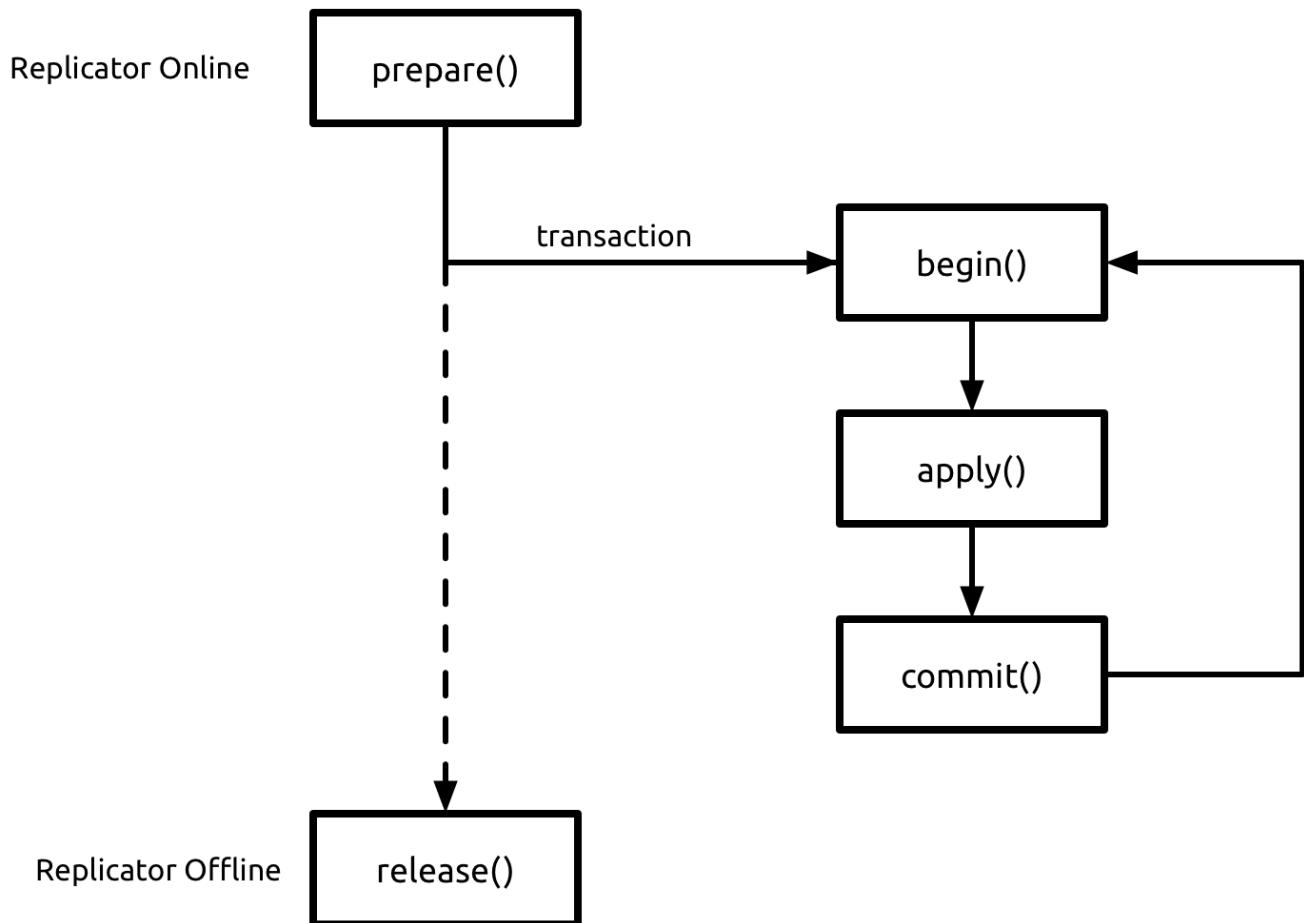
7.2.4. JavaScript Batchloader Scripts

The JavaScript batchloader enables data to be loaded into datawarehouse and other targets through a simplified JavaScript command script. The script implements specific functions for specification stages for the apply process, from preparation to commit, allowing for internal data, external commands, and other operations to be executed in sequence.

The JavaScript batchloader was included in Tungsten Replicator 3.0 and replaced the older script-based batchloading mechanism.

The actual loading process works through the specification of a JavaScript batchload script that defines what operations to perform during each stage of the batchloading process. These mirror the basic steps in the operation of applying the data that is being batchloaded, as shown in [Figure 7.1, "Batchloading: JavaScript"](#).

Figure 7.1. Batchloading: JavaScript



To summarize:

- `prepare()` is called when the replicator goes online
- `begin()` is called before a single transaction starts
- `apply()` is called to copy and load the raw CSV data
- `commit()` is called after the raw data has been loaded
- `release()` is called when the replicator goes offline

7.2.4.1. JavaScript Batchloader Scripts `apply()` Function

7.2.4.2. JavaScript Batchloader Scripts `begin()` Function

7.2.4.3. JavaScript Batchloader Scripts `commit()` Function

7.2.4.4. JavaScript Batchloader Scripts `prepare()` Function

7.2.4.5. JavaScript Batchloader Scripts `release()` Function

7.2.4.6. JavaScript Batchloader with Parallel Apply

The JavaScript batchloader can be used with parallel apply to enable multiple threads to be generated and apply data to the target database. This can be useful in datawarehouse environments where simultaneous loading (and commit) enables effective application of multiple table data into the datawarehouse.

- The defined JavaScript methods like prepare, begin, commit, and release are called independently for each environment. This means that you should ensure actions in these methods do not conflict with each other.
- CSV files are divided across the scripts. If there is a large number of files that all take about the same time to load and there are three threads (parallelization=3), each individual load script will see about a third of the files. You should therefore not code assumptions that you have seen all tables or CSV files in a single script.
- Parallel load script is only recommended for data sources like Hadoop that are idempotent. When applying to a data source that is non-idempotent (for example MySQL or potentially Vertica) you should just use a single thread.

7.2.4.7. Batchloading Javascript Scripts

7.2.5. Staging Tables

Staging tables are intermediate tables that help with data loading. There are different usage patterns for staging tables.

7.2.5.1. Staging Table Names

Tungsten assumes that staging tables, if present, follow certain conventions for naming and provides a number of configuration properties for generating staging table names that match the base tables in the data warehouse without colliding with them.

Property	Description
<code>stageColumnPrefix</code>	Prefix for seqno, row_id, and opcode columns generated by Tungsten
<code>stageTablePrefix</code>	Prefix for stage table name
<code>stageSchemaPrefix</code>	Prefix for the schema in which the stage tables reside

These values are set in the static properties file that defines the replication service. They can be set at install time using `--property` options. The following example shows typical values from a service properties file.

```
replicator.applier.dbms.stageColumnPrefix=tungsten_
replicator.applier.dbms.stageTablePrefix=stage_xxx_
replicator.applier.dbms.stageSchemaPrefix=load_
```

If your data warehouse contains a table named `foo` in schema `bar`, these properties would result in a staging table name of `load_bar.stage_xxx_foo` for the staging table. The Tungsten generated column containing the `seqno`, if present, would be named `tungsten_seqno`.

Note

Staging tables are by default in the same schema as the table they update. You can put them in a different schema using the `stageSchemaPrefix` property as shown in the example.

7.2.5.2. Whole Record Staging

Whole record staging loads the entire CSV file into an identical table, then runs queries to apply rows to the base table or tables in the data warehouse. One of the strengths of whole record staging is that it allows you to construct a merge script that can handle any combination of `INSERT`, `UPDATE`, or `DELETE` operations. A weakness is that whole record staging can result in sub-optimal I/O for workloads that consist mostly of `INSERT` operations.

For example, suppose we have a base table created by the following `CREATE TABLE` command:

```
CREATE TABLE `mydata` (
  `id` int(11) NOT NULL,
  `f_data` float DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

A whole record staging table would look as follows.

```
CREATE TABLE `stage_xxx_croc_mydata` (
  `tungsten_opcode` char(1) DEFAULT NULL,
  `tungsten_seqno` int(11) DEFAULT NULL,
  `tungsten_row_id` int(11) DEFAULT NULL,
  `id` int(11) NOT NULL,
  `f_data` float DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Note that this table does not have a primary key defined. Most data warehouses do not use primary keys and many of them do not even permit it in the create table syntax.

Note also that the non-primary columns must permit nulls. This is required for deletes, which contain only the Tungsten generated columns plus the primary key.

7.2.5.3. Delete Key Staging

Another approach is to load `INSERT` rows directly into the base data warehouse tables without staging. All you need to stage is the keys for deleted records. This reduces I/O considerably for workloads that have mostly inserts. The downside is that it may require introduce ordering dependencies between `DELETE` and `INSERT` operations that require special handling by upstream applications to generate transactions that will load without conflicts.

Delete key staging tables can be as simple as the follow example:

```
CREATE TABLE `stage_xxx_croc_mydata` (
  `id` int(11) NOT NULL,
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

7.2.5.4. Staging Table Generation

Tungsten does not generate staging tables automatically. Creation of staging tables is the responsibility of users, but using the `ddlsync` tool with the right template can be simplified.

7.2.6. Character Sets

Character sets are a headache in batch loading because all updates are written and read from CSV files, which can result in invalid transactions along the replication path. Such problems are very difficult to debug. Here are some tips to improve chances of happy replicating.

- Use UTF8 character sets consistently for all string and text data.
- Force Tungsten to convert data to Unicode rather than transferring strings:

```
shell> tpm ... --mysql-use-bytes-for-string=false
```

- When starting the replicator for MySQL replication, include the following option `tpm` file:

```
shell> tpm ... --java-file-encoding=UTF8
```

7.2.7. CSV Formats

7.2.8. Time Zones

Time zones are another headache when using batch loading. For best results applications should standardize on a single time zone, preferably UTC, and use this consistently for all data. To ensure the Java VM outputs time data correctly to CSV files, you must set the JVM time zone to be the same as the standard time zone for your data. Here is the JVM setting in wrapper.conf:

```
# To ensure consistent handling of dates in heterogeneous and batch replication
# you should set the JVM timezone explicitly. Otherwise the JVM will default
# to the platform time, which can result in unpredictable behavior when
# applying date values to slaves. GMT is recommended to avoid inconsistencies.
wrapper.java.additional.5=-Duser.timezone=GMT
```

Note

Beware that MySQL has two very similar data types: `TIMESTAMP` and `DATETIME`. Timestamps are stored in UTC and convert back to local time on display. Datetimes by contrast do not convert back to local time. If you mix timezones and use both data types your time values will be inconsistent on loading.

7.3. Additional Configuration and Deployment Options

7.3.1. Deploying Multiple Replicators on a Single Host

It is possible to install multiple replicators on the same host. This can be useful, either when building complex topologies with multiple services, and in heterogeneous environments where you are reading from one database and writing to another that may be installed on the same single server.

When installing multiple replicator services on the same host, different values must be set for the following configuration parameters:

- RMI network port used for communicating with the replicator service.

Set through the `--rmi-port` [363] parameter to `tpm`. Note that RMI ports are configured in pairs; the default port is 10000, port 10001 is used automatically. When specifying an alternative port, the subsequent port must also be available. For example, specifying port 10002 also requires 10003.

- THL network port used for exchanging THL data.

Set through the `--thl-port` parameter to `tpm`. The default THL port is 2112. This option is required for services operating as masters (extractors).

- Master THL port, i.e. the port from which a slave will read THL events from the master

Set through the `--master-thl-port` parameter to `tpm`. When operating as a slave, the explicit THL port should be specified to ensure that you are connecting to the THL port correctly.

- Master hostname

Set through the `--master-thl-host` parameter to `tpm`. This is optional if the master hostname has been configured correctly through the `--master` [351] parameter.

- Installation directory used when the replicator is installed.

Set through the `--install-directory` [347] or `--install-directory` [347] parameters to `tpm`. This directory must have been created, and be configured with suitable permissions before installation starts. For more information, see [Section B.3.3, "Directory Locations and Configuration"](#).

For example, to create two services, one that reads from MySQL and another that writes to MongoDB on the same host:

1. Extract the Tungsten Replicator software into a single directory.
2. Extractor reading from MySQL:

```
shell> ./tools/tpm configure extractor \
--install-directory=/opt/extractor \
--master-host1 \
--members=host1 \
--replication-password=password \
--replication-user=tungsten \
--start=true
```

This is a standard configuration using the default ports, with the directory `/opt/extractor`.

3. Reset the configuration:

```
shell> tpm configure defaults --reset
```

4. Applier for writing to MongoDB:

```
shell> ./tools/tpm configure applier \
--datasource-type=mongodb \
--install-directory=/opt/applier \
--master-host1 \
--members=host1 \
--start=true \
--topology=master-slave \
--rmi-port=10002 \
--master-thl-port=2112 \
--master-thl-host=host1 \
--thl-port=2113
```

In this configuration, the master THL port is specified explicitly, along with the THL port used by this replicator, the RMI port used for administration, and the installation directory `/opt/applier`.

When multiple replicators have been installed, checking the replicator status through `trepctl` depends on the replicator executable location used. If `/opt/extractor/tungsten/tungsten-replicator/bin/trepctl`, the extractor service status will be reported. If `/opt/applier/tungsten/tungsten-replicator/bin/trepctl` is used, then the applier service status will be reported.

Alternatively, a specific replicator can be checked by explicitly specifying the RMI port of the service. For example, to check the extractor service:

```
shell> treptl -port 10000 status
```

Or to check the applier service:

```
shell> treptl -port 10002 status
```

When an explicit port has been specified in this way, the executable used is irrelevant. Any valid `treptl` instance will work.

7.4. Deploying SSL Secured Replication and Administration

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

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

You will also need an SSL certificate. These can either be self-generated or obtained from an official signing authority. The certificates themselves must be stored within a Java keystore and truststore. To create your certificates and add them to the keystore or truststore, see [Section 7.4.1, "Creating the Truststore and Keystore"](#). Instructions are provided for self-generated, self-signed, and officially signed versions of the necessary certificates.

For JMX RMI authentication, a password file and authentication definition must also be generated. This information is required by the JMX system to support the authentication and encryption process. See [Section 7.4.2, "SSL and Administration Authentication"](#) for more information.

Once the necessary files are available, you need to use `tpm` to install, or update an existing installation with the SSL configuration. See [Section 7.4.3, "Configuring the Secure Service through tpm"](#).

Note

Although not strictly required for installation, it may be useful to have the OpenSSL package installed. This contains a number of tools and utilities for dealing with certificate authority and general SSL certificates.

7.4.1. Creating the Truststore and Keystore

The SSL configuration works through two separate files that define the server and client side of the encryption configuration. Because individual hosts within a Tungsten Replication 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 Replication, all the hosts in the dataservice can use the same keystore and truststore certificates. The `tpm` command will distribute these files along with the configuration when a new installation is deployed, or when updating an existing deployment.

7.4.1.1. Creating Your Own Client and Server Certificates

Because the client and server components of the Tungsten Replication 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 Replication can open the keystore and read the contents.

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

Enter key password for <any>
(RTURN 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 Replication 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: 87dbe1
Valid from: Wed Jul 31 17:15:05 BST 2013 until: Tue Oct 29 16:15:05 GMT 2013
Certificate fingerprints:
    MD5: 8D:8B:F5:66:7E:34:08:5A:05:E7:A5:91:A7:FF:69:7E
    SHA1: 28:3B:E4:14:2C:80:6B:D5:50:9E:18:2A:22:B9:74:C5:C0:CF:C0:19
    SHA256: 1A:8D:83:BF:D3:00:55:58:DC:08:0C:F0:0C:4C:B8:8A:7D:9E:60:5E:C2:3D:6F:16:F1:B4:E8:C2:3C:87:38:26
    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`.

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

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

The two files, the keystore (`keystore.jks`), and truststore (`truststore.ts`), along with their corresponding passwords can be now be used with `tpm` to configure the cluster. See [Section 7.4.3, “Configuring the Secure Service through tpm”](#).

7.4.1.2. Creating a Custom Certificate and Getting it Signed

You can create your own certificate and get it signed by an authority such as VeriSign or Thawte. To do this, the certificate must be created first, then you create a certificate signing request, send this to your signing authority, and then import the signed certificate and the certificate authority certificate into your keystore and truststore.

Create the certificate:

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

Enter key password for <any>
(RTURN if same as keystore password):
```

Create a new signing request the certificate:

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

This creates a certificate request, `certrequest.pem`. This must be sent the to the signing authority to be signed.

- *Official Signing*

Send the certificate file to your signing authority. They will send a signed certificate back, and also include a root CA and/or intermediary CA certificate. Both these and the signed certificate must be included in the keystore and truststore files.

First, import the returned signed certificate:

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

Now install the root CA certificate:

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

Note

If the import of your certificate with `keytool` fails, it may be due to an incompatibility with some versions of OpenSSL, which fail to create suitable certificates for third-party tools. In this case, see [Section 7.4.1.4, “Converting SSL Certificates For keytool”](#) for more information.

And an intermediary certificate if you were sent one:

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

Now export the signed certificate so that it can be added to the truststore. Although you can import the certificate supplied, by exporting the certificate in your keystore for inclusion into your truststore you can ensure that the two certificates will match:

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

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

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

- *Self-Signing*

If you have setup your own certificate authority, you can self-sign the request using [openssl](#):

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

Convert the certificate to a plain PEM certificate:

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

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

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

This certificate can be imported into your keystore and truststore.

To import your signed certificate into your keystore:

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

Then export the certificate for use in your truststore:

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

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

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

This completes the setup of your truststore and keystore. The files created can be used in your [tpm](#) configuration. See [Section 7.4.3, “Configuring the Secure Service through tpm”](#).

7.4.1.3. Using an existing Certificate

If you have an existing certificate (for example with your MySQL, HTTP server or other configuration) that you want to use, you can import that certificate into your truststore and keystore. When using this method, you must import the signed certificate, and the certificate for the signing authority.

When importing the certificate into your keystore and truststore, the certificate supplied by the certificate authority can be used directly, but must be imported alongside the certificate authorities root and/or intermediary certificates. All the certificates must be imported for the SSL configuration to work.

The certificate should be in the PEM format if it is not already. You can convert to the PEM format by using the [openssl](#) tool:

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

First, import the returned signed certificate:

```
shell> keytool -import -file certificate.pem -keypass password \
    -keystore keystore.jks -storepass password
```

Note

If the import of your certificate with [keytool](#) fails, it may be due to an incompatibility with some versions of OpenSSL, which fail to create suitable certificates for third-party tools. In this case, see [Section 7.4.1.4, “Converting SSL Certificates for keytool”](#) for more information.

Now install the root CA certificate:

```
shell> keytool -import -file cacert.pem -keystore password \
    -keystore keystore.jks -storepass password
```

And an intermediary certificate if you were sent one:

```
shell> keytool -import -file intercert.pem -keystore password \
    -keystore keystore.jks -storepass password
```

Now export the signed certificate so that it can be added to the truststore:

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

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

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

7.4.1.4. Converting SSL Certificates for keytool

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

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

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

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

```
shell> openssl pkcs12 -export -in client-cert.pem -inkey client-key.pem >client.p12
Enter Export Password:
Verifying - Enter Export Password:
```

To import the converted certificate into a keystore, specifying the destination keystore name, as well as the source PKCS12 password used in the previous step:

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

The same process can be used to import server certificates into truststore, by converting the server certificate and private key:

```
shell> openssl pkcs12 -export -in server-cert.pem -inkey server-key.pem >server.p12
Enter Export Password:
Verifying - Enter Export Password:
```

Then importing that into a truststore

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

For official CA certificates, the generated certificate information should be valid for importing using [keytool](#), and this file should not need conversion.

7.4.2. SSL and Administration Authentication

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

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

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

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

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

Now a corresponding password configuration must be created using the `tpasswd` tool. By default, plain-text passwords are generated:

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

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

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

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

The password file, and the JMX security properties file will be needed during configuration. See Section 7.4.3, “Configuring the Secure Service through `tpm`”.

7.4.3. Configuring the Secure Service through `tpm`

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

Configuring SSL for THL Only

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

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

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

Where:

- `--thl-ssl [345]`

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

- `--java-keystore-path [349]`

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

- `--java-keystore-password [349]`

The password for the keystore.

- `--java-truststore-path [350]`

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

- `--java-truststore-password [350]`

The password for the truststore.

Note

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

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

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

Configuring SSL for Administration

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

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

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

```
shell> ./tools/tpm update \
  --java-keystore-path=~/keystore.jks \
  --java-keystore-password=password \
  --java-truststore-path=~/truststore.ts \
  --java-truststore-password=password \
  --rmi-ssl=true \
  --rmi-authentication=true \
  --rmi-user=tungsten \
  --java-jmxremote-access-path=~/jmxremote.access \
  --java-passwordstore-path=~/password.store
```

Where:

- `--rmi-ssl` [345]

If set to `true`, enables RMI SSL encryption.

- `--rmi-authentication` [344]

If set to `true`, enables authentication for the RMI service.

- `--rmi-user` [363]

The user that will be used when performing administration. This should match the username used when creating the password file and security properties.

- `--java-jmxremote-access-path` [349]

The path to the file containing the JMX RMI configuration, as configured in [Section 7.4.2, “SSL and Administration Authentication”](#).

- `--java-passwordstore-path` [349]

The location of the password file created when setting the password, as described in [Section 7.4.2, “SSL and Administration Authentication”](#).

- `--java-keystore-path` [349]

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

- `--java-keystore-password` [349]

The password for the keystore.

- `--java-truststore-path` [350]

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

- `--java-truststore-password [350]`

The password for the truststore.

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

SSL Settings During an Upgrade

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

Configuring SSL for THL and Administration

To configure both JMX and THL SSL encrypted communication, you must specify the SSL and JMX security properties. The SSL properties are the same as those used for enabling SSL on THL, but adding the necessary configuration parameters for the JMX settings:

```
shell> ./tools/tpm update \
--thl-ssl=true \
--rmi-ssl=true \
--java-keystore-path=~/keystore.jks \
--java-keystore-password=password \
--java-truststore-path=~/truststore.ts \
--java-truststore-password=password \
--rmi-authentication=true \
--rmi-user=tungsten \
--java-jmxremote-access-path=~/jmxremote.access \
--java-passwordstore-path=~/password.store
```

This configures SSL and security for authentication. These options for `tpm` can be used to update an existing installation, or defined when creating a new deployment.

Important

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

7.5. Deployment Security

Tungsten Replication supports and is by default configured to use SSL, TLS and certificates for both communication and authentication for all components within the system. This security is enabled by default and includes:

- Authentication between command-line tools (`trepctl`), and between and background services.
- SSL/TLS between command-line tools and background services.
- SSL/TLS between Tungsten Replicator and datasources.
- File permissions and access by all components.

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

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

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

7.5.1. Disabling Security

Because security is enabled by default, there may be situations where the security must be disabled for the entire installation.

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

- `--disable-security-controls` [342]

This has the same effect as adding `--file-protection-level=none` [346], `--rmi-ssl=false` [345], `--thl-ssl=false` [345], `--rmi-authentication=false` [344].

- `--file-protection-level=none` [346]

Disables file level protection, including ownership and file mode settings.

- `--rmi-ssl=false` [345]

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` [344]

Disables the use of authentication when accessing and controlling services.

- `--thl-ssl=false` [345]

Disables the use of SSL/TLS for THL transmission between replicators.

7.5.2. Creating Suitable Certificates

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

- Generating a TLS Certificate

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

```
shell> keytool -genkey -alias tls \
    -validity 365 \
    -keyalg RSA -keystore /etc/tungsten/tls.jks \
    -dname "cn=Continuent, ou=IT, o=VMware, c=US" \
    -storepass mykeystorepass -keypass mykeystorepass
```

7.5.3. Installing from a Staging Host with Manually Generated Certificates

Follow the steps in [Section 7.5.2, “Creating Suitable Certificates”](#) to create the TLS certificate.

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

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

7.5.4. Installing via INI File with Manually Generated Certificates

Follow the steps in [Section 7.5.2, “Creating Suitable Certificates”](#) to create the TLS certificate.

- Transfer the generated certificates to the same path on all hosts.
- Update your configuration to specify these certificates and the keystore password:

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

7.5.5. Replacing the TLS Certificate from a Staging Directory

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

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

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

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

Then perform an update and replace the entire release directory:

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

7.5.6. Removing TLS Encryption from a Staging Directory

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

```
shell> tpm configure SERVICE \  
    --thl-ssl=false \  
    --rmi-ssl=false \  
    --rmi-authentication=false
```

Then perform an update and replace the entire release directory:

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

Chapter 8. Operations Guide

There are a number of key operations that enable you to monitor and manage your replication cluster. Tungsten Replicator includes a small number of tools that can help with this process, including the core `trepctl` command, for controlling the replication system, and `thl`, which provides an interface to the Tungsten History Log and information about the changes that have been recorded to the log and distributed to the slaves.

During the installation process the file `/opt/continuent/share/env.sh` will have been created which will seed the shell with the necessary `$PATH` and other details to more easily manage your cluster. You can load this script manually using:

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

Once loaded, all of the tools for controlling and monitoring your replicator installation should be part of your standard `PATH`.

8.1. The Tungsten Replication Home Directory

After installing Tungsten Replication the home directory will be filled with a set of new directories. The home directory is specified by `--home-directory` [347] or `--install-directory` [347]. 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 Replication.
- `backups` - Storage for backup files created through `trepctl`. See [Section D.1.1, “The backups Directory”](#) for more information.
- `thl` - Storage for THL files created by the replicator. Each replication service gets a dedicated sub-directory for storing THL files. See [Section D.1.5, “The thl Directory”](#) for more information.
- `relay` - Temporary storage for downloaded MySQL binary logs before they are converted into THL files.
- `share` - Storage for files that must persist between different software versions. The `env.sh` script will setup your shell environment to allow easy access to Tungsten Replication tools.

8.2. Establishing the Shell Environment

The tools required to operate Tungsten Replication 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` [361] 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` [346] and `--profile-script` [361] 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` [346] the `trepctl` command would be accessed as `mm_trepctl`.

8.3. Checking Replication Status

To check the replication status you can use the `trepctl` command. This accepts a number of command-specific verbs that provide status and control information for your configured cluster. The basic format of the command is:

```
shell> repctl [-host hostname] command
```

The `-host` option is not required, and enables you to check the status of a different host than the current node.

To get the basic information about the currently configured services on a node and current status, use the `services` verb command:

```
shell> trepctl services
Processing services command...
NAME          VALUE
----          -----
appliedLastSeqno: 211
appliedLatency : 17.66
role           : slave
serviceName    : firstrep
serviceType    : local
started        : true
state          : ONLINE
Finished services command...
```

In the above example, the output shows the last sequence number and latency of the host, in this case a slave, compared to the master from which it is processing information. In this example, the last sequence number and the latency between that sequence being processed on the master and applied to the slave is 17.66 seconds. You can compare this information to that provided by the master, either by logging into the master and running the same command, or by using the host command-line option:

```
shell> trepctl -host host1 services
Processing services command...
NAME          VALUE
----          -----
appliedLastSeqno: 365
appliedLatency : 0.614
role           : master
serviceName    : firstrep
serviceType    : local
started        : true
state          : ONLINE
Finished services command...
```

By comparing the `appliedLastSeqno` for the master against the value on the slave, it is possible to determine that the slave and the master are not yet synchronized.

For a more detailed output of the current status, use the `status` command, which provides much more detailed output of the current replication status:

```
shell> trepctl status
Processing status command...
NAME          VALUE
----          -----
appliedLastEventId   : mysql-bin.000064:000000002757461:0
appliedLastSeqno     : 212
appliedLatency       : 263.43
channels           : 1
clusterName         : default
currentEventId      : NONE
currentTimeMillis   : 1365082088916
dataServerHost       : host2
extensions          :
latestEpochNumber   : 0
masterConnectUri    : thl://host1:2112/
masterListenUri     : thl://host2:2112/
maximumStoredSeqNo  : 724
minimumStoredSeqNo  : 0
offlineRequests     : NONE
pendingError         : NONE
pendingErrorCode     : NONE
pendingErrorEventId  : NONE
pendingErrorSeqno    : -1
pendingExceptionMessage: NONE
pipelineSource       : thl://host1:2112/
relativeLatency     : 655.915
resourcePrecedence   : 99
rmiPort              : 10000
role                 : slave
seqnoType            : java.lang.Long
serviceName          : firstrep
serviceType          : local
simpleServiceName    : firstrep
siteName             : default
sourceId             : host2
state                : ONLINE
timeInStateSeconds   : 893.32
uptimeSeconds         : 9370.031
version              : Tungsten Replicator 5.0.1 build 136
Finished status command...
```

Similar to the host specification, `trepctl` provides information for the default service. If you have installed multiple services, you must specify the service explicitly:

```
shell> trepctl -service servicename status
```

If the service has been configured to operate on an alternative management port, this can be specified using the `-port` option. The default is to use port 10000.

The above command was executed on the slave host, `host2`. Some key parameter values from the generated output:

- `appliedLastEventId`

This shows the last event from the source event stream that was applied to the database. In this case, the output shows that source of the data was a MySQL binary log. The portion before the colon, `mysql-bin.000064` is the filename of the binary log on the master. The portion after the colon is the physical location, in bytes, within the binary log file.

- `appliedLastSeqno`

The last sequence number for the transaction from the Tungsten stage that has been applied to the database. This indicates the last actual transaction information written into the slave database.

When using parallel replication, this parameter returns the minimum applied sequence number among all the channels applying data.

- `appliedLatency`

The `appliedLatency` is the latency between the commit time and the time the last committed transaction reached the end of the corresponding pipeline within the replicator.

In replicators that are operating with parallel apply, `appliedLatency` indicates the latency of the trailing channel. Because the parallel apply mechanism does not update all channels simultaneously, the figure shown may trail significantly from the actual latency.

- `masterConnectUri`

On a master, the value will be empty.

On a slave, the URI of the master Tungsten Replicator from which the transaction data is being read from. The value supports multiple URLs (separated by comma) for topologies with multiple masters.

- `maximumStoredSeqNo`

The maximum transaction ID that has been stored locally on the machine in the THL. Because Tungsten Replicator operates in stages, it is sometimes important to compare the sequence and latency between information being ready from the source into the THL, and then from the THL into the database. You can compare this value to the `appliedLastSeqno`, which indicates the last sequence committed to the database. The information is provided at a resolution of milliseconds.

- `pipelineSource`

Indicates the source of the information that is written into the THL. For a master, `pipelineSource` is the MySQL binary log. For a slave, `pipelineSource` is the THL of the master.

- `relativeLatency`

The `relativeLatency` is the latency between now and timestamp of the last event written into the local THL. An increasing `relativeLatency` indicates that the replicator may have stalled and stopped applying changes to the dataserver.

- `state`

Shows the current status for this node. In the event of a failure, the status will indicate that the node is in a state other than `ON-LINE` [207]. The `timeInStateSeconds` will indicate how long the node has been in that state, and therefore how long the node may have been down or unavailable.

The easiest method to check the health of your cluster is to compare the current sequence numbers and latencies for each slave compared to the master. For example:

```
shell> treptctl -host host2 status|grep applied
appliedLastEventId      : mysql-bin.000076:0000000087725114;0
appliedLastSeqno        : 2445
appliedLatency          : 252.0
...
shell> treptctl -host host1 status|grep applied
appliedLastEventId      : mysql-bin.000076:0000000087725114;0
appliedLastSeqno        : 2445
appliedLatency          : 2.515
```

Note

For parallel replication and complex multi-service replication structures, there are additional parameters and information to consider when checking and confirming the health of the cluster.

The above indicates that the two hosts are up to date, but that there is a significant latency on the slave for performing updates.

Tungsten Replicator Schema

Tungsten Replicator creates and updates information in a special schema created within the database which contains more specific information about the replication information transferred. The schema is named according to the servicename of the replication configuration, for example if the server is `firstrep`, the schema will be `tungsten_firstrep`.

The sequence number of the last transferred and applied transaction is recorded in the `trep_commit_seqno` table.

8.3.1. Understanding Replicator States

Each node within the cluster will have a specific state that indicates whether the node is up and running and servicing requests, or whether there is a fault or problem. Understanding these states will enable you to clearly identify the current operational status of your nodes and cluster as a whole.

A list of the possible states for the replicator includes:

- [START \[207\]](#)

The replicator service is starting up and reading the replicator properties configuration file.

- [OFFLINE: NORMAL \[207\]](#)

The node has been deliberately placed into the offline mode by an administrator. No replication events are processed, and reading or writing to the underlying database does not take place.

- [OFFLINE: ERROR \[207\]](#)

The node has entered the offline state because of an error. No replication events are processed, and reading or writing to the underlying database does not take place.

- [GOING-ONLINE: PROVISIONING \[207\]](#)

The replicator is currently reading provisioning information from the master database before entering the [ONLINE \[207\]](#) state.

- [GOING-ONLINE: RESTORING \[207\]](#)

The replicator is preparing to go online and is currently restoring data from a backup.

- [GOING-ONLINE: SYNCHRONIZING \[207\]](#)

The replicator is preparing to go online and is currently preparing to process any outstanding events from the incoming event stream. This mode occurs when a slave has been switched online after maintenance, or in the event of a temporary network error where the slave has reconnected to the master.

- [ONLINE \[207\]](#)

The node is currently online and processing events, reading incoming data and applying those changes to the database as required. In this mode the current status and position within the replication stream is recorded and can be monitored. Replication will continue until an error or administrative condition switches the node into the [OFFLINE \[207\]](#) state.

- [GOING-OFFLINE \[207\]](#)

The replicator is processing any outstanding events or transactions that were in progress when the node was switched offline. When these transactions are complete, and the resources in use (memory, network connections) have been closed down, the replicator will switch to the [OFFLINE: NORMAL \[207\]](#) state. This state may also be seen in a node where auto-enable is disabled after a start or restart operation.

In general, the state of a node during operation will go through a natural progression within certain situations. In normal operation, assuming no failures or problems, and no management requested offline, a node will remain in the [ONLINE \[207\]](#) state indefinitely.

Maintenance on Tungsten Replicator or the dataserver must be performed while in the [OFFLINE \[207\]](#) state. In the [OFFLINE \[207\]](#) state, write locks on the THL and other files are released, and reads or writes from the dataserver are stopped until the replicator is [ONLINE \[207\]](#) again.

8.3.2. Replicator States During Operations

During a maintenance operation, a node will typically go through the following states at different points of the operation:

Operation	State
Node operating normally	ONLINE [207]

Operation	State
Administrator puts node into offline state	GOING-OFFLINE [207]
Node is offline	OFFLINE: NORMAL [207]
Administrator puts node into online state	GOING-ONLINE: SYNCHRONIZING [207]
Node catches up with master	ONLINE [207]

In the event of a failure, the sequence will trigger the node into the error state and then recovery into the online state:

Operation	State
Node operating normally	ONLINE [207]
Failure causes the node to go offline	OFFLINE: ERROR [207]
Administrator fixes error and puts node into online state	GOING-ONLINE: SYNCHRONIZING [207]
Node catches up with master	ONLINE [207]

During an error state where a backup of the data is restored to a node in preparation of bringing the node back into operation:

Operation	State
Node operating normally	ONLINE [207]
Failure causes the node to go offline	OFFLINE: ERROR [207]
Administrator restores node from backup data	GOING-ONLINE: RESTORING [207]
Once restore is complete, node synchronizes with the master	GOING-ONLINE: SYNCHRONIZING [207]
Node catches up with master	ONLINE [207]

8.3.3. Changing Replicator States

You can manually change the replicator states on any node by using the `trepcctl` command.

To switch to the `OFFLINE` [207] state if you are currently `ONLINE` [207]:

```
shell> trepcctl offline
```

Unless there is an error, no information is reported. The current state can be verified using the `trepcctl status`:

```
shell> trepcctl status
Processing status command...
...
state          : OFFLINE: NORMAL
timeInStateSeconds : 21.409
uptimeSeconds   : 935.072
```

To switch back to the `ONLINE` [207] state:

```
shell> trepcctl online
```

When using replicator states in this manner, the replication between hosts is effectively paused. Any outstanding events from the master will be replicated to the slave with the replication continuing from the point where the node was switched to the `OFFLINE` [207] state. The sequence number and latency will be reported accordingly, as seen in the example below where the node is significantly behind the master:

```
shell> trepcctl status
Processing status command...
NAME           VALUE
-----
appliedLastEventId : mysql-bin.000004:000000005162941;0
appliedLastSeqno : 21
appliedLatency  : 179.366
```

8.4. 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.

8.4.1. Identifying a Transaction Mismatch

When a mismatch occurs, the replicator service will indicate that there was a problem applying a transaction on the slave. The replication process stops applying changes to the slave when the first transaction fails to be applied to the slave. This prevents multiple-statements from failing.

When checking the replication status with `trepctl`, the `pendingError` and `pendingExceptionMessage` will show the error indicating the failure to insert the statement. For example:

```
shell> repctl status
...
pendingError          : Event application failed: seqno=120 fragno=0 message=java.sql.SQLException: »
    Statement failed on slave but succeeded on master
pendingErrorCode      : NONE
pendingErrorEventId   : mysql-bin.000012:000000000012967:0
pendingErrorSeqno     : 120
pendingExceptionMessage: java.sql.SQLException: Statement failed on slave but succeeded on master
    insert into messages values (0,'Trial message','Jack','Jill',now())
...
...
```

The `trepsvc.log` log file will also contain the error information about the failed statement. For example:

```
...
INFO  | jvm 1    | 2013/06/26 10:14:12 | 2013-06-26 10:14:12,423 [firstcluster -
    q-to-dbms-0] INFO pipeline.SingleThreadStageTask Performing emergency
    rollback of applied changes
INFO  | jvm 1    | 2013/06/26 10:14:12 | 2013-06-26 10:14:12,424 [firstcluster -
    q-to-dbms-0] INFO pipeline.SingleThreadStageTask Dispatching error event:
    Event application failed: seqno=120 fragno=0 message=java.sql.SQLException:
    Statement failed on slave but succeeded on master
INFO  | jvm 1    | 2013/06/26 10:14:12 | 2013-06-26 10:14:12,424 [firstcluster -
    pool-2-thread-1] ERROR management.OpenReplicatorManager Received error notification,
    shutting down services :
INFO  | jvm 1    | 2013/06/26 10:14:12 | Event application failed: seqno=120 fragno=0
    message=java.sql.SQLException: Statement failed on slave but succeeded on master
INFO  | jvm 1    | 2013/06/26 10:14:12 | insert into messages values (0,'Trial message',
    'Jack','Jill',now())
INFO  | jvm 1    | 2013/06/26 10:14:12 | com.continuent.tungsten.replicator.applier.ApplierException:
    java.sql.SQLException: Statement failed on slave but succeeded on master
...
...
```

Once the error or problem has been found, the exact nature of the error should be determined so that a resolution can be identified:

1. Identify the reason for the failure by examining the full error message. Common causes are:

- **Duplicate primary key**

A row or statement is being inserted or updated that already has the same insert ID or would generate the same insert ID for tables that have auto increment enabled. The insert ID can be identified from the output of the transaction using `thl`. Check the 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:

```
shell> repctl status
...
pendingError          : Event application failed: seqno=10 fragno=0 »
    message=java.sql.SQLException: Statement failed on slave but succeeded on master
pendingErrorCode      : NONE
pendingErrorEventId   : mysql-bin.000032:0000000000001872:0
pendingErrorSeqno     : 10
pendingExceptionMessage: java.sql.SQLException: Statement failed on slave but succeeded on master
    insert into myent values (0,'Test Message')
...
...
```

The error can be generated when an insert or update has taken place on the slave rather than on the master.

To resolve this issue, check the full THL for the statement that failed. The information is provided in the error message, but full examination of the THL can help with identification of the full issue. For example, to view the THL for the sequence number:

```
shell> thl list -seqno 10
SEQ# = 10 / FRAG# = 0 (last frag)
- TIME = 2014-01-09 16:47:40.0
- EPOCH# = 1
- EVENTID = mysql-bin.000032:0000000000001872:0
- SOURCEID = host1
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=firstcluster;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- SQL(0) = SET INSERT_ID = 2
- OPTIONS = [##charset = UTF-8, autocommit = 1, sql_auto_is_null = 0, foreign_key_checks = 1, >
    unique_checks = 1, sql_mode = '', character_set_client = 33, collation_connection = 33, >
    collation_server = 8]
- SCHEMA = test
- SQL(1) = insert into myent values (0,'Test Message')
```

In this example, an `INSERT` operation is inserting a new row. The generated insert ID is also shown (in line 9, `SQL(0)`)... Check the destination database and determine the what the current value of the corresponding row:

```
mysql> select * from myent where id = 2;
+----+-----+
| id | msg      |
+----+-----+
| 2  | Other Message |
+----+-----+
1 row in set (0.00 sec)
```

The actual row values are different, which means that either value may be correct. In complex data structures, there may be multiple statements or rows that trigger this error if following data also relies on this value.

For example, if multiple rows have been inserted on the slave, multiple transactions may be affected. In this scenario, checking multiple sequence numbers from the THL will highlight this information.

- **Missing table or schema**

If a table or database is missing, this should be reported in the detailed error message. For example:

```
Caused by: java.sql.SQLSyntaxErrorException: Unable to switch to database »
  'contacts' Error was: Unknown database 'contacts'
```

This error can be caused when maintenance has occurred, a table has failed to be initialized properly, or the

- **Incompatible table or schema**

A modified table structure on the slave can cause application of the transaction to fail if there are missing or different column specifications for the table data.

This particular error can be generated when changes to the table definition have been made, perhaps during a maintenance window.

Check the table definition on the master and slave and ensure they match.

2. Choose a resolution method:

Depending on the data structure and environment, resolution can take one of the following forms:

- **Skip the transaction on the slave**

If the data on the slave is considered correct, or the data in both tables is the same or similar, the transaction from the master to the slave can be skipped. This process involves placing the replicator online and specifying one or more transactions to be skipped or ignored. At the end of this process, the replicator should be in the `ONLINE` [207] state.

For more information on skipping single or multiple transactions, see [Section 8.4.2, “Skipping Transactions”](#).

- **Delete the offending row or rows on the slave**

If the data on the master is considered canonical, then the data on the slave can be removed, and the replicator placed online.

Warning

Deleting data on the slave may cause additional problems if the data is used by other areas of your application, relations to foreign tables.

For example:

```
mysql> delete from myent where id = 2;
```

```
Query OK, 1 row affected (0.01 sec)
```

Now place the replicator online and check the status:

```
shell> trepctl online
```

- **Restore or reprovision the slave**

If the transaction cannot be skipped, or the data safely deleted or modified, and only a single slave is affected, a backup of an existing, working, slave can be taken and restored to the broken slave.

The `tungsten_provision_slave` command automates this process. See [Section 8.5, “Provision or Reprovision a Slave”](#) for more information on reprovisioning.

To perform a backup and restore, see [Section 8.6, “Creating a Backup”](#), or [Section 8.7, “Restoring a Backup”](#). To reprovision a slave from the master or another slave, see `tungsten_provision_slave`.

8.4.2. 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 8.4.1, “Identifying a Transaction Mismatch”](#).

- **Skipping a Single Transaction**

If the error was caused by only a single statement or transaction, the transaction can be skipped using `trepctl online`:

```
shell> trepctl online -skip-seqno 10
```

The individual transaction will be skipped, and the next transaction (11), will be applied to the destination database.

- **Skipping a Transaction Range**

If there is a range of statements that need to be skipped, specify a range by defining the lower and upper limits:

```
shell> trepctl online -skip-seqno 10-20
```

This skips all of the transaction within the specified range, and then applies the next transaction (21) to the destination database.

- **Skipping Multiple Transactions**

If there are transactions mixed in with others that need to be skipped, the specification can include single transactions and ranges by separating each element with a comma:

```
shell> trepctl online -skip-seqno 10,12-14,16,19-20
```

In this example, only the transactions 11, 15, 17 and 18 would be applied to the target database. Replication would then continue from transaction 21.

Regardless of the method used to skip single or multiple transactions, the status of the replicator should be checked to ensure that replication is online.

8.5. Provision or Reprovision a Slave

The command performs three operations automatically:

1. Performs a backup of a remote slave
2. Copies the backup to the current host
3. Restores the backup

Warning

When using `tungsten_provision_slave` you must be logged in to the slave that has failed or that you want to reprovision. You cannot reprovision a slave remotely.

To use `tungsten_provision_slave`:

1. Log in to the failed slave.
2. Select the active slave within the dataservice that you want to use to reprovision the failed slave. You may use the master but this will impact performance on that host. If you use MyISAM tables the operation will create some locking in order to get a consistent snapshot.
3. Run `tungsten_provision_slave` specifying the source you have selected:

```
shell> tungsten_provision_slave --source=host2
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 9.19, “The `tungsten_provision_slave` Script”](#).

8.6. Creating a Backup

The `trepctl backup` command backs up a datasource using the default backup tool. During installation, `xtrabackup-full` will be used if `xtrabackup` has been installed. Otherwise, the default backup tool used is `mysqldump`.

Important

For consistency, all backups should include a copy of all `tungsten_SERVICE` schemas. This ensures that when the Tungsten Replicator service is restarted, the correct start points for restarting replication are recorded with the corresponding backup data. Failure to include the `tungsten_SERVICE` schemas may prevent replication from being restart effectively.

Backing up a datasource can occur while the replicator is online:

```
shell> trepctl backup
Backup of dataSource 'host3' succeeded; uri=storage://file-system/store-0000000001.properties
```

By default the backup is created on the local filesystem of the host that is backed up in the `backups` directory of the installation directory. For example, using the standard installation, the directory would be `/opt/continuent/backups`. An example of the directory content is shown below:

```
total 130788
drwxrwxr-x 2 tungsten tungsten      4096 Apr  4 16:09 .
drwxrwxr-x 3 tungsten tungsten      4096 Apr  4 11:51 ..
-rw-r--r-- 1 tungsten tungsten       71 Apr  4 16:09 storage.index
-rw-r--r-- 1 tungsten tungsten 133907646 Apr  4 16:09 store-0000000001-mysqldump_2013-04-04_16-08_42.sql.gz
-rw-r--r-- 1 tungsten tungsten      317 Apr  4 16:09 store-0000000001.properties
```

For information on managing backup files within your environment, see [Section D.1.1, “The `backups` Directory”](#).

The `storage.index` contains the backup file index information. The actual backup data is stored in the GZipped file. The properties of the backup file, including the tool used to create the backup, and the checksum information, are location in the corresponding `.properties` file. Note that each backup and property file is uniquely numbered so that it can be identified when restoring a specific backup.

A backup can also be initiated and run in the background by adding the & (ampersand) to the command:

```
shell> trepctl backup &
Backup of dataSource 'host3' succeeded; uri=storage://file-system/store-0000000001.properties
```

8.6.1. 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 `--rep1-backup-method=xtrabackup-full` [330] setting.

To use `xtrabackup-full` without changing the configuration, specify the backup agent to `trepctl backup`:

```
shell> trepctl backup --backup xtrabackup-full
Backup completed successfully; URI=storage://file-system/store-000000006.properties
```

8.6.2. Using a Different Directory Location

The default backup location the `backups` directory of the Tungsten Replication 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.

8.6.3. Creating an External Backup

There are several considerations to take into account when you are using a tool other than Tungsten Replication 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 8.7.2, “Restoring an External Backup”](#) for more details on how to do that. When using `mysqldump` use the `--master-data=2` option. The `xtrabackup` tool will print the binary log position in the command output.

Using `mysqldump` can be a very simple way to take consistent backup. Be aware that it can cause locking on MyISAM tables so running it against your master will cause application delays. The example below shows the bare minimum for arguments you should provide:

```
shell> mysqldump --opt --single-transaction --all-databases --add-drop-database --master-data=2
```

8.7. 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 8.5, “Provision or Reprovision a Slave”](#).

To restore a backup, use the `trepctl restore` command :

1. Put the replication service offline using `trepctl`:

```
shell> trepctl offline
```

2. Restore the backup using `trepctl restore`:

```
shell> trepctl restore
```

3. Put the replication service online using `trepctl`:

```
shell> trepctl online
```

By default, the restore process takes the latest backup available for the host being restored. Tungsten Replication *does not* automatically locate the latest backup within the dataservice across all datasources.

8.7.1. Restoring a Specific Backup

To restore a specific backup, specify the location of the corresponding properties file using the format:

```
storage://storage-type/location
```

For example, to restore the backup from the filesystem using the information in the properties file `store-0000000004.properties`, login to the failed host:

1. Put the replication service offline using `trepctl`:

```
shell> trepctl offline
```

2. Restore the backup using `trepctl restore`:

```
shell> trepctl restore \
-uri storage://file-system/store-0000000004.properties
```

3. Put the replication service online using `trepctl`:

```
shell> trepctl online
```

8.7.2. Restoring an External Backup

If a backup has been performed outside of Tungsten Replication, for example from filesystem snapshot or a backup performed outside of the dataservice, follow these steps:

1. Put the replication service offline using `trepctl`:

```
shell> trepctl offline
```

2. Reset the THL, either using `thl` or by deleting the files directly:

```
shell> thl -service alpha purge
```

3. Restore the data or files using the external tool. This may require the database server to be stopped. If so, you should restart the database server before moving to the next step.

Note

The backup must be complete and the `tungsten` specific schemas must be part of the recovered data, as they are required to restart replication at the correct point. See [Section 8.6.3, “Creating an External Backup”](#) for more information on creating backups.

4. There is some additional work if the backup was taken of the master server. There may be a difference between the binary log position of the master and what is represented in the `trep_commit_seqno`. If these values are the same, you may proceed without further work. If not, the content of `trep_commit_seqno` must be updated.

- Retrieve the contents of `trep_commit_seqno`:

```
shell> echo "select seqno,source_id, eventid from tungsten_alpha.trep_commit_seqno" | tpm mysql
seqno source_id eventid
32033674 host1 mysql-bin.000032:000000473860407:-1
```

- Compare the results to the binary log position of the restored backup. For this example we will assume the backup was taken at `mysql-bin.000032:473863524`. Return to the master and find the correct sequence number for that position :

```
shell> ssh host1
shell> thl list -service alpha -low 32033674 -headers | grep 473863524
```

```
32033678 32030709 0 true 2014-10-17 16:58:11.0 mysql-bin.000032:000000473863524;-1 dbl-east.continuent.com
shell> exit
```

- Return to the slave node and run `tungsten_set_position` to update the `trep_commit_seqno` table :

```
shell> tungsten_set_position --service=alpha --source=host1 --seqno=32033678
```

5. Put the replication service online using `trepctl`:

```
shell> trepctl online
```

8.7.3. Restoring from Another Slave

If a restore is being performed as part of the recovery procedure, consider using the `tungsten_provision_slave` tool. This 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 8.5, "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`:

```
shell> trepctl backup
Backup of dataSource 'host2' succeeded; uri=storage://file-system/store-000000006.properties
```

2. Copy the backup information from `host2` to `host3`. See [Section D.1.1.3, "Copying Backup Files"](#) for more information on copying backup information between hosts. If you are using `xtrabackup` there will be additional files needed before the next step. The example below uses `scp` to copy a `mysqldump` backup:

```
shell> cd /opt/continuent/backups
shell> scp store-[0]*6[.-]* host3:$PWD/
store-000000006-mysqldump-812096863445699665.sql          100% 234MB 18.0MB/s 00:13
store-000000006.properties                                100% 314      0.3KB/s 00:00
```

If you are using `xtrabackup`:

```
shell> cd /opt/continuent/backups/xtrabackup
shell> rsync -aze ssh full_xtrabackup_2014-08-16_15-44_86 host3:$PWD/
```

3. Put the replication service offline using `trepctl`:

```
shell> trepctl offline
```

4. Restore the backup using `trepctl restore` :

```
shell> trepctl restore
```

Note

Check the ownership of files if you have trouble transferring files or restoring the backup. They should be owned by the Tungsten system user to ensure proper operation.

5. Put the replication service online using `trepctl`:

```
shell> trepctl online
```

8.7.4. 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: 13306
130520 14:37:08 [Note]      - '0.0.0.0' resolves to '0.0.0.0';
130520 14:37:08 [Note] Server socket created on IP: '0.0.0.0'.
130520 14:37:08 [ERROR] Fatal error: Can't open and lock privilege tables: Table 'mysql.host' doesn't exist
130520 14:37:08 [ERROR] /usr/sbin/mysqld: File '/var/run/mysqld/mysqld.pid' not found (Errcode: 13)
130520 14:37:08 [ERROR] /usr/sbin/mysqld: Error reading file 'UNKNOWN' (Errcode: 9)
```

```
130520 14:37:08 [ERROR] /usr/sbin/mysqld: Error on close of 'UNKNOWN' (Errcode: 9)
```

Performing a restore operation on this slave may not work. To recover from another running slave, `host3`, the MySQL data files can be copied over to `host2` directly using the following steps:

1. Put the `host2` replication service offline using `trepctl`:

```
shell> trepctl offline
```

2. Put the `host3` replication service offline using `trepctl`:

```
shell> trepctl offline
```

3. Stop the `mysqld` service on `host2`:

```
shell> sudo /etc/init.d/mysql stop
```

4. Stop the `mysqld` service on `host3`:

```
shell> sudo /etc/init.d/mysql stop
```

5. Delete the `mysqld` data directory on `host2`:

```
shell> sudo rm -rf /var/lib/mysql/*
```

6. If necessary, ensure the `tungsten` user can write to the MySQL directory:

```
shell> sudo chmod 777 /var/lib/mysql
```

7. Use `rsync` on `host3` to send the data files for MySQL to `host2`:

```
shell> rsync -aze ssh /var/lib/mysql/* host2:/var/lib/mysql/
```

You should synchronize all locations that contain data. This includes additional folders such as `innodb_data_home_dir` or `innodb_log_group_home_dir`. Check the `my.cnf` file to ensure you have the correct paths.

Once the files have been copied, the files should be updated to have the correct ownership and permissions so that the Tungsten service can read them.

8. Start the `mysqld` service on `host3`:

```
shell> sudo /etc/init.d/mysql start
```

9. Put the `host3` replication service online using `trepctl`:

```
shell> trepctl online
```

10. Update the ownership and permissions on the data files on `host2`:

```
host2 shell> sudo chown -R mysql:mysql /var/lib/mysql
host2 shell> sudo chmod 770 /var/lib/mysql
```

11. Start the `mysqld` service on `host2`:

```
shell> sudo /etc/init.d/mysql start
```

12. Put the `host2` replication service online using `trepctl`:

```
shell> trepctl online
```

8.8. Deploying Automatic Replicator Recovery

Automatic recovery enables the replicator to go back [ONLINE](#) [207] in the event of a transient failure that is triggered during either the [ONLINE](#) [207] or [GOING-ONLINE:SYNCHRONIZING](#) [207] state that would otherwise trigger a change of states to [OFFLINE](#) [207]. For example, connection failures, or restarts in the MySQL service, trigger the replicator to go [OFFLINE](#) [207]. With autorecovery enabled, the replicator will attempt to put the replicator [ONLINE](#) [207] again to keep the service running. Failures outside of these states will not trigger autorecovery.

Autorecovery operates by scheduling an attempt to go back online after a transient failure. If autorecovery is enabled, the process works as follows:

1. If a failure is identified, the replicator attempts to go back online after a specified delay. The delay allows the replicator time to decide whether autorecovery should be attempted. For example, if the MySQL server restarts, the delay gives time for the MySQL server to come back online before the replicator goes back online.

2. Recovery is attempted a configurable number of times. This presents the replicator from continually attempting to go online within a service that has a more serious failure. If the replicator fails to go [ONLINE \[207\]](#) within the configurable reset interval, then the replicator will go to the [OFFLINE \[207\]](#) state.
3. If the replicator remains in the [ONLINE \[207\]](#) state for a configurable period of time, then the automatic recovery is deemed to have succeeded. If the autorecovery fails, then the autorecovery attempts counter is incremented by one.

The configurable parameters are set using `tpm` within the static properties for the replicator:

- [--auto-recovery-max-attempts \[329\]](#)

Sets the maximum number of attempts to automatically recover from any single failure trigger. This prevents the autorecovery mechanism continually attempting autorecover. The current number of attempts is reset if the replicator remains online for the configured reset period.

- [--auto-recovery-delay-interval \[329\]](#)

The delay between entering the [OFFLINE \[207\]](#) state, and attempting autorecovery. On servers that are busy, use some form of network or HA solution, or have high MySQL restart/startup times, this value should be configured accordingly to give the underlying services time to startup again after failure.

- [--auto-recovery-reset-interval \[329\]](#)

The duration after a successful autorecovery has been completed that the replicator must remain in the [ONLINE \[207\]](#) state for the recovery process to be deemed to have succeeded. The number of attempts for autorecovery is reset to 0 (zero) if the replicator stays up for this period of time.

Auto recovery is enabled only when the [--auto-recovery-max-attempts \[329\]](#) parameter is set to a non-zero value.

To enable:

```
shell> tpm update alpha --auto-recovery-max-attempts=5
```

The autorecovery status can be monitored within `trepsvc.log` and through the `autoRecoveryEnabled` and `autoRecoveryTotal` parameters output by `trepctl`. For example:

```
shell> repctl status
Processing status command...
NAME          VALUE
----          -----
...
autoRecoveryEnabled : false
autoRecoveryTotal   : 0
...
```

The above output indicates that the autorecovery service is disabled. The `autoRecoveryTotal` is a count of the number of times the autorecovery has been completed since the replicator has started.

8.9. Migrating and Seeding Data

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

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

1. Deploy Tungsten Replication using the model or system appropriate according to [Chapter 2, Deployment](#). Ensure that the Tungsten Replication is not started automatically by excluding the [--start \[365\]](#) or [--start-and-report \[365\]](#) 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> repctl -service alpha reset -all -y
```

4. On the master

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

```
shell> startall
shell> trepctl online
```

5. 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-ss11
      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 Replication replication. If one of the servers does not show an error, it may be replicating from an intermediate server. If so, you can proceed normally and assume this server stopped at the same position as the host is replicating from.

6. On the master

Take the replicator offline and clear the THL:

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

7. On the master

Start replication, using the *lowest* binary log file and log position from the slave information determined in step 5.

```
shell> trepctl online -from-event 000025:181268
```

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

8. On each slave

- 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
```

- If the final position of MySQL replication matches the lowest across all slaves, start Tungsten Replication 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
```

- Check that replication is operating correctly by using `trepctl status` on the master and each slave to confirm the correct position.
- 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 Replication should be operating as the replication service for your MySQL servers. Use the information in [Chapter 8, Operations Guide](#) to monitor and administer the service.

8.9.2. 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 Replication service, one solution is to create the new Tungsten Replication service, synchronize the content, and then install a service that migrates data from the existing native

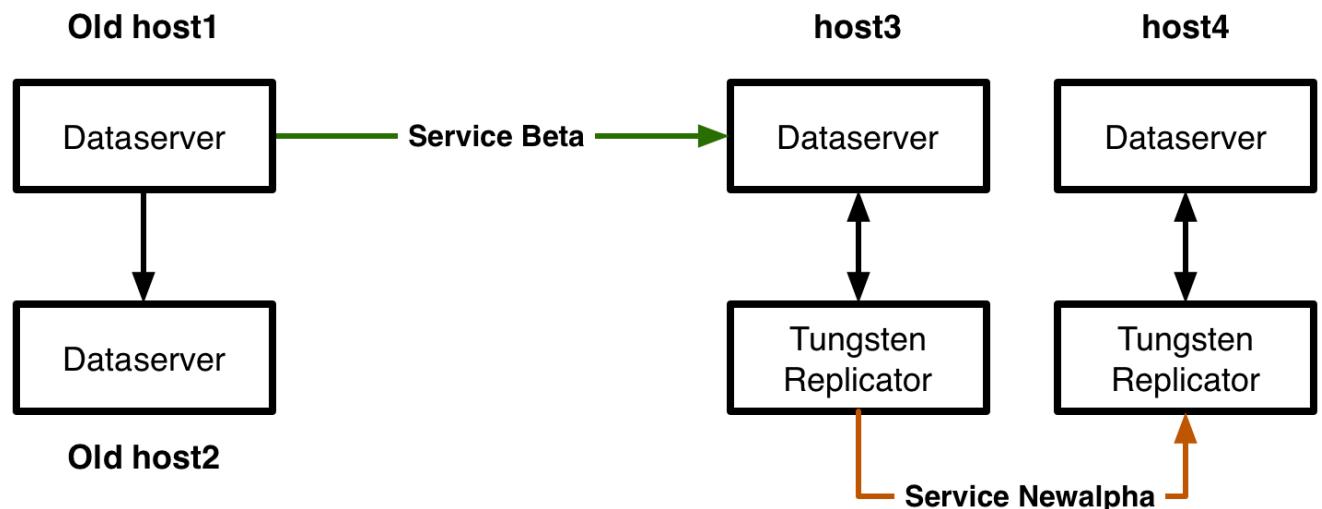
service to the new service while applications are reconfigured to use the new service. The two can then be executed in parallel until applications have been migrated.

The basic structure is shown in [Figure 8.1, “Migration: Migrating Native Replication using a New Service”](#). The migration consists of two steps:

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

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

Figure 8.1. Migration: Migrating Native Replication using a New Service



To configure the service:

1. Stop replication on a slave for the existing native replication installation :

```
mysql> STOP SLAVE;
```

Obtain the current slave position within the master binary log :

```
mysql> SHOW SLAVE STATUS\G
...
      Master_Host: host3
      Master_Log_File: mysql-bin.000002
      Exec_Master_Log_Pos: 559
...
```

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

3. Restart the slave using native replication :

```
mysql> START SLAVE;
```

4. On the master and each slave within the new service, restore the backup data and start the database service
5. Setup the new Tungsten Replication 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`.

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.

8.9.3. 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` [207] 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 `ddlsync` 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 has completed loading :

```
mysql> START SLAVE;
```

7. Monitor replication status on the master and slaves :

```
shell> trepctl status
```

8.9.4. 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.

8.10. Using the Parallel Extractor

Parallel Extractor in 2.2.1. The parallel extractor functionality was added in Tungsten Replicator 2.2.1, and initially supported only extraction from Oracle masters.

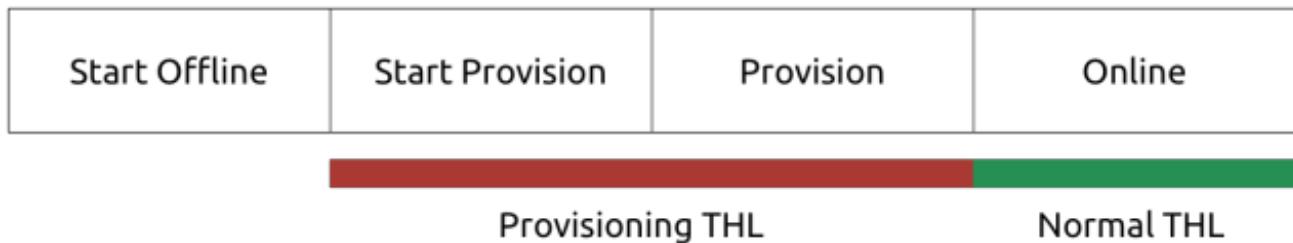
The parallel extractor reads information from the source database schema in chunks and then feeds this information into the THL data stream as row-based `INSERT` operations. When the slave connects, these are applied to the slave database as with a normal `INSERT` operations. The parallel extractor is particularly useful in heterogeneous environments such as Oracle to MySQL where the slave data does already exist on the slave.

The basic provisioning process operates in two stages:

1. Provisioning data is extracted and inserted into the THL. One event is used to contain all of the data from a single table. If the table is too large to be contained in a single event, the data will be distributed over multiple events.
2. Once provisioning has finished, data is extracted from the CDC as normal and added to the THL using the normal THL extraction thread.

This allows existing data to be extracted and processed through the replicator path, including filters within the applier. Once the initial data has been extracted, the change data to be applied. A diagram of the replication scheme at different stages is provided below:

Figure 8.2. Parallel Extractor: Extraction Sequence

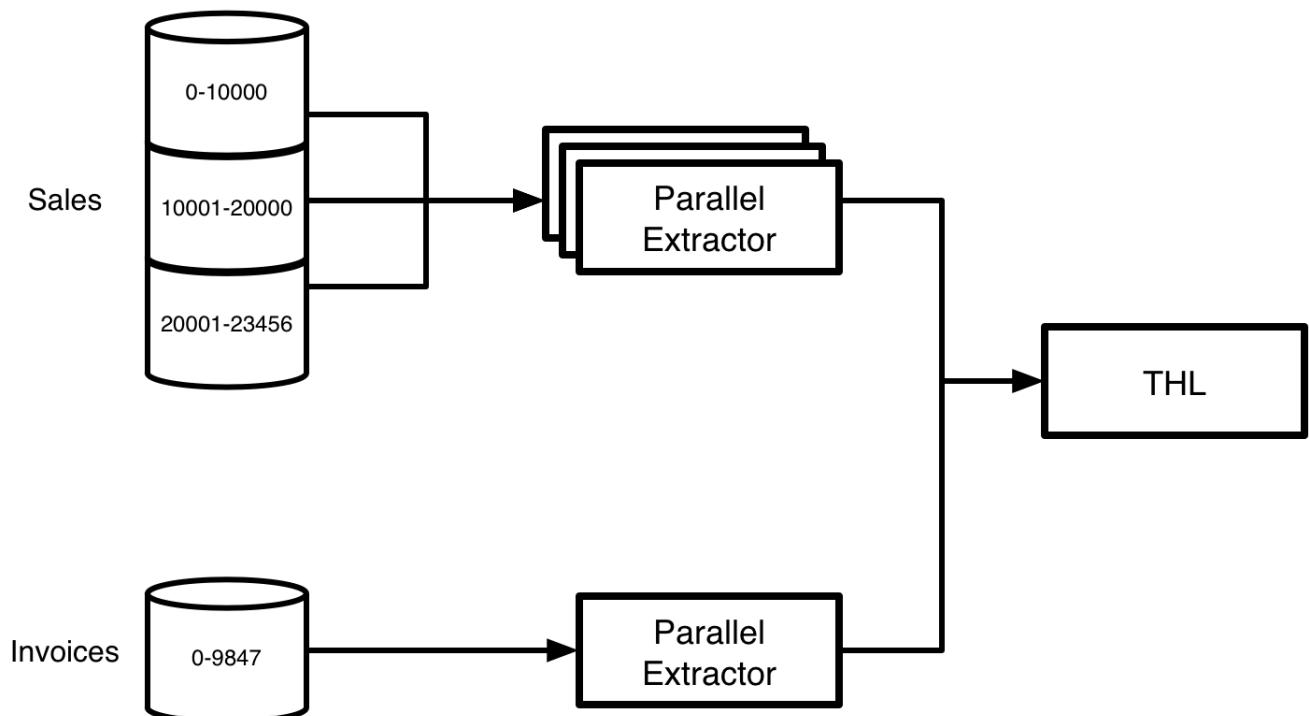


The parallel extractor happens in a multi-threaded process that extracts multiple tables, and multiple ranges from a single table in parallel. A chunking thread identifies all the tables, and also identifies the keys and chunks that can be extracted from each table. It then coordinates the multiple threads:

- Multiple chunks from the source tables are extracted in parallel
- Multiple tables are extracted in parallel

For example, when reading from two different tables in a single schema, the process might look like the figure below:

Figure 8.3. Parallel Extractor: Extraction Operation



Because multiple threads are used to read information from the tables, the process is very quick, although it implies additional load on the source database server, since the queries must load all of the data into memory.

To use the parallel extractor to provision data into the slave, the configuration must be performed as part of the installation process when configuring the master replicator for the first time, or when re-initializing the replicator on a master after a `trepctl reset` operation.

To setup provisioning with parallel extractor:

1. Install master Tungsten Replicator using `tpm`, but do not enable automatic starting (i.e. do not use the `--start` [365] or `--start-and-report` [365] options).
2. Install the slave replicator as normal.
3. On the master:
 - a. Start the replicator in `OFFLINE` [207] mode using `replicator start offline`:

```
shell> replicator start offline
```

- b. Put the replicator into the `ONLINE` [207] state, using the `-provision` option:

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

If you have an identifiable reference number, such as a the system change number or MySQL event, then this can be specified on the command-line to the `trepctl online -provision` command:

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

During the provisioning process, the replicator will show the status `GOING-ONLINE:PROVISIONING` [207] until all of the data has been read from the existing database.

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

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

Important

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

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

8.10.1. Advanced Configuration Parameters

The provisioning process can be controlled using a number of properties that can be configured when the replicator is installed by using `--property` option. For example:

```
shell> tpm update alpha \
  --property=replicator.extractor.parallel-extractor.ChunkDefinitionFile=/opt/continuent/share/chunks.csv
```

- `replicator.extractor.parallel-extractor.ChunkDefinitionFile`

The path to a file that contains a specification of the tables that should be included in the extraction. If no file is specified then all tables and extracted for provisioning.

The format of the chunk definition file is a Comma Separated Values (CSV) file, with each line containing the schema name, optional table name, optional chunk size and optional column list to be extracted. For example:

```
SALES
```

Would extract all tables within the schema `SALES`:

```
SALES,INVOICES
```

Would extract only the `INVOICES` table from the `SALES` schema.

```
SALES,INVOICES,1000
```

Would extract only the `INVOICES` table from the `SALES` schema, only in chunks of 1000 rows.

To extract only specific columns, add the column list to the end of the schema, table and chunk size. For example:

```
SALES, INVOICES, 1000, INVOICENO, CUSTOMERID, VALUE
```

Multiple lines can be used to define a range of schemas, tables, and chunk size combinations.

- `replicator.extractor.parallel-extractor.chunk_size`

The chunk size defines the number of rows that are extracted from the source tables and inserted into the THL. This number should be adjusted when extracting very large rows from the source tables.

- `replicator.extractor.parallel-extractor.add_truncate_table`

If set to `true`, a `TRUNCATE` statement is inserted into the THL before the first row data is inserted into the THL. This empties the target table before the provision data is replicated.

- `replicator.extractor.parallel-extractor.extract_channels`

Specifies the number of simultaneous threads that will be used to extract information. Defaults to a single thread.

- `replicator.extractor.parallel-extractor.queue_size`

The number of events buffered inside the parallel extractor before they are inserted into the THL. Defaults to 20 transactions.

8.11. Switching Master Hosts

In the event of a failure, or during the process of performing maintenance on a running cluster, the roles of the master and slaves within the cluster may need to be swapped.

The basic sequence of operation for switching master and slaves is:

1. Switch slaves to offline state
2. Switch master to offline status
3. Set an existing slave to have the `master` (in [Tungsten Clustering for MySQL 5.0 Manual]) role
4. Set each slave with the `slave` (in [Tungsten Clustering for MySQL 5.0 Manual]) role, updating the master URI (where the THL logs will be loaded) to the new master host
5. Switch the new master to online state
6. Switch the new slaves to online state

Depending on the situation when the switch is performed, the switch can be performed either without waiting for the hosts to be synchronized (i.e. in a failure situation), or by explicitly waiting for slave that will be promoted to the master role.

To perform an ordered switch of the master. In the example below, master host `host1` will be switched to `host3`, and the remaining hosts (`host1` and `host2`) will be configured as slaves to the new master:

1. If you are performing the switch as part of maintenance or other procedures, you should perform a safe switch, ensuring the slaves are up to date with the master:
 - a. Synchronize the database and the transaction history log. This will ensure that the two are synchronized, and provide you with a sequence number to ensure the slaves are up to date:

```
shell> treptl -host host1 flush
Master log is synchronized with database at log sequence number: 1405
```

Keep a note of the sequence number.

- b. For each current slave within the cluster, wait until the master sequence number has been reached, and then put the slave into the offline state:

```
shell> treptl -host host2 wait -applied 1405
shell> treptl -host host2 offline
shell> treptl -host host3 wait -applied 1405
shell> treptl -host host3 offline
```

If the master has failed, or once the slaves and masters are in sync, you can perform the remainder of the steps to execute the physical switch.

2. Switch the master to the offline state:

```
shell> treptl -host host1 offline
```

3. Configure the new designated master to the *master* (in [Tungsten Clustering for MySQL 5.0 Manual]) role:

```
shell> trepctl -host host3 setrole -role master
```

Switch the master to the online state:

```
shell> trepctl -host host3 online
```

4. For each slave, set the role to slave, supplying the URI of the THL service on the master:

```
shell> trepctl -host host1 setrole -role slave -uri thl://host3:2112
```

In the above example we are using the default THL port (2112).

Put the new slave into the online state:

```
shell> trepctl -host host1 online
```

Repeat for the remaining slaves:

```
shell> trepctl -host host2 setrole -role slave -uri thl://host3:2112
shell> trepctl -host host2 online
```

Once completed, the state of each host can be checked to confirm that the switchover has completed successfully:

```
appliedLastEventId      : mysql-bin.000005:0000000000002100;0
appliedLastSeqno        : 1405
appliedLatency          : 0.094
dataServerHost          : host1
masterConnectUri        : thl://host3:2112
role                  : slave
state                 : ONLINE
-----
appliedLastEventId      : mysql-bin.000005:0000000000002100;0
appliedLastSeqno        : 1405
appliedLatency          : 0.149
dataServerHost          : host2
masterConnectUri        : thl://host3:2112
role                  : slave
state                 : ONLINE
-----
appliedLastEventId      : mysql-bin.000005:0000000000002100;0
appliedLastSeqno        : 1405
appliedLatency          : 0.061
dataServerHost          : host3
masterConnectUri        : thl://host1:2112/
role                  : master
state                 : ONLINE
```

In the above, `host1` and `host2` are now getting the THL information from `host1`, with each acting as a slave to the `host1` as master.

8.12. Configuring Parallel Replication

The replication stream within MySQL is by default executed in a single-threaded execution model. Using Tungsten Replicator, the application of the replication stream can be applied in parallel. This improves the speed at which the database is updated and helps to reduce the effect of slaves lagging behind the master which can affect application performance. Parallel replication operates by distributing the events from the replication stream from different database schemas in parallel on the slave. All the events in one schema are applied in sequence, but events in multiple schemas can be applied in parallel. Parallel replication will not help in those situations where transactions operate across schema boundaries.

Parallel replication supports two primary options:

- Number of parallel channels — this configures the maximum number of parallel operations that will be performed at any one time. The number of parallel replication streams should match the number of different schemas in the source database, although it is possible to exhaust system resources by configuring too many. If the number of parallel threads is less than the number of schemas, events are applied in a round-robin fashion using the next available parallel stream.
- Parallelization type — the type of parallelization to be employed. The disk method is the recommended solution.

Parallel replication can be enabled during installation by setting the appropriate options during the initial configuration and installation. To enable parallel replication after installation, you must configure each host as follows:

1. Put the replicator offline:

```
shell> trepctl offline
```

2. Reconfigure the replication service to configure the parallelization:

```
shell> tpm update firstrep --host=host2 \
    --channels=5 --svc-parallelization-type=disk
```

3. Then restart the replicator to enable the configuration:

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

The current configuration can be confirmed by checking the `channels` configured in the status information:

```
shell> treptctl status
Processing status command...
NAME          VALUE
----          -----
appliedLastEventId : mysql-bin.000005:0000000000004263;0
appliedLastSeqno : 1416
appliedLatency   : 1.0
channels        : 5
...
```

More detailed information can be obtained by using the `treptctl status -name stores` command, which provides information for each of the parallel replication queues:

```
shell> treptctl status -name stores
Processing status command (stores)...
NAME          VALUE
----          -----
activeSeqno    : 0
doChecksum     : false
flushIntervalMillis : 0
fsyncOnFlush   : false
logConnectionTimeout : 28800
logDir         : /opt/continuent/thl/firstrep
logFileRetainMillis : 604800000
logFileSize    : 100000000
maximumStoredSeqNo : 1416
minimumStoredSeqNo : 0
name           : thl
readOnly       : false
storeClass     : com.continuent.tungsten.replicator.thl.THL
timeoutMillis  : 2147483647
NAME          VALUE
----          -----
criticalPartition : -1
discardCount    : 0
estimatedOfflineInterval : 0.0
eventCount      : 0
headSeqno       : -1
intervalGuard   : AtomicIntervalGuard (array is empty)
maxDelayInterval : 60
maxOfflineInterval : 5
maxSize         : 10
name           : parallel-queue
queues          : 5
serializationCount : 0
serialized       : false
stopRequested   : false
store.0         : THLParallelReadTask task_id=0 thread_name=store-thl-0 »
    hi_seqno=0 lo_seqno=0 read=0 accepted=0 discarded=0 events=0
store.1         : THLParallelReadTask task_id=1 thread_name=store-thl-1 »
    hi_seqno=0 lo_seqno=0 read=0 accepted=0 discarded=0 events=0
store.2         : THLParallelReadTask task_id=2 thread_name=store-thl-2 »
    hi_seqno=0 lo_seqno=0 read=0 accepted=0 discarded=0 events=0
store.3         : THLParallelReadTask task_id=3 thread_name=store-thl-3 »
    hi_seqno=0 lo_seqno=0 read=0 accepted=0 discarded=0 events=0
store.4         : THLParallelReadTask task_id=4 thread_name=store-thl-4 »
    hi_seqno=0 lo_seqno=0 read=0 accepted=0 discarded=0 events=0
storeClass      : com.continuent.tungsten.replicator.thl.THLParallelQueue
syncInterval    : 10000
Finished status command (stores)...
```

To examine the individual threads in parallel replication, you can use the `treptctl status -name shards` status option, which provides information for each individual shard thread:

```
Processing status command (shards)...
NAME          VALUE
----          -----
appliedLastEventId: mysql-bin.000005:0000000013416909;0
appliedLastSeqno : 1432
appliedLatency   : 0.0
```

```

eventCount      : 28
shardId        : cheffy
stage          : q-to-dbms
...
Finished status command (shards)...

```

8.13. Performing Database or OS Maintenance

When performing database or operating system maintenance, datasources should be temporarily disabled by placing them into the [OFFLINE](#) [207] state. For maintenance operations on a master, the current master should be switched, the required maintenance steps performed, and then the master switched back. Detailed steps are provided below for different scenarios.

8.13.1. Performing Maintenance on a Single Slave

To perform maintenance on a single slave, you should ensure that your application is not using the slave, perform the necessary maintenance, and then re-enable the slave within your application.

The steps are:

1. Put the replicator into the offline state to prevent replication and changes being applied to the database:

```
shell> trepctl -host host1 offline
```

To perform operating system maintenance, including rebooting the system, the replicator can be stopped completely:

```
shell> replicator stop
```

2. Perform the required maintenance, including updating the operating system, software or hardware changes.
3. Validate the server configuration :

```
shell> tpm validate
```

4. Put the replicator back online:

```
shell> trepctl -host host1 online
```

Or if you have stopped the replicator, restart the service again:

```
shell> replicator start
```

Once the datasource is back online, monitor the status of the service and ensure that the replicator has started up and that transactions are being extracted or applied.

8.13.2. Performing Maintenance on a Master

Maintenance, including MySQL admin or schema updates, should not be performed directly on a master as this may upset the replication and therefore availability and functionality of the slaves which are reading from the master.

To effectively make the modifications, you should switch the master host, then operate on the master as if it were slave, removing it from the replicator service configuration. This helps to minimize any problems or availability that might be caused by performing operations directly on the master.

The complete sequence and commands required to perform maintenance on an active master are shown in the table below. The table assumes a dataservice with three datasources:

Step	Description	Command	host1	host2	host3
1	Initial state		Master	Slave	Slave
2	Switch master to host2	See Section 8.11, "Switching Master Hosts"	Slave	Master	Slave
3	Put slave into OFFLINE state	trepctl -host host1 offline	Offline	Master	Slave
4	Perform maintenance		Offline	Master	Slave
5	Validate the host1 server configuration	tpm validate	Offline	Master	Slave
6	Put the slave online	trepctl -host host1 online	Slave	Master	Slave
7	Ensure the slave has caught up	trepctl -host host1 status	Slave	Master	Slave
8	Switch master back to host1	See Section 8.11, "Switching Master Hosts"	Master	Slave	Slave

8.13.3. Performing Maintenance on an Entire Dataservice

To perform maintenance on all of the machines within a replicator service, a rolling sequence of maintenance must be performed carefully on each machine in a structured way. In brief, the sequence is as follows

1. Perform maintenance on each of the current 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:

Step	Description	Command	host1	host2	host3
1	Initial state		Master	Slave	Slave
2	Set the slave <code>host2</code> offline	<code>trepctl -host host2 offline</code>	Master	Offline	Slave
3	Perform maintenance		Master	Offline	Slave
4	Validate the <code>host2</code> server configuration	<code>tpm validate</code>	Master	Offline	Slave
5	Set slave <code>host2</code> online	<code>trepctl -host host2 online</code>	Master	Slave	Slave
6	Ensure the slave (<code>host2</code>) has caught up	<code>trepctl -host host2 status</code>	Master	Slave	Slave
7	Set the slave <code>host3</code> offline	<code>trepctl -host host3 offline</code>	Master	Slave	Offline
8	Perform maintenance		Master	Slave	Offline
9	Validate the <code>host3</code> server configuration	<code>tpm validate</code>	Master	Slave	Offline
10	Set the slave <code>host3</code> online	<code>trepctl -host host3 online</code>	Master	Slave	Slave
11	Ensure the slave (<code>host3</code>) has caught up	<code>trepctl -host host3 status</code>	Master	Slave	Slave
12	Switch master to <code>host2</code>	See Section 8.11, "Switching Master Hosts"	Slave	Master	Slave
13	Set the slave <code>host1</code>	<code>trepctl -host host1 offline</code>	Offline	Master	Slave
14	Perform maintenance		Offline	Master	Slave
15	Validate the <code>host1</code> server configuration	<code>tpm validate</code>	Offline	Master	Slave
16	Set the slave <code>host1</code> online	<code>trepctl -host host1 online</code>	Slave	Master	Slave
17	Ensure the slave (<code>host1</code>) has caught up	<code>trepctl -host host1 status</code>	Master	Slave	Slave
18	Switch master back to <code>host1</code>	See Section 8.11, "Switching Master Hosts"	Master	Slave	Slave

8.13.4. Upgrading or Updating your JVM

When upgrading your JVM version or installation, care should be taken as changing the JVM will momentarily remove and replace required libraries and components which may upset the operation of Tungsten Replication while the upgrade or update takes place.

For this reason, JVM updates or changes must be treated as an OS upgrade or event, requiring a master switch and controlled stopping of services during the update process.

A sample sequence for this in a 3-node cluster is described below:

Step	Description	Command	host1	host2	host3
1	Initial state		Master	Slave	Slave
2	Stop all services on <code>host2</code> .	<code>stopall</code>	Master	Stopped	Slave
3	Update the JVM		Master	Stopped	Slave

Step	Description	Command	host1	host2	host3
4	Start all services on <code>host2</code> slave.	<code>startall</code>	Master	Slave	Slave
5	Stop all services on <code>host3</code> .	<code>stopall</code>	Master	Slave	Stopped
6	Update the JVM		Master	Slave	Stopped
7	Start all services on <code>host3</code> slave.	<code>startall</code>	Master	Slave	Slave
8	Stop all services on <code>host1</code> .	<code>stopall</code>	Stopped	Slave	Slave
9	Update the JVM		Stopped	Slave	Slave
10	Start all services on <code>host1</code> Master.	<code>startall</code>	Master	Slave	Slave

The status of all services on all hosts should be checked to ensure they are running and operating as normal once the update has been completed.

8.14. Making Online Schema Changes

Similar to the maintenance procedure, schema changes to an underlying dataserver may need to be performed on dataservers that are not part of an active dataservice. Although many inline schema changes, such as the addition, removal or modification of an existing table definition will be correctly replicated to slaves, other operations, such as creating new indexes, or migrating table data between table definitions, is best performed individually on each dataserver while it has been temporarily taken out of the dataservice.

Note

If you are attempting an Online schema change and running in a MSMM environment, then you should follow the steps in [Performing Schema Changes](#).

The basic process is to temporarily put each slave offline, perform the schema update, and then put the slave online and monitor it and catch up.

Operations supported by these online schema changes must be backwards compatible. Changes to the schema on slaves that would otherwise break the replication cannot be performed using the online method.

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.

Step	Description	Command	host1	host2	host3
1	Initial state		Master	Slave	Slave
2	Set the slave <code>host2</code> offline	<code>trepctl -host host2 offline</code>	Master	Offline	Slave
3	Connect to dataserver for <code>host2</code> and update schema		Master	Offline	Slave
4	Set the slave online	<code>trepctl -host host2 online</code>	Master	Slave	Slave
5	Ensure the slave (<code>host2</code>) has caught up	<code>trepctl -host host2 status</code>	Master	Slave	Slave
6	Set the slave <code>host3</code> offline	<code>trepctl -host host3 offline</code>	Master	Slave	Offline
7	Connect to dataserver for <code>host3</code> and update schema		Master	Slave	Offline
8	Set the slave (<code>host3</code>) online	<code>trepctl -host host3 online</code>	Master	Slave	Slave
9	Ensure the slave (<code>host3</code>) has caught up	<code>trepctl -host host3 status</code>	Master	Slave	Slave
10	Switch master to <code>host2</code>	See Section 8.11, "Switching Master Hosts"	Slave	Master	Slave
11	Set the slave <code>host1</code> offline	<code>trepctl -host host1 offline</code>	Offline	Master	Slave
12	Connect to dataserver for <code>host1</code> and update schema		Offline	Master	Slave
13	Set the slave <code>host1</code> online	<code>trepctl -host host1 online</code>	Slave	Master	Slave
14	Ensure the slave (<code>host1</code>) has caught up	<code>trepctl -host host1 status</code>	Master	Slave	Slave
15	Switch master back to <code>host1</code>	See Section 8.11, "Switching Master Hosts"	Master	Slave	Slave

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.

8.15. Upgrading Tungsten Replicator

To upgrade an existing installation of Tungsten Replicator, the upgrade must be performed from a staging directory containing the new release. The process updates the Tungsten Replicator software and restarts the replicator service using the current configuration.

For installations using Tungsten Replicator 2.1.1 and later where `tpm` has been used to perform the installation, use the instructions in [Section 8.15.2, "Upgrading Tungsten Replication using tpm"](#).

8.15.1. Upgrading Tungsten Replication to use tpm

The `tpm` is used to set configuration information and create, install and update deployments. Using `tpm` provides a number of key benefits:

- Simpler deployments, and easier configuration and configuration updates for existing installations.
- Easier multi-host deployments.
- Faster deployments and updates; `tpm` performs commands and operations in parallel to speed up installation and updates.
- Simplified update procedure. `tpm` can update all the hosts within your configured service, automatically taking hosts offline, updating the software and configuration, and putting hosts back online.
- Extensive checking and verification of the environment and configuration to prevent potential problems and issues.

To upgrade your installation to use `tpm`, the following requirements must be met:

- Existing installation should be a master/slave, multi-master or fan-in configuration. Star topologies may not upgrade correctly.

Once the prerequisites have been met, use the following upgrade steps:

1. First fetch your existing configuration into the `tpm` system. This collects the configuration from one or more hosts within your service and creates a suitable configuration:

To fetch the configuration:

```
shell> ./tools/tpm fetch --user=tungsten --hosts=host1,host2,host3,host4 \
--release-directory=autodetect
```

Where:

- `--user` [370] is the username used by Tungsten Replication on local and remote hosts.
- `--hosts` [347] is a comma-separated list of hosts in your configuration. Hosts should be listed explicitly. The keyword `autodetect` can be used, which will search existing configuration files for known hosts.
- `--release-directory` (or `--directory`) is the directory where the current Tungsten Replication installation is installed. Specifying `autodetect` searches a list of common directories for an existing installation. If the directory cannot be found using this method, it should be specified explicitly.

The process will collect all the configuration information for the installed services on the specified or autodetected hosts, creating the file, `deploy.cfg`, within the current staging directory.

2. Once the configuration information has been loaded and configured, update your existing installation to the new version and `tpm` based configuration by running the update process:

If there any problems with the configuration, inconsistent configuration parameters, associated deployment issues (such as problems with MySQL configuration), or warnings about the environment, it will be reported during the update process. If the configuration discovery cannot be completed, the validation will fail. For example, the following warnings were generated upgrading an existing Tungsten Replication installation:

```
shell> ./tools/tpm update
...
WARN >> host1 >> Unable to run '/etc/init.d/mysql status' or »
      the database server is not running (DatasourceBootScriptCheck)
.
WARN >> host3 >> Unable to run '/etc/init.d/mysql status' or »
      the database server is not running (DatasourceBootScriptCheck)
WARN >> host1 >> "sync_binlog" is set to 0 in the MySQL »
```

```

configuration file for tungsten@host1:3306 (WITH PASSWORD) this setting »
can lead to possible data loss in a server failure (MySQLSettingsCheck)

WARN  >> host2 >> "sync_binlog" is set to 0 in the MySQL »
configuration file for tungsten@host2:3306 (WITH PASSWORD) this »
setting can lead to possible data loss in a server failure (MySQLSettingsCheck)

WARN  >> host4 >> "sync_binlog" is set to 0 in the MySQL »
configuration file for tungsten@host4:3306 (WITH PASSWORD) this setting »
can lead to possible data loss in a server failure (MySQLSettingsCheck)
WARN  >> host3 >> "sync_binlog" is set to 0 in the MySQL »
configuration file for tungsten@host3:3306 (WITH PASSWORD) this setting »
can lead to possible data loss in a server failure (MySQLSettingsCheck)
WARN  >> host2 >> MyISAM tables exist within this instance - These »
tables are not crash safe and may lead to data loss in a failover (MySQLMyISAMCheck)
WARN  >> host4 >> MyISAM tables exist within this instance - These »
tables are not crash safe and may lead to data loss in a failover (MySQLMyISAMCheck)
ERROR >> host1 >> You must enable sudo to use xtrabackup
ERROR >> host3 >> You must enable sudo to use xtrabackup
WARN  >> host3 >> MyISAM tables exist within this instance - These »
tables are not crash safe and may lead to data loss in a failover (MySQLMyISAMCheck)

#####
# Validation failed
#####
#####
# Errors for host3
#####
ERROR >> host3 >> You must enable sudo to use xtrabackup (XtrabackupSettingsCheck)
Add --root-command-prefix=true to your command
-----
#####
# Errors for host1
#####
ERROR >> host1 >> You must enable sudo to use xtrabackup (XtrabackupSettingsCheck)
Add --root-command-prefix=true to your command
-----
```

These issues should be fixed before completing the update. Use [tpm configure](#) to update settings within Tungsten Replication if necessary before performing the update. Some options can be added to the update statement (as in the above example) to update the configuration during the upgrade process. Issues with MySQL should be corrected before performing the update.

Once the upgrade has been completed, the Tungsten Replication service will be updated to use [tpm](#). For more information on using [tpm](#), see [Chapter 10, The tpm Deployment Command](#). When upgrading Tungsten Replication in future, use the instructions provided in [Section 8.15.2, "Upgrading Tungsten Replication using tpm"](#).

8.15.2. Upgrading Tungsten Replication using tpm

To upgrade an existing installation on Tungsten Replication, 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.

The method for the upgrade process depends on whether [ssh](#) access is available with [tpm](#). If [ssh](#) access has been enabled, use the method in [Upgrading with ssh Access \[230\]](#). If [ssh](#) access has not been configured, use [Upgrading without ssh Access \[231\]](#)

Upgrading with ssh Access

To perform an upgrade of an entire cluster, where you have [ssh](#) access to the other hosts in the cluster:

1. On your staging server, download the release package.
2. Unpack the release package:

```
shell> tar zxf tungsten-replicator-5.0.1-136.tar.gz
```

3. Change to the unpackaged directory:

```
shell> cd tungsten-replicator-5.0.1-136
```

4. Fetch a copy of the existing configuration information:

```
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 `fetch` command to `tpm` supports the following arguments:

- `--hosts` [347]

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` [370]

The username to be used when logging in to other hosts.

- `--directory`

The installation directory of the current Tungsten Replication installation. If `autodetect` is specified, then `tpm` will look for the installation directory by checking any running Tungsten Replication processes.

The current configuration information will be retrieved to be used for the upgrade:

```
shell> ./tools/tpm fetch --hosts=host1,host2,host3 --directory=/opt/continuent --user=tungsten
...
NOTE  >> Configuration loaded from host1,host2,host3
```

5. Optionally check that the current configuration matches what you expect by using `tpm reverse`:

```
shell> ./tools/tpm reverse
# Options for the alpha data service
tools/tpm configure alpha \
--enable-slave-th1-listener=false \
--install-directory=/opt/continuent \
--master=host1 \
--members=host1,host2,host3 \
--replication-password=password \
--replication-user=tungsten \
--start=true
```

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 Replication updates. These issues should be addressed and the `update` command re-executed.

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

```
shell> ./tools/tpm update
.....
#####
# Next Steps
#####
Once your services start successfully replication will begin.
To look at services and perform administration, run the following command
from any database server.

$CONTINUENT_ROOT/tungsten/tungsten-replicator/bin/trepctl services

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

NOTE  >> Command successfully completed
```

The update process should now be complete. The current version can be confirmed by using `trepctl status`.

Upgrading 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 replicator service.

To upgrade all the hosts within a replicator service using this method:

1. Upgrade the configured slaves in the replicator service first
2. Switch the current master to one of the upgraded slaves, using the method shown in [Section 8.11, “Switching Master Hosts”](#)

3. Upgrade the master
4. Switch the master back to the original master, using the method shown in [Section 8.11, "Switching Master Hosts"](#)

For more information on performing maintenance across a cluster, see [Section 8.13.3, "Performing Maintenance on an Entire Dataservice"](#).

To upgrade a single host with **tpm**:

1. Download the release package.
2. Unpack the release package:

```
shell> tar zxf tungsten-replicator-5.0.1-136.tar.gz
```

3. Change to the unpackaged directory:

```
shell> cd tungsten-replicator-5.0.1-136
```

4. Execute **tpm update**, specifying the installation directory. This will update only this host:

```
shell> ./tools/tpm update --directory=/opt/continuent
NOTE  >> Configuration loaded from host1
.

#####
# Next Steps
#####
Once your services start successfully replication will begin.
To look at services and perform administration, run the following command
from any database server.

$CONTINUENT_ROOT/tungsten/tungsten-replicator/bin/trepctl services

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

NOTE  >> Command successfully completed
```

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

8.15.3. Installing an Upgraded JAR Patch

Warning

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

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

1. Determine your staging directory or untarred installation directory:

```
shell> tpm query staging
```

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

```
shell> cd tungsten-replicator-5.0.1-136
```

2. Change to the correct directory:

```
shell> cd tungsten-replicator/lib
```

3. Copy the existing JAR to a backup file:

```
shell> cp tungsten-replicator.jar tungsten-replicator.jar.orig
```

4. Copy the replacement JAR into the directory:

```
shell> cp /tmp/tungsten-replicator.jar .
```

5. Change back to the root directory of the staging directory:

```
shell> cd ../../..
```

6. Update the release:

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

8.16. Monitoring Tungsten Replication

It is your responsibility to properly monitor your deployments of Tungsten Replication 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 Replication and Tungsten Replicator services are running.
2. Make sure all datasources and replication services are [ONLINE \[207\]](#).
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 Replication 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.

8.16.1. Managing Log Files with logrotate

You can manage the logs generated by Tungsten Replication using [logrotate](#).

- `trepvc.log`

```
/opt/continuent/tungsten/tungsten-replicator/log/trepvc.log {
    notifempty
    daily
    rotate 3
    missingok
    compress
    copytruncate
}
```

8.16.2. Monitoring Status Using cacti

Graphing Tungsten Replicator data is supported through Cacti extensions. These provide information gathering for the following data points:

- Applied Latency
- Sequence Number (Events applied)
- Status (Online, Offline, Error, or Other)

To configure the Cacti services:

1. Download both files from <https://github.com/continuent/monitoring/tree/master/cacti>
2. Place the PHP script into `/usr/share/cacti/scripts`.
3. Modify the installed PHP file with the appropriate `$ssh_user` and `$tungsten_home` location from your installation:
 - `$ssh_user` should match the `>user` used during installation.
 - `>$tungsten_home` is the installation directory and the `tungsten` subdirectory. For example, if you have installed into `/opt/continuent`, use `/opt/continuent/tungsten`.

Add SSH arguments to specify the correct `id_rsa` file if needed.

4. Ensure that the configured `>$ssh_user` has the correct SSH authorized keys to login to the server or servers being monitored. The user must also have the correct permissions and rights to write to the cache directory.
5. Test the script by running it by hand:

```
shell> php -q /usr/share/cacti/scripts/get_replicator_stats.php --hostname repiserver
```

If you are using multiple replication services, add `--service servicename` to the command.

6. Import the XML file as a Cacti template.

- Add the desired graphs to your servers running Tungsten Replication. If you are using multiple replication services, you'll need to specify the desired service to graph. A graph must be added for each individual replication service.

Once configured, graphs can be used to display the activity and availability.

Figure 8.4. Cacti Monitoring: Example Graphs



8.16.3. Monitoring Status Using nagios

In addition to the scripts bundled with the software, there is a Ruby gem available with expanded checks and a mechanism to add custom checks. See <https://github.com/continuent/continuent-monitors-nagios> for more details.

Integration with Nagios is supported through a number of scripts that output information in a format compatible with the [Nagios NRPE plugin](#). Using the plugin the check commands, such as `check_tungsten_latency` can be executed and the output parsed for status information.

The available commands are:

- [check_tungsten_latency](#)
- [check_tungsten_online](#)
- [check_tungsten_services](#)

To configure the scripts to be executed through NRPE:

1. Install the Nagios NRPE server.
2. Start the NRPE daemon:

```
shell> sudo /etc/init.d/nagios-nrpe-server start
```

3. Add the IP of your Nagios server to the `/etc/nagios/nrpe.cfg` configuration file. For example:

```
allowed_hosts=127.0.0.1,192.168.2.20
```

4. Add the Tungsten check commands that you want to execute to the `/etc/nagios/nrpe.cfg` configuration file. For example:

```
command[check_tungsten_online]=/opt/continuent/tungsten/cluster-home/bin/check_tungsten_online
```

5. Restart the NRPE service:

```
shell> sudo /etc/init.d/nagios-nrpe-server start
```

6. If the commands need to be executed with superuser privileges, the `/etc/sudo` or `/etc/sudoers` file must be updated to enable the commands to be executed as root through `sudo` as the `nagios` user. This can be achieved by updating the configuration file, usually performed by using the `visudo` command:

```
nagios      ALL=(tungsten) NOPASSWD: /opt/continuent/tungsten/cluster-home/bin/check*
```

In addition, the `sudo` command should be added to the Tungsten check commands within the Nagios `nrpe.cfg`, for example:

```
command[check_tungsten_online]=/usr/bin/sudo -u tungsten /opt/continuent/tungsten/cluster-home/bin/check_tungsten_online
```

Restart the NRPE service for these changes to take effect.

7. Add an entry to your Nagios `services.cfg` file for each service you want to monitor:

```
define service {
    host_name database
    service_description check_tungsten_online
    check_command      check_nrpe! -H $HOSTADDRESS$ -t 30 -c check_tungsten_online
    retry_check_interval 1
    check_period       24x7
    max_check_attempts 3
    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.

Chapter 9. Command-line Tools

Tungsten Replication 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 Replication 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
- `ddlscan` — Data definition layer scanner and translator
- `setupCDC.sh` — Setup Oracle Change Data Control services
- `updateCDC.sh` — Update an existing Oracle Change Data Control service

Commands related to the core Tungsten Replicator

- `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 the Hadoop Deployments

- `load-reduce-check` — build DDL, materialize and compare replicated data
- `materialize` — materializer of views of replicated data into tables

9.1. 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 9.1. `check_tungsten_latency` Options

Option	Description
<code>-c</code>	Report a critical status if the latency is above this level
<code>--perfdata</code>	Show the latency performance information
<code>--perslave-perfdata</code>	Show the latency performance information on a per-slave basis
<code>-w</code>	Report a warning status if the latency is above this level

The command outputs information in the following format:

```
LEVEL: DETAIL
```

Where DETAIL includes detailed information about the status report, and LEVEL is:

- CRITICAL — latency on at least one node is above the specified threshold level for a critical report. The host reporting the high latency will be included in the DETAIL portion:

For example:

```
CRITICAL: host2=0.506s
```

- WARNING — latency on at least one node is above the specified threshold level for a warning report. The host reporting the high latency will be included in the DETAIL portion:

For example:

```
WARNING: host2=0.506s
```

- OK — status is OK; the highest reported latency will be included in the output.

For example:

```
OK: All slaves are running normally (max_latency=0.506)
```

The `-w` and `-c` options must be specified on the command line, and the critical figure must be higher than the warning figure. For example:

```
shell> check_tungsten_latency -w 0.1 -c 0.5
CRITICAL: host2=0.506s
```

Performance information can be included in the output to monitor the status. The format for the output is included in the DETAIL block and separates the maximum latency information for each node with a semicolon, and the detail block with a pipe symbol. For example:

```
shell> check_tungsten_latency -w 1 -c 1 --perfdata
OK: All slaves are running normally (max_latency=0.506) | max_latency=0.506;1;1;;
```

Performance information for all the 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;;
```

9.2. The `check_tungsten_online` Command

The `check_tungsten_online` command checks whether all the hosts in a given service are online and running.

Table 9.2. `check_tungsten_online` Options

Option	Description
<code>-h</code>	Display the help text
<code>-port</code>	RMI port for the replicator being checked

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 `treptcl` output.

For example:

```
shell> check_tungsten_online
OK: All services are online
```

9.3. 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 9.3. `check_tungsten_services` Options

Option	Description
<code>-h</code>	Display the help text.
<code>-r</code>	Check the replication services status.

The command outputs information in the following format:

LEVEL: DETAIL

Where DETAIL includes detailed information about the status report, and LEVEL is:

- CRITICAL — status is critical and requires immediate attention.

For example:

CRITICAL: Replicator is not running

- OK — status is OK.

For example:

OK: All services (Replicator) are online

This output is easily parseable by various monitoring tools, including Nagios NRPE, and can be used to monitor the status of your services quickly without resorting to using the full `trepctl` output.

Note

The `check_tungsten_services` only confirms that the services and processes are running; their state is not confirmed. To check state with a similar interface, use the `check_tungsten_online` command.

To check the services:

- To check the replicator services:

```
shell> check_tungsten_services -r
OK: All services (Replicator) are online
```

9.4. The `deployall` Command

The `deployall` tool installs the required startup scripts into the correct location so that all required services can be automatically started and stopped during the startup and shutdown of your server.

To use, the tool should be executed with superuser privileges, either directly using `sudo`, or by logging in as the superuser and running the command directly:

```
shell> sudo deployall
Adding system startup for /etc/init.d/treplicator ...
/etc/rc0.d/K80treplicator -> ../init.d/treplicator
/etc/rc1.d/K80treplicator -> ../init.d/treplicator
/etc/rc6.d/K80treplicator -> ../init.d/treplicator
/etc/rc2.d/S80treplicator -> ../init.d/treplicator
/etc/rc3.d/S80treplicator -> ../init.d/treplicator
/etc/rc4.d/S80treplicator -> ../init.d/treplicator
/etc/rc5.d/S80treplicator -> ../init.d/treplicator
```

The startup scripts are added to the correct runlevels to enable operation during standard startup and shutdown levels.

See [Section 2.7, “Configuring Startup on Boot”](#).

To remove the scripts from the system, use `undeployall`.

9.5. 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 [Chapter 4, “Heterogeneous Deployments”](#).

For example, to generate Oracle DDL from an existing MySQL database:

```
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:

```
ddlscan [-conf path] [-db db] [-opt opt val] [-out file] [-pass secret] [-path path] [-rename file] [-service name] [-tableFile file] [-tables regex] [-template file] [-url jdbcUrl] [-user user]
```

The available options are as follows:

Table 9.4. `ddlscan` Command-line Options

Option	Description
<code>-conf path</code> [239]	Path to a static-{svc}.properties file to read JDBC connection address and credentials
<code>-db db</code> [239]	Database to use (will substitute \${DBNAME} in the URL, if needed)
<code>-opt opt val</code> [240]	Option(s) to pass to template, try: -opt help me
<code>-out file</code> [240]	Render to file (print to stdout if not specified)
<code>-pass secret</code> [239]	JDBC password
<code>-path path</code> [240]	Add additional search path for loading Velocity templates
<code>-rename file</code> [240]	Definitions file for renaming schemas, tables and columns
<code>-service name</code> [239]	Name of a replication service instead of path to config
<code>-tableFile file</code> [240]	New-line separated definitions file of tables to find
<code>-tables regex</code> [240]	Comma-separated list of tables to find
<code>-template file</code> [239]	Specify template file to render
<code>-url jdbcUrl</code> [239]	JDBC connection string (use single quotes to escape)
<code>-user user</code> [239]	JDBC username

`ddlscan` supports three different methods for execution:

- Using an explicit JDBC URL, username and password:

```
shell> ddlscan -user tungsten -url 'jdbc:mysql:thin://tr-hadoop1:13306/test' -user user \
    -pass password ...
```

This is useful when a deployment has not already been installed.

- By specifying an explicit configuration file:

```
shell> ddlscan -conf /opt/continuent/tungsten/tungsten-replicator/conf/static-alpha.properties ...
```

- When an existing deployment has been installed, by specifying one of the active services:

```
shell> ddlscan -service alpha ...
```

In addition, the following two options must be specified on the command-line:

- The template to be used (using the `-template` [239] 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 9.5, “`ddlscan` Supported Templates”](#).
- The `-db` [239] 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> ddlskan -service alpha -template ddl-mysql-oracle.vm -db test
```

9.5.1. Optional Arguments

The following arguments are optional:

- `-tables` [240]

A comma-separate list of the tables to be extracted.

```
shell> ddlskan -service alpha -template ddl-mysql-oracle.vm -db test -tables typetwo,typethree
```

- `-tableFile` [240]

A file containing a list of the files to be extracted. The file should be formatted as Comma Separated Values (CSV), only the first column is extracted. For example, the file:

```
typetwo,Table of type two customer forms
typethree,Table of type three customer forms
```

Could be used with `ddlskan`:

```
shell> ddlskan -service alpha -template ddl-mysql-oracle.vm -db test -tableFile tablelist.txt
```

- `-rename` [240]

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` [240]

The path to additional Velocity templates to be searched when specifying the template name.

- `-opt` [240]

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> ddlskan -service alpha -template ddl-mysql-oracle.vm -db test -opt schemaPrefix mysql_
```

- `-out` [240]

Sends the generated DDL output to a file, in place of sending it to standard output.

- `-help` [240]

Generates the help text of arguments.

9.5.2. Supported Templates and Usage

Table 9.5. `ddlskan` Supported Templates

file	Description
<code>ddl-check-pkeys.vm</code>	Reports which tables are without primary key definitions
<code>ddl-mysql-hive-0.10.vm</code>	Generates DDL from a MySQL host suitable for the base tables in a Hadoop/Hive Environment
<code>ddl-mysql-hive-0.10-staging.vm</code>	Generates DDL from a MySQL host suitable for the staging tables in a Hadoop/Hive Environment
<code>ddl-mysql-hive-metadata.vm</code>	Generates metadata as JSON to be used within a Hadoop/Hive Environment
<code>ddl-mysql-oracle.vm</code>	Generates Oracle schema from a MySQL schema
<code>ddl-mysql-oracle-cdc.vm</code>	Generates Oracle tables with CDC capture information from a MySQL schema
<code>ddl-mysql-redshift.vm</code>	Generates DDL from a MySQL host suitable for the base tables in Amazon Redshift.

file	Description
<code>ddl-mysql-redshift-staging.vm</code>	Generates DDL from a MySQL host suitable for the staging tables in Amazon Redshift.
<code>ddl-mysql-vertica.vm</code>	Generates DDL suitable for the base tables in HP Vertica
<code>ddl-mysql-vertica-staging.vm</code>	Generates DDL suitable for the staging tables in HP Vertica
<code>ddl-oracle-mysql.vm</code>	Generates DDL for MySQL tables from an Oracle schema
<code>ddl-oracle-mysql-pk-only.vm</code>	Generates Primary Key DDL statements from an Oracle database for MySQL

9.5.2.1. `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> ddlskan -template ddl-check-pkeys.vm \
  -user tungsten -pass password -db sales \
  -url jdbc:mysql://localhost:13306/sales
/*
SQL generated on Thu Sep 04 10:23:52 BST 2014 by ./ddlskan utility of Tungsten

url = jdbc:mysql://localhost:13306/sales
user = tungsten
dbName = sales
*/

/* ERROR: sales.dummy1 has no primary key! */
SQL generated on Thu Sep 04 10:23:52 BST 2014 by ./ddlskan utility of Tungsten

url = jdbc:mysql://localhost:13306/sales
user = tungsten
dbName = sales
*

/* ERROR: sales.dummy2 has no primary key! */
SQL generated on Thu Sep 04 10:23:52 BST 2014 by ./ddlskan utility of Tungsten

url = jdbc:mysql://localhost:13306/sales
user = tungsten
dbName = sales
*/
```

For certain environments, particularly heterogeneous replication, the lack of primary keys can lead to inefficient replication, or even fail to replicate data at all.

9.5.2.2. `ddl-mysql-hive-0.10.vm`

Generates DDL suitable for a carbon-copy form of the table from the MySQL host:

```
shell> ddlskan -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 ddlskan utility
--
-- url = jdbc:mysql://tr-hadoop1:13306/test
-- user = tungsten
-- dbName = test
--

DROP TABLE IF EXISTS test.sales;

CREATE TABLE test.sales
(
  id INT,
  salesman STRING,
  planet STRING,
  value DOUBLE
);
```

Wherever possible, the closest Hive equivalent datatype is used for each source datatype, as follows:

MySQL Datatype	Hive Datatype
<code>DATETIME</code>	<code>STRING</code>
<code>TIMESTAMP</code>	<code>TIMESTAMP</code>

MySQL Datatype	Hive Datatype
DATE	STRING
YEAR	INT
TIME	STRING
TINYINT	TINYINT
TINYINT UNSIGNED	SMALLINT
SMALLINT	SMALLINT
SMALLINT UNSIGNED	INT
MEDIUMINT	INT
INT	INT
INT UNSIGNED	BIGINT
BIGINT	BIGINT
BIGINT UNSIGNED	STRING
DECIMAL	STRING
VARCHAR	STRING
CHAR	STRING
BINARY	BINARY
VARBINARY	BINARY
TEXT	STRING
BLOB	BINARY
FLOAT	DOUBLE
DOUBLE	DOUBLE
ENUM	STRING
SET	STRING
BIT	STRING

The template supports the following optional parameters to change behavior:

- `-opt schemaPrefix`

A prefix to be placed in front of all schemas. For example, if called with `schemaPrefix` set to `mysql_`:

```
shell> ddlscan ... -opt schemaPrefix mysql_
```

The schema name will be prefixed, translating the schema name from `sales` into `mysql_sales`.

- `-opt tablePrefix`

A prefix to be placed in front of all schemas. For example, if called with `tablePrefix` set to `mysql_`:

```
shell> ddlscan ... -opt tablePrefix mysql_
```

The table name will be prefixed, translating the tablename from `sales` into `mysql_sales`.

9.5.2.3. `ddl-mysql-hive-0.10-staging.vm`

Staging tables within Hive define the original table columns with additional columns to track the operation type, sequence number, timestamp and unique key for each row. For example, the table `sales` in MySQL:

```
mysql> describe sales;
+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| id   | int(11) | NO  | PRI | NULL | auto_increment |
| salesman | char(20) | YES | | NULL | |
| planet | char(20) | YES | | NULL | |
| value | float | YES | | NULL | |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Generates the following Hive-compatible DDL when using this template:

```
shell> ddlsync -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 ddlsync utility
--
-- url = jdbc:mysql://tr-hadoop1:13306/test
-- user = tungsten
-- dbName = test
--

DROP TABLE IF EXISTS test.stage_xxx_sales;

CREATE EXTERNAL TABLE test.stage_xxx_sales
(
  tungsten_opcode STRING ,
  tungsten_seqno INT ,
  tungsten_row_id INT ,
  tungsten_commit_timestamp TIMESTAMP ,
  id INT,
  salesman STRING,
  planet STRING,
  value DOUBLE )
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\001' ESCAPED BY '\\'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE LOCATION '/user/tungsten/staging/test/sales' ;
```

Wherever possible, the closest Hive equivalent datatype is used for each source datatype, see [ddl-mysql-hive-0.10.vm](#) for more information.

9.5.2.4. [ddl-mysql-hive-metadata.vm](#)

The Hadoop tools require information about the schema in JSON format so that the table names and primary key information can be used when materializing data from the staging tables into the base tables. This template generates that information in JSON format:

```
shell> ddlsync -user tungsten -url 'jdbc:mysql://tr-hadoop1:13306/test' -pass password \
-template ddl-mysql-hive-metadata.vm -db test

{
  "tables": [
    {
      "schema": "test",
      "name": "sales",
      "keys": [ "id" ],
      "columns": [
        { "name": "id", "type": "INT" },
        { "name": "salesman", "type": "STRING" },
        { "name": "planet", "type": "STRING" },
        { "name": "value", "type": "DOUBLE" }
      ]
    }
  ]
}
```

9.5.2.5. [ddl-mysql-oracle.vm](#)

When translating MySQL tables to Oracle compatible schema, the following datatypes are migrated to their closest Oracle equivalent:

MySQL Datatype	Oracle Datatype
INT	NUMBER(10, 0)
BIGINT	NUMBER(19, 0)
TINYINT	NUMBER(3, 0)
SMALLINT	NUMBER(5, 0)
MEDIUMINT	NUMBER(7, 0)
DECIMAL(x,y)	NUMBER(x, y)
FLOAT	FLOAT
CHAR(n)	CHAR(n)
VARCHAR(n)	VARCHAR2(n) (n < 2000), CLOB n > 2000)
DATE	DATE

MySQL Datatype	Oracle Datatype
DATETIME	DATE
TIMESTAMP	DATE
TEXT	CLOB
BLOB	BLOB
ENUM(...)	VARCHAR(255)
ENUM(...)	VARCHAR(4000)
BIT(1)	NUMBER(1)

The following additional transformations happen automatically:

- Table names are translated to uppercase.
- Column names are translated to uppercase.
- If a column name is a reserved word in Oracle, then the column name has an underscore character appended (for example, `TABLE` becomes `TABLE_`).

In addition to the above translations, errors will be raised for the following conditions:

- If the table name starts with a number.
- If the table name exceeds 30 characters in length.
- If the table name is a reserved word in Oracle.

Warnings will be raised for the following conditions:

- If the column or column name started with a number.
- If the column name exceeds 30 characters in length, the column name will be truncated.
- If the column name is a reserved word in Oracle.

9.5.2.6. `ddl-mysql-oracle-cdc.vm`

The `ddl-mysql-oracle-cdc.vm` template generates identical tables in Oracle, from their MySQL equivalent, but with additional columns for CDC capture. For example:

```
shell> ddlsync -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 ./ddlsync utility of Tungsten

url = jdbc:mysql://tr-hadoop1:13306/test
user = tungsten
dbName = test
*/

DROP TABLE test.sales;
CREATE TABLE test.sales
(
  id NUMBER(10, 0) NOT NULL,
  salesman CHAR,
  planet CHAR,
  value FLOAT,
  CDC_OP_TYPE VARCHAR(1), /* CDC column */
  CDC_TIMESTAMP TIMESTAMP, /* CDC column */
  CDC_SEQUENCE_NUMBER NUMBER PRIMARY KEY /* CDC column */
);
```

For information on the datatypes translated, see `ddl-mysql-oracle.vm`.

9.5.2.7. `ddl-mysql-redshift.vm`

9.5.2.8. `ddl-mysql-redshift-staging.vm`

9.5.2.9. `ddl-mysql-vertica.vm`

The `ddl-mysql-vertica.vm` template generates DDL for generating tables within an HP Vertica database from an existing MySQL database schema. For example:

```
shell> ddlsync -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 ./ddlsync utility of Tungsten

url = jdbc:mysql://tr-hadoop1:13306/test
user = tungsten
dbName = test
*/
CREATE SCHEMA test;

DROP TABLE test.sales;

CREATE TABLE test.sales
(
    id INT ,
    salesman CHAR(20) ,
    planet CHAR(20) ,
    value FLOAT ) ORDER BY id;
```

Because Vertica does not explicitly support primary keys, a default projection for the key order is created based on the primary key of the source table.

The templates translates different datatypes as follows:

MySQL Datatype	Vertica Datatype
DATETIME	DATETIME
TIMESTAMP	TIMESTAMP
DATE	DATE
TIME	TIME
TINYINT	TINYINT
SMALLINT	SMALLINT
MEDIUMINT	INT
INT	INT
BIGINT	INT
VARCHAR	VARCHAR
CHAR	CHAR
BINARY	BINARY
VARBINARY	VARBINARY
TEXT, TINYTEXT, MEDIUMTEXT, LONGTEXT	VARCHAR(65000)
BLOB, TINYBLOB, MEDIUMBLOB, LONGBLOB	VARBINARY(65000)
FLOAT	FLOAT
DOUBLE	DOUBLE PRECISION
ENUM	VARCHAR
SET	VARCHAR(4000)
BIT(1)	BOOLEAN
BIT	CHAR(64)

In addition, the following considerations should be taken into account:

- `DECIMAL` MySQL type is not supported.
- `TEXT` types in MySQL are converted to a `VARCHAR` in Vertica of the maximum supported size.
- `BLOB` types in MySQL are converted to a `VARBINARY` in Vertica of the maximum supported size.
- `SET` types in MySQL are converted to a `VARCHAR` in Vertica of 4000 characters, designed to work in tandem with the `settostring` filter.

- `ENUM` types in MySQL are converted to a `VARCHAR` in Vertica of the size of the longest `ENUM` value, designed to work in tandem with the `enumtostring` filter.

9.5.2.10. `ddl-mysql-vertica-staging.vm`

The `ddl-mysql-vertica-staging.vm` template generates DDL for HP Vertica staging tables. These include the full table definition, in addition to three columns used to define the staging data, including the operation code, sequence number and unique row ID. For example:

```
shell> ddlskan -user tungsten -url 'jdbc:mysql://tr-hadoop1:13306/test' -pass password \
-template ddl-mysql-vertica-staging.vm -db test
/*
SQL generated on Thu Sep 11 14:22:06 BST 2014 by ./ddlskan utility of Tungsten

url = jdbc:mysql://tr-hadoop1:13306/test
user = tungsten
dbName = test
*/
CREATE SCHEMA test;

DROP TABLE test.stage_xxx_sales;

CREATE TABLE test.stage_xxx_sales
(
  tungsten_opcode CHAR(1) ,
  tungsten_seqno INT ,
  tungsten_row_id INT ,
  id INT ,
  salesman CHAR(20) ,
  planet CHAR(20) ,
  value FLOAT ) ORDER BY tungsten_seqno, tungsten_row_id;
```

9.5.2.11. `ddl-oracle-mysql.vm`

The `ddl-oracle-mysql.vm` template generates the DDL required to create a schema within MySQL based on the existing Oracle schema. For example:

```
shell> ddlskan -service sales -template ddl-oracle-mysql.vm -db sales
/*
SQL generated on Thu Sep 11 04:29:08 PDT 2014 by ./ddlskan utility of Tungsten

url = jdbc:oracle:thin:@//tr-fromoracle1:1521/ORCL
user = SALES_PUB
dbName = sales
*/
/* ERROR: no tables found! Is database and tables option specified correctly? */

[tungsten@tr-fromoracle1 ~]$ ddlskan -service sales -template ddl-oracle-mysql.vm -db SALES
/*
SQL generated on Thu Sep 11 04:29:16 PDT 2014 by ./ddlskan utility of Tungsten

url = jdbc:oracle:thin:@//tr-fromoracle1:1521/ORCL
user = SALES_PUB
dbName = SALES
*/

DROP TABLE IF EXISTS sales.sample;
CREATE TABLE sales.sample
(
  id DECIMAL(38) /* NUMBER(38, ?) */ NOT NULL,
  msg CHAR(80),
  PRIMARY KEY (id)
) ENG
```

Columns are translated as follows:

Oracle Datatype	MySQL Datatype
DATE	DATETIME
NUMBER(0)	NUMERIC
NUMBER(n) where n < 3	TINYINT
NUMBER(n) where n < 5	SMALLINT
NUMBER(n) where n < 7	MEDIUMINT
NUMBER(n) where n < 10	INT
NUMBER(n) where n < 19	BIGINT

Oracle Datatype	MySQL Datatype
NUMBER	DECIMAL
FLOAT	FLOAT
VARCHAR	VARCHAR
LONG	LONGTEXT
BFILE	VARCHAR(1024)
CHAR	CHAR
CLOB	LONGTEXT
BLOB	LONGBLOB
LONG RAW	LONGBLOB
TIMESTAMP	TIMESTAMP
RAW	VARBINARY

The following additional transformations happen automatically:

- If a column name is a reserved word in MySQL, then the column name has an underscore character appended (for example, `TABLE` becomes `TABLE_`).

An error is raised in the following conditions:

- If the size of a `FLOAT` is larger than 53 points of precision.

9.5.2.12. `ddl-oracle-mysql-pk-only.vm`

The `ddl-oracle-mysql-pk-only.vm` template generates alter table statements to add the primary key, as determined from the Oracle primary key or index information. For example:

```
shell> ddlsync -service hadoop -template ddl-oracle-mysql-pk-only.vm -db HADOOP
/*
SQL generated on Thu Sep 11 06:17:28 PDT 2014 by ./ddlsync utility of Tungsten

url = jdbc:oracle:thin:@//tr-fromoracle1:1521/ORCL
user = HADOOP_PUB
dbName = HADOOP
*/

ALTER TABLE hadoop.sample ADD PRIMARY KEY (ID);
```

Note that it does not generate table DDL, only statements to alter existing tables with primary key information.

9.6. 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 9.6. `dsctl` Commands

Option	Description
<code>get</code>	Return the available position information
<code>help</code>	Print the help display
<code>reset</code>	Clear the datasource position information
<code>set</code>	Set the position

These must be used in conjunction with one of the following options to select the required datasources or service:

Table 9.7. `dsctl` Command-line Options

Option	Description
<code>-conf</code>	Path to the static services properties file

Option	Description
<code>-ds</code>	Name of the datasource
<code>-service</code>	Name of the replication service to get information from

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.

9.6.1. `dsctl get` Command

Returns the current datasource status and position, returning the information as a JSON string. The example below has been formatted for clarity:

```
shell> dsctl
[
  {
    "last_frag" : true,
    "applied_latency" : 1,
    "extract_timestamp" : "2015-01-21 21:46:57.0",
    "eventid" : "mysql-bin.000014:0000000000005645;-1",
    "source_id" : "tr-11",
    "epoch_number" : 22,
    "update_timestamp" : "2015-01-21 21:46:58.0",
    "task_id" : 0,
    "shard_id" : "tungsten_alpha",
    "segno" : 22,
    "fragno" : 0
  }
]
```

9.6.2. `dsctl set` Command

Table 9.8. `dsctl` Command-line Options

Option	Description
<code>-epoch</code>	Epoch Number
<code>-event-id</code>	Source Event ID
<code>-segno</code>	Sequence number
<code>-source-id</code>	Source ID

Sets the current replicator position. When using this option, the `-segno`, `-epoch`, `-event-id`, and `-source-id` options must be specified to set the corresponding values in the replicator.

For example:

```
shell> dsctl set -segno 22 -epoch 22 -event-id "mysql-bin.000014:0000000000005645;-1" -source-id tr-11
Service "alpha" datasource "global" position was set to: segno=22 epoch_number=22 eventid=mysql-bin.000014:0000000000005645;-1 so
```

9.6.3. `dsctl reset` Command

Clears the current replicator status and position information:

```
shell> dsctl reset
Service "alpha" datasource "global" catalog information cleared
```

9.6.4. `dsctl help` Command

Displays the current help text:

```
shell> dsctl help
Datasource Utility
Syntax: dsctl [conf|service] [-ds name] [operation]
Configuration (required if there's more than one service):
  -conf path      - Path to a static-<svc>.properties file
  OR
  -service name   - Name of a replication service to get datasource configuration from
Options:
  [-ds name]       - Name of the datasource (default: global)
Operations:
  get             - Return all available position information
  set -segno ###  - Set position (all four parameters required)
  -epoch ###
```

```
-event-id AAAAAAAA.#####:#####
-source-id AAA.AAA.AAA
reset          - Clear datasource position information
help           - Print this help display
```

9.7. `env.sh` Script

9.8. The load-reduce-check Tool

Important

The `load-reduce-check` tool is not part of the standard replicator distribution. The tool is part of the [continuent-tools-hadoop](#) repository, available from [Github](#).

The `load-reduce-check` tool provides a single command to perform the final steps to convert data loaded through the `Hadoop applier` into a final, Hive-compatible table providing a carbon copy of the data within Hive as extracted from the source database.

The four steps, each of which can be enabled or disabled individually are:

1. [Section 9.8.1, "Generating Staging DDL"](#)

Accesses the source database, reads the schema definition, and generates the necessary DDL for the staging tables within Hive. Tables are by default prefixed with `stage_xxx_`, and created in a Hive schema matching the source schema.

2. [Section 9.8.2, "Generating Live DDL"](#)

Accesses the source database, reads the schema definition, and generates the necessary DDL for the tables within Hive. Tables are created with an identical table and schema name to the source schema.

3. [Section 9.8.3, "Materializing a View"](#)

Execute a view materialization, where the data in any existing table, and the staging table are merged into the final table data. This step is identical to the process executed when running the `materialize` tool.

4. [Section 9.8.4, "Compare Loaded Data"](#)

Compares the data within the source and materialized tables and reports any differences.

The `load-reduce-check` tool

9.8.1. Generating Staging DDL

9.8.2. Generating Live DDL

9.8.3. Materializing a View

9.8.4. Compare Loaded Data

9.9. The `materialize` Command

Important

The `materialize` tool is not part of the standard replicator distribution. The tool is part of the [continuent-tools-hadoop](#) repository, available from [Github](#).

9.10. The `multi_trepctl` Command

The `multi_trepctl` command provides unified status and operation support across your Tungsten Replication installation across multiple hosts without the need to run the `trepctl` command across multiple hosts and/or services individually.

```
multi_trepctl
backups [ --by-service ] [ --fields appliedLastSeqNo appliedLatency host roles serviceName state ]
```

```
heartbeat [ --host, --hostself ]
list
masterof [ --output json|list|yaml ] [ --path, --paths ] [ --role, --roles ]
run [ --service, --serviceself ] [ --skip-headers ] [ --sort-by ]
```

The default operation, with no further command-line commands or arguments displays the status of all the hosts and services identified as related to the current host. In a typical single-service deployment, the command outputs the status of all services by determining the relationship between hosts connected to the default service:

```
shell> multi_trepctl
| host | serviceName | role | state | appliedLastSeqno | appliedLatency |
| tr-ms1 | alpha | master | ONLINE | 54 | 0.867 |
| tr-ms2 | alpha | slave | ONLINE | 54 | 1.945 |
| tr-ms3 | alpha | slave | ONLINE | 54 | 42.051 |
```

On a server with multiple services, information is output for each service and host:

```
shell> multi_trepctl
| host | servicename | role | state | appliedlastseqno | appliedLatency |
| east1 | east | master | ONLINE | 53 | 0.000 |
| east1 | west | slave | OFFLINE:ERROR | -1 | -1.000 |
| west1 | west | master | ONLINE | 294328 | 0.319 |
| west1 | east | slave | ONLINE | 53 | 119.834 |
| west2 | west | master | ONLINE | 231595 | 0.316 |
| west2 | east | slave | ONLINE | 53 | 181.128 |
| west3 | east | slave | ONLINE | 53 | 204.790 |
| west3 | west | slave | ONLINE | 231595 | 22.895 |
```

9.10.1. 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 9.9. `multi_trepctl` Command-line Options

Option	Description
<code>--by-service</code> [250]	Sort the output by the service name
<code>--fields</code> [250]	Fields to be output during summary
<code>--host</code> [251], <code>--hosts</code> [251]	Host or hosts on which to limit output
<code>--output</code> [251]	Specify the output format
<code>--paths</code> [251], <code>--path</code> [251]	Directory or directories to check when looking for tools
<code>--role</code> [251], <code>--roles</code> [251]	Role or roles on which to limit output
<code>--service</code> [251], <code>--services</code> [251]	Service or services on which to limit output
<code>--skip-headers</code> [252]	Skip the headers
<code>--sort-by</code> [252]	Sort by a specified field

Where:

- `--by-service` [250]

Order the output according to the service name and role within the service:

```
shell> multi_trepctl --by-service
| host | servicename | role | state | appliedlastseqno | appliedLatency |
| east1 | east | master | ONLINE | 64 | 59.380 |
| west1 | east | slave | ONLINE | 64 | 60.889 |
| west2 | east | slave | ONLINE | 64 | 60.970 |
| west3 | east | slave | ONLINE | 64 | 61.097 |
| west1 | west | master | ONLINE | 294328 | 0.319 |
| west2 | west | master | ONLINE | 231595 | 0.316 |
| east1 | west | slave | OFFLINE:ERROR | -1 | -1.000 |
| west3 | west | slave | ONLINE | 231595 | 22.895 |
```

- `--fields` [250]

Limited the output to the specified list of fields from the output of fields output by `trepctl`. For example, to limit the output to the host, role, and `appliedLatency`:

```
shell> multi_trepctl --fields=host,role,appliedlatency
| host | role | appliedlatency |
| tr-ms1 | master | 0.524 |
```

```
| tr-ms2 | slave | 0.000 |
| tr-ms3 | slave | -1.000 |
```

- [--host \[251\], --hosts \[251\]](#)

Limit the output to the host, or a comma-separated list of hosts specified. For example:

```
shell> multi_trepctl --hosts=tr-ms1,tr-ms3
| host | servicename | role | state | appliedlastseqno | appliedlatency |
| tr-ms1 | alpha | master | ONLINE | 2322 | 0.524 |
| tr-ms3 | alpha | slave | OFFLINE:ERROR | -1 | -1.000 |
```

- [--output \[251\]](#)

Specify the output format.

Table 9.10. `multi_trepctl`--`output` Option

Option	--output [251]	
Description	Specify the output format	
Value Type	string	
Default	info	
Valid Values	json	JSON format
	list	List format
	name	Name (simplified text) format
	tab	Tab-delimited format
	yaml	YAML format

For example, to output the current status in JSON format:

```
shell> multi_trepctl --output json
[
  {
    "appliedlastseqno": 2322,
    "appliedlatency": 0.524,
    "host": "tr-ms1",
    "role": "master",
    "servicename": "alpha",
    "state": "ONLINE"
  },
  {
    "appliedlastseqno": 2322,
    "appliedlatency": 0.0,
    "host": "tr-ms2",
    "role": "slave",
    "servicename": "alpha",
    "state": "ONLINE"
  },
  {
    "appliedlastseqno": -1,
    "appliedlatency": -1.0,
    "host": "tr-ms3",
    "role": "slave",
    "servicename": "alpha",
    "state": "OFFLINE:ERROR"
  }
]
```

- [--path \[251\], --paths \[251\]](#)

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:

- [--role \[251\], --roles \[251\]](#)

Limit the output to show only the specified role or comma-separated list of roles:

```
shell> multi_trepctl --roles=slave
| host | servicename | role | state | appliedlastseqno | appliedlatency |
| tr-ms2 | alpha | slave | ONLINE | 2322 | 0.000 |
| tr-ms3 | alpha | slave | OFFLINE:ERROR | -1 | -1.000 |
```

- [--service \[251\], --services \[251\]](#)

Limit the output to the specified service or comma-separated list of services:

```
shell> multi_trepctl --service=west
| host | servicename | role | state | appliedlastseqno | appliedlatency |
| east1 | east | master | ONLINE | 53 | 0.000 |
| west1 | east | slave | ONLINE | 53 | 119.834 |
| west2 | east | slave | ONLINE | 53 | 181.128 |
| west3 | east | slave | ONLINE | 53 | 204.790 |
```

- [--skip-headers \[252\]](#)

Prevents the generation of the headers when generating the list output format:

```
shell> multi_trepctl --skip-headers
| tr-ms1 | alpha | master | ONLINE | 2322 | 0.524 |
| tr-ms2 | alpha | slave | ONLINE | 2322 | 0.000 |
| tr-ms3 | alpha | slave | OFFLINE:ERROR | -1 | -1.000 |
```

- [--sort-by \[252\]](#)

Sort by the specified fieldname. For example, to sort the output by the latency:

```
shell> multi_trepctl --sort-by appliedlatency
| host | servicename | role | state | appliedlastseqno | appliedlatency |
| tr-ms3 | alpha | slave | OFFLINE:ERROR | -1 | -1.000 |
| tr-ms2 | alpha | slave | ONLINE | 2322 | 0.000 |
| tr-ms1 | alpha | master | ONLINE | 2322 | 0.524 |
```

9.10.2. 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 9.11. [multi_trepctl](#) Commands

Option	Description
backups	List all the backups available across all configured hosts and services
heartbeat	Inserts a heartbeat on all masters within the service
list	List the information about each service
masterof	List all the masters of configured hosts and services
run	Run the specified trepctl command on all hosts/services

In addition to the two primary commands, [multi_trepctl](#) can execute commands that would normally be applied to [trepctl](#), running them on each selected host, service or directory according to the options. The output format and expectation is controlled through the [list](#) and [run](#) commands.

For example:

```
shell> multi_trepctl status
```

Outputs the long form of the status information (as per [trepctl status](#)) for each identified host.

9.10.2.1. multi_trepctl backups Command

Lists the available backups across all replicators.

```
shell> multi_trepctl backups
| host | servicename | backup_date | prefix | agent |
| host1 | alpha | 2014-08-15 09:40:37 | store-0000000002 | mysqldump |
| host1 | alpha | 2014-08-15 09:36:57 | store-0000000001 | mysqldump |
| host2 | alpha | 2014-08-12 07:02:29 | store-0000000001 | mysqldump |
```

9.10.2.2. multi_trepctl heartbeat Command

Runs the [trepctl heartbeat](#) command on all hosts that are identified as masters.

```
shell> multi_trepctl heartbeat
host: host1
servicename: alpha
role: master
state: ONLINE
appliedlastseqno: 8
appliedlatency: 2.619
```

```
output:
```

9.10.2.3. multi_trepctl masterof Command

Lists which hosts are masters of others within the configured services.

```
shell> multi_trepctl masterof
| servicename | host | uri
| alpha       | host1 | thl://host1:2112/ |
```

9.10.2.4. multi_trepctl list Command

The `multi_trepctl list` mode is the default mode for `multi_trepctl` and outputs the current status across all hosts and services as a table:

```
shell> multi_trepctl
| host | servicename | role | state | appliedlastseqno | appliedlatency |
| host1 | firstrep | master | OFFLINE:ERROR | -1 | -1.000 |
| host2 | firstrep | slave | GOING-ONLINE:SYNCHRONIZING | 5271 | 4656.264 |
| host3 | firstrep | slave | OFFLINE:ERROR | -1 | -1.000 |
| host4 | firstrep | slave | OFFLINE:ERROR | -1 | -1.000 |
```

Or selected hosts and services if options are specified. For example, to get the status only for `host1` and `host2`:

```
shell> multi_trepctl --hosts=host1,host2
| host | servicename | role | state | appliedlastseqno | appliedlatency |
| host1 | firstrep | master | ONLINE | 5277 | 0.476 |
| host2 | firstrep | slave | ONLINE | 5277 | 0.000 |
```

The `multi_trepctl` command implies that the status or information is being output from each of the commands executed on the remote hosts and services.

9.10.2.5. multi_trepctl run Command

The `multi_trepctl run` command can be used where the output of the corresponding `trepctl` command cannot be formatted into a convenient list. For example, to execute a backup on every host within a deployment:

```
shell> multi_trepctl run backup
```

The same filters and host or service selection can also be made:

```
shell> multi_trepctl run backup --hosts=host1,host2,host3
host: host1
servicename: firstrep
output: |
  Backup completed successfully; URI=storage://file-system/store-000000005.properties
---
host: host2
servicename: firstrep
output: |
  Backup completed successfully; URI=storage://file-system/store-000000001.properties
---
host: host3
servicename: firstrep
output: |
  Backup completed successfully; URI=storage://file-system/store-000000001.properties
... 
```

Return from the command will only take place when remote commands on each host have completed and returned.

9.11. The replicator Command

The `replicator` is the wrapper script that handles the execution of the replicator service.

Table 9.12. `replicator` Commands

Option	Description
<code>condrestart</code> [254]	Restart only if already running
<code>console</code> [254]	Launch in the current console (instead of a daemon)
<code>dump</code> [254]	Request a Java thread dump (if replicator is running)
<code>install</code> [254]	Install the service to automatically start when the system boots
<code>remove</code> [254]	Remove the service from starting during boot
<code>restart</code> [254]	Stop replicator if already running and then start

Option	Description
<code>start [255]</code>	Start in the background as a daemon process
<code>status [255]</code>	Query the current status
<code>stop [255]</code>	Stop if running (whether as a daemon or in another console)

These commands and options are described below:

`condrestart [254]`

Table 9.13. **replicator** Commands Options for `condrestart [254]`

Option	Description
<code>offline</code>	Start in OFFLINE state

Restart the replicator, only if it is already running. This can be useful to use when changing configuration or performing database management within automated scripts, as the replicator will be only be restart if it was previously running.

For example, if the replicator is running, `replicator condrestart` operates as `replicator restart`:

```
shell> replicator condrestart
Stopping Tungsten Replicator Service...
Waiting for Tungsten Replicator Service to exit...
Stopped Tungsten Replicator Service.
Starting Tungsten Replicator Service...
Waiting for Tungsten Replicator Service.....
running: PID:26646
```

However, if not already running, the operation does nothing:

```
shell> replicator condrestart
Stopping Tungsten Replicator Service...
Tungsten Replicator Service was not running.
```

`console [254]`

Table 9.14. **replicator** Commands Options for `console [254]`

Option	Description
<code>offline</code>	Start in OFFLINE state

Launch in the current console (instead of a daemon)

`dump [254]`

Request a Java thread dump (if replicator is running)

`install [254]`

Installs the startup scripts for running the replicator at boot. For an alternative method of deploying these start-up scripts, see [deployall](#).

`remove [254]`

Removes the startup scripts for running the replicator at boot. For an alternative method of removing these start-up scripts, see [undeploy-all](#).

`restart [254]`

Table 9.15. **replicator** Commands Options for `restart [254]`

Option	Description
<code>offline</code>	Stop and restart in OFFLINE state

Warning

Restarting a running replicator temporarily stops and restarts replication.

Stops the replicator, if it is already running, and then restarts it:

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

```
Starting Tungsten Replicator Service...
Waiting for Tungsten Replicator Service.....
running: PID:26248
```

`start [255]`

Table 9.16. `replicator` Commands Options for `start [255]`

Option	Description
<code>offline</code>	Start in OFFLINE state

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

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

`status [255]`

Checks the execution status of the replicator:

```
shell> replicator status
Tungsten Replicator Service is running: PID:27015, Wrapper:STARTED, Java:STARTED
```

If the replicator is not running:

```
shell> replicator status
Tungsten Replicator Service is not running.
```

This only provides the execution state of the replicator, not the actual state of replication. To get detailed information on the status of replication use `trepctl status`.

`stop [255]`

Stops the replicator if it is already running:

```
shell> replicator stop
Stopping Tungsten Replicator Service...
Waiting for Tungsten Replicator Service to exit...
Stopped Tungsten Replicator Service.
```

9.12. 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 9.17, “`setupCDC.conf` Configuration Options”.

Table 9.17. `setupCDC.conf` Configuration Options

INI File Option	CmdLine Option	Description
<code>cdc_type [256]</code>	<code>cdc_type [256]</code>	The CDC type to be used to extract data, either synchronous (using triggers) or asynchronous (using log processing).
<code>delete_publisher</code>	<code>delete_publisher</code>	Whether the publisher user should be deleted.
<code>delete_subscriber</code>	<code>delete_subscriber</code>	Whether the subscriber user should be deleted.
<code>enable_archivelog</code>	<code>enable_archivelog</code>	If set to true, the Oracle instance will be configured to enable archive logging (required by Oracle CDC), and then the Oracle instance will be restarted.
<code>pub_password</code>	<code>pub_password</code>	The publisher password that will be created for the CDC service.
<code>pub_tablespace</code>	<code>pub_tablespace</code>	Use the specified tablespace for CDC publishing instead of system tablespace
<code>pub_user</code>	<code>pub_user</code>	The publisher user that will be created for this CDC service.
<code>service</code>	<code>service</code>	The service name of the Tungsten Replicator service that will be created.
<code>source_user</code>	<code>source_user</code>	The source schema user with rights to access the database.
<code>specific_path</code>	<code>specific_path</code>	The path where the tungsten.tables file is located; the file must be in a shared location accessible by Tungsten Replicator.

INI File Option	CmdLine Option	Description
<code>specific_tables</code>	<code>specific_tables</code>	If enabled, extract only the tables defined within a tungsten.tables file.
<code>sys_pass</code>	<code>sys_pass</code>	The system password to connect to Oracle as SYSDBA.
<code>sys_user [258]</code>	<code>sys_user [258]</code>	The system user to connect to Oracle as SYSDBA.
<code>tungsten_pwd</code>	<code>tungsten_pwd</code>	The password for the subscriber user.
<code>tungsten_user</code>	<code>tungsten_user</code>	The subscriber (Tungsten user) that will subscribe to the changes and read the information from the CDC tables.

Where:

`cdc_type [256]`

Option	<code>cdc_type [256]</code>	
Config File Options	<code>cdc_type [256]</code>	
Description	The CDC type to be used to extract data, either synchronous (using triggers) or asynchronous (using log processing).	
Value Type	string	
Valid Values	CDCASYNC	Enable asynchronous capture
	CDCSYNC	Enable synchronous capture

The CDC type to be used to extract data, either synchronous (using triggers) or asynchronous (using log processing).

`delete_publisher`

Option	<code>delete_publisher</code>	
Config File Options	<code>delete_publisher</code>	
Description	Whether the publisher user should be deleted.	
Value Type	string	
Valid Values	0	Do not delete the user before creation
	1	Delete the user before creation

Whether the publisher user should be deleted.

`delete_subscriber`

Option	<code>delete_subscriber</code>	
Config File Options	<code>delete_subscriber</code>	
Description	Whether the subscriber user should be deleted.	
Value Type	string	
Valid Values	0	Do not delete the user before creation
	1	Delete the user before creation

Whether the subscriber user should be deleted.

`enable_archivelog`

Option	<code>enable_archivelog</code>	
Config File Options	<code>enable_archivelog</code>	
Description	If set to true, the Oracle instance will be configured to enable archive logging (required by Oracle CDC), and then the Oracle instance will be restarted.	
Value Type	boolean	
Valid Values	0	Do not automatically enable the archive log during setup
	1	Automatically enable the archive log during setup

If set to true, the Oracle instance will be configured to enable archive logging (required by Oracle CDC), and then the Oracle instance will be restarted.

`pub_password`

Option	<code>pub_password</code>
Config File Options	<code>pub_password</code>
Description	The publisher password that will be created for the CDC service.
Value Type	string

The publisher password that will be created for the CDC service.

`pub_tablespace`

Option	<code>pub_tablespace</code>
Config File Options	<code>pub_tablespace</code>
Description	Use the specified tablespace for CDC publishing instead of system tablespace
Value Type	string

By default, the system tablespace is used for holding the publisher tables. Using the system tablespace should only be used during testing, as the tablespace is typically not large enough to hold the required change data. If set to 1, use the created tablespace (matching the value of `pub_user`) which is assumed to be large enough to hold the change information.

`pub_user`

Option	<code>pub_user</code>
Config File Options	<code>pub_user</code>
Description	The publisher user that will be created for this CDC service.
Value Type	string

The publisher user that will be created for this CDC service.

`service`

Option	<code>service</code>
Config File Options	<code>service</code>
Description	The service name of the Tungsten Replicator service that will be created.
Value Type	string

The service name of the Tungsten Replicator service that will be created.

`source_user`

Option	<code>source_user</code>
Config File Options	<code>source_user</code>
Description	The source schema user with rights to access the database.
Value Type	string

The source schema user with rights to access the database.

`specific_path`

Option	<code>specific_path</code>
Config File Options	<code>specific_path</code>
Description	The path where the tungsten.tables file is located; the file must be in a shared location accessible by Tungsten Replicator.
Value Type	string

The path where the tungsten.tables file is located; the file must be in a shared location accessible by Tungsten Replicator.

`specific_tables`

Option	specific_tables	
Config File Options	specific_tables	
Description	If enabled, extract only the tables defined within a tungsten.tables file.	
Value Type	string	
Valid Values	0	Extract all tables
	1	Use a tables file to select tables

If enabled, extract only the tables defined within a tungsten.tables file.

[sys_pass](#)

Option	sys_pass
Config File Options	sys_pass
Description	The system password to connect to Oracle as SYSDBA.
Value Type	string

The system password to connect to Oracle as SYSDBA.

[sys_user \[258\]](#)

Option	sys_user [258]
Config File Options	sys_user [258]
Description	The system user to connect to Oracle as SYSDBA.
Value Type	string

The system user to connect to Oracle as SYSDBA.

[tungsten_pwd](#)

Option	tungsten_pwd
Config File Options	tungsten_pwd
Description	The password for the subscriber user.
Value Type	string

The password for the subscriber user.

[tungsten_user](#)

Option	tungsten_user
Config File Options	tungsten_user
Description	The subscriber (Tungsten user) that will subscribe to the changes and read the information from the CDC tables.
Value Type	string

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
```

9.13. The startall Command

The [startall](#) will start all configured services within the configured directory:

```
Starting Tungsten Replicator Service...
Waiting for Tungsten Replicator Service.....
running: PID:29842
```

If a service is already running, then a notification of the current state will be provided:

```
Starting Tungsten Replicator Service...
Tungsten Replicator Service is already running.
```

Note that if any service is not running, and a suitable `PID` is found, the file will be deleted and the services started, for example:

```
Removed stale pid file: /opt/continuent/releases/tungsten-replicator-5.0.1-136_pid25898/tungsten-connector/bin/../var/tconnector
```

9.14. The stopall Command

The `stopall` command stops all running services if they are already running:

```
shell> stopall
Stopping Tungsten Replicator Service...
Waiting for Tungsten Replicator Service to exit...
Stopped Tungsten Replicator Service.
```

9.15. The thl Command

The `thl` command provides an interface to the THL data, including the ability to view the list of available files, details of the enclosed event information, and the ability to purge THL files to reclaim space on disk beyond the configured log retention policy.

The command supports two command-line options that are applicable to all operations, as shown in [Table 9.18, “thl Options”](#).

Table 9.18. `thl` Options

Option	Description
<code>-conf path</code>	Path to the configuration file containing the required replicator service configuration
<code>-service servicename</code>	Name of the service to be used when looking for THL information

For example, to execute a command on a specific service:

```
shell> thl index -service firstrep
```

Individual operations are selected by use of a specific command to the `thl` command. Supported commands are:

- `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.

9.15.1. thl list Command

The `list` command to the `thl` command outputs a list of the sequence number information from the THL. By default, the entire THL as stored on disk is output. Command-line options enable you to select individual sequence numbers, sequence number ranges, or all the sequence information from a single file.

```
thl list
[-seqno #]
[-low #] | [-high #]
[-file filename] [-no-checksum] [-sql] [-charset] [-headers] [-json] [-specs] [-charset]
```

There are three selection mechanisms:

- `-seqno # [259]`

Output the THL sequence for the specific sequence number. When reviewing or searching for a specific sequence number, for example when the application of a sequence on a slave has failed, the replication data for that sequence number can be individually viewed. For example:

```
shell> thl list -seqno 15
SEQ# = 15 / FRAG# = 0 (last frag)
- TIME = 2013-05-02 11:37:00.0
- EPOCH# = 7
- EVENTID = mysql-bin.000004:0000000000003345:0
- SOURCEID = host1
- METADATA = [mysql server_id=1687011;unsafe_for_block_commit;dbms_type=mysql;»
```

```

    service=firstrep;shard=cheffy]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [##charset = UTF-8, autocommit = 1, sql_auto_is_null = 0, foreign_key_checks = 0, »
    unique_checks = 0, sql_mode = 'NO_AUTO_VALUE_ON_ZERO', character_set_client = 33, »
    collation_connection = 33, collation_server = 8]
- SCHEMA = cheffy
- SQL(0) = CREATE TABLE `access_log` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `userid` int(10) unsigned DEFAULT NULL,
  `datetime` int(10) unsigned NOT NULL DEFAULT '0',
...

```

If the sequence number selected contains multiple fragments, each fragment will be output. Depending on the content of the sequence number information, the information can be output containing only the header/metadata information or only the table data (row or SQL) that was contained within the fragment. See [-headers](#) and [-sql](#) for more information.

Note

Unsigned integers are displayed and stored in the THL as their negative equivalents, and translated to the correct unsigned type when the data is applied to the target database.

- [-low #](#) and/or [-high #](#)

Specify the start ([-low](#)) or end ([-high](#)) of the range of sequence numbers to be output. If only [-low](#) is specified, then all sequence numbers from that number to the end of the THL are output. If [-high](#) is specified, all sequence numbers from the start of the available log file to the specified sequence number are output. If both numbers are specified, output all the sequence numbers within the specified range. For example:

```
shell> thl list -low 320
```

Will output all the sequence number fragments from number 320.

```
shell> thl list -high 540
```

Will output all the sequence number fragments up to and including 540.

```
shell> thl list -low 320 -high 540
```

Will output all the sequence number fragments from number 320 up to, and including, sequence number 540.

- [-file filename](#) [260]

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](#) [260]

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](#) [260]

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](#) [260]

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.000005:000000579721045;0 host3
```

```

2047 1412 1 true 2013-05-03 20:58:14.0 mysql-bin.000005:0000000579721116;0 host3
2048 1412 0 false 2013-05-03 20:58:14.0 mysql-bin.000005:0000000580759206;0 host3
2048 1412 1 true 2013-05-03 20:58:14.0 mysql-bin.000005:0000000580759277;0 host3
2049 1412 0 false 2013-05-03 20:58:16.0 mysql-bin.000005:0000000581791468;0 host3
2049 1412 1 true 2013-05-03 20:58:16.0 mysql-bin.000005:0000000581791539;0 host3
2050 1412 0 false 2013-05-03 20:58:18.0 mysql-bin.000005:0000000582812644;0 host3

```

The format of the fields output is:

Sequence No	Epoch	Fragment	Last	Fragment	Date/Time	EventID	SourceID	Comments
-------------	-------	----------	------	----------	-----------	---------	----------	----------

For more information on the fields displayed, see [Section 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.000004:000000244490614;0"
  },
  {
    "lastFrag" : true,
    "epoch" : 7,
    "seqno" : 320,
    "time" : "2013-05-02 11:41:19.0",
    "frag" : 1,
    "comments" : "",
    "sourceId" : "host1",
    "eventId" : "mysql-bin.000004:000000244490685;0"
  }
]
```

For more information on the fields displayed, see [THL SEQNO \[459\]](#).

- [-specs](#)

Shows the column specifications, such as identified type, length, and additional settings, when viewing events within row-based replication. This can be helpful when examining THL data in heterogeneous replication deployments.

For example:

```

shell> thl list -low 5282 -specs
SEQ# = 5282 / FRAG# = 0 (last frag)
- TIME = 2014-01-30 05:46:26.0
- EPOCH# = 5278
- EVENTID = mysql-bin.000017:0000000000001117;0
- SOURCEID = host1
- METADATA = [mysql_server_id=1687011;dbms_type=mysql;is_metadata=true;]
  services:firstrep;shard=tungsten_firstrep;heartbeat=MASTER_ONLINE
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- SQL(0) =
- ACTION = UPDATE
- SCHEMA = tungsten_firstrep
- TABLE = heartbeat
- ROW# = 0
- COL(index=1 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=null) = 1
- COL(index=2 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=null) = 1416
- COL(index=3 name= type=12 [VARCHAR] length=0 unsigned=false blob=false desc=null) = [B@65b60280
- COL(index=4 name= type=93 [TIMESTAMP] length=0 unsigned=false blob=false desc=null) = 2014-01-30 05:46:26.0
- COL(index=5 name= type=93 [TIMESTAMP] length=0 unsigned=false blob=false desc=null) = 2013-05-03 12:05:47.0
- COL(index=6 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=null) = 1015
- COL(index=7 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=null) = 0
- COL(index=8 name= type=12 [VARCHAR] length=0 unsigned=false blob=false desc=null) = [B@105e55ab
- KEY(index=1 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=null) = 1

```

When identifying the different data types, the following effects should be noted:

- [CHAR](#) and [VARCHAR](#) are identified as type 12, [VARCHAR](#)
- [SET](#) is identified as an [INTEGER](#)

- When the value is either `NULL` or `0` (Zero), date and time fields are shown as type 0, `NULL`
- `ENUM` is identified as an `OTHER`
- `BLOB` and `TEXT` are identified as type 2004, `BLOB`
- `-timezone`

Specify the timezone to use when display date or time values. When not specified, times are displayed using UTC.

9.15.2. 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` [260] ignores the checksum information on events in the event that the checksum is corrupt.

9.15.3. 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)
LogIndexEntry thl.data.0000000382(5902:5921)
LogIndexEntry thl.data.0000000383(5922:5941)
LogIndexEntry thl.data.0000000384(5942:5961)
LogIndexEntry thl.data.0000000385(5962:5981)
LogIndexEntry thl.data.0000000386(5982:6001)
LogIndexEntry thl.data.0000000387(6002:6021)
LogIndexEntry thl.data.0000000388(6022:6041)
LogIndexEntry thl.data.0000000389(6042:6061)
```

```
LogIndexEntry thl.data.0000000390(6062:6081)
LogIndexEntry thl.data.0000000391(6082:6101)
LogIndexEntry thl.data.0000000392(6102:6121)
LogIndexEntry thl.data.0000000393(6122:6141)
LogIndexEntry thl.data.0000000394(6142:6161)
LogIndexEntry thl.data.0000000395(6162:6181)
LogIndexEntry thl.data.0000000396(6182:6201)
LogIndexEntry thl.data.0000000397(6202:6221)
LogIndexEntry thl.data.0000000398(6222:6241)
LogIndexEntry thl.data.0000000399(6242:6261)
LogIndexEntry thl.data.0000000400(6262:6266)
```

The above shows a range of THL sequences from 5802 to 6266.

To delete all of the THL from the start of the list, sequence no 5802, to 6021 (inclusive), use the `-high` to specify the highest number to be removed (6021):

```
shell> thl purge -high 6021
WARNING: The purge command will break replication if you delete all
events or delete events that have not reached all slaves.
Are you sure you wish to delete these events [y/N]?
y
Deleting events where SEQ# <=6021
2017-02-10 16:31:36,235 [- main] INFO thl.THLManagerCtrl Transactions deleted
```

Running a `thl index`, sequence numbers from 6022 to 6266 are still available:

```
shell> thl index
LogIndexEntry thl.data.0000000388(6022:6041)
LogIndexEntry thl.data.0000000389(6042:6061)
LogIndexEntry thl.data.0000000390(6062:6081)
LogIndexEntry thl.data.0000000391(6082:6101)
LogIndexEntry thl.data.0000000392(6102:6121)
LogIndexEntry thl.data.0000000393(6122:6141)
LogIndexEntry thl.data.0000000394(6142:6161)
LogIndexEntry thl.data.0000000395(6162:6181)
LogIndexEntry thl.data.0000000396(6182:6201)
LogIndexEntry thl.data.0000000397(6202:6221)
LogIndexEntry thl.data.0000000398(6222:6241)
LogIndexEntry thl.data.0000000399(6242:6261)
LogIndexEntry thl.data.0000000400(6262:6266)
```

To delete the last two THL files, specify the sequence number at the start of the file, 6242 to the `-low` to specify the sequence number:

```
shell> thl purge -low 6242 -y
WARNING: The purge command will break replication if you delete all
events or delete events that have not reached all slaves.
Deleting events where SEQ# >= 6242
2017-02-10 16:40:42,463 [- main] INFO thl.THLManagerCtrl Transactions deleted
```

A `thl index` shows the sequence as removed:

```
shell> thl index
LogIndexEntry thl.data.0000000388(6022:6041)
LogIndexEntry thl.data.0000000389(6042:6061)
LogIndexEntry thl.data.0000000390(6062:6081)
LogIndexEntry thl.data.0000000391(6082:6101)
LogIndexEntry thl.data.0000000392(6102:6121)
LogIndexEntry thl.data.0000000393(6122:6141)
LogIndexEntry thl.data.0000000394(6142:6161)
LogIndexEntry thl.data.0000000395(6162:6181)
LogIndexEntry thl.data.0000000396(6182:6201)
LogIndexEntry thl.data.0000000397(6202:6221)
LogIndexEntry thl.data.0000000398(6222:6241)
```

The confirmation message can be bypassed by using the `-y` option, which implies that the operation should proceed without further confirmation.

The optional argument `-no-checksum` [260] ignores the checksum information on events in the event that the checksum is corrupt.

When purging, the THL files must be writeable; the replicator must either be offline or stopped when the purge operation is completed.

A `purge` operation may fail for the following reasons:

- Fatal error: The disk log is not writable and cannot be purged.

The replicator is currently running and not in the `OFFLINE` [207] state. Use `trepctl offline` to release the write lock on 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.

9.15.4. thl info Command

The [info](#) command to [thl](#) command provides the current information about the THL, including the identified log directory, sequence number range, and the number of individual events with the available span. The lowest and highest THL file and sizes are also given. For example:

```
shell> thl info
log directory = /opt/continuent/thl/alpha/
log files = 41
logs size = 193.53 MB
min seq# = 0
max seq# = 228
events = 228
oldest file = thl.data.0000000001 (95.48 MB, 2013-12-18 11:53:00)
newest file = thl.data.0000000041 (0.98 MB, 2013-12-18 12:34:32)
```

The optional argument [-no-checksum](#) [260] ignores the checksum information on events in the event that the checksum is corrupt.

9.15.5. thl help Command

The [help](#) command to the [thl](#) command outputs the current help message text.

9.16. 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 9.16.1, “trepctl Options”](#).

```
trepctl
backup [-backup agent] [-limit s] [-storage agent]
capabilities
check
clear
clients [-json]
flush [-limit s]
heartbeat [-name] [-host name]
kill [-y]
load
offline
offline-deferred [-at-event event] [-at-heartbeat [heartbeat]] [-at-seqno seqno] [-at-time YYYY-MM-DD_hh:mm:ss] [-immediate]
online [-base-seqno x] [-force] [-from-event event] [-no-checksum] [-provision [SCN]] [-skip-seqno seqdef] [-until-event event] [-until-heartbeat [name]] [-until-seqno seqno] [-until-time YYYY-MM-DD_hh:mm:ss] [-port number]
properties [-filter name] [-values]
purge [-limit s]
reset [-all] [-db] [-relay] [-thl] [-y]
restore [-retry N] [-service name]
services [-full] [-json]
setrole [-rolemasterrelayslave] [-uri]
shard [-delete shard] [-insert shard] [-list] [-update shard]
status [-json] [-namechannel-assignmentsserviceshardsstagesstoretaskswatches]
```

```
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 9.16.2, “trepctl Global Commands”](#). Information on individual commands can be found in [Section 9.16.3, “trepctl Service Commands”](#).

9.16.1. trepctl Options

Table 9.19. **trepctl** Command-line Options

Option	Description
<code>-host name</code>	Host name of the replicator
<code>-port number</code>	Port number of the replicator
<code>-retry N</code>	Number of times to retry the connection
<code>-service name</code>	Name of the replicator service
<code>-verbose</code>	Enable verbose messages for operations

Global command-line options enable you to select specific hosts and services. If available, **trepctl** will read the active configuration to determine the host, service, and port information. If this is unavailable or inaccessible, the following rules are used to determine which host or service to operate upon:

- If no host is specified, then **trepctl** defaults to the host on which the command is being executed.
- If no service is specified:
 - If only one service has been configured, then **trepctl** defaults to showing information for the configured service.
 - If multiple services are configured, then **trepctl** returns an error, and requests a specific service be selected.

To use the global options:

- `-host`

Specify the host for the operation. The replicator service must be running on the remote host for this operation to work.

- `-port`

Specify the base TCP/IP port used for administration. The default is port 10000; port 10001 is also used. When using different ports, `port` and `port+1` is used, i.e. if port 4996 is specified, then port 4997 will be used as well. When multiple replicators are installed on the same host, different numbers may be used.

- `-service`

The servicename to be used for the requested status or control operation. When multiple services have been configured, the servicename must be specified.

```
shell> trepctl status
Processing status command...
Operation failed: You must specify a service name with the -service flag
```

- `-verbose`

Turns on verbose reporting of the individual operations. This includes connectivity to the replicator service and individual operation steps. This can be useful when diagnosing an issue and identifying the location of a particular problem, such as timeouts when access a remote replicator.

- `-retry`

Retry the request operation the specified number of times. The default is 10.

9.16.2. 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 9.20. `trepctl` Replicator Wide Commands

Option	Description
<code>kill</code>	Shutdown the replication services immediately
<code>services</code>	List the configured replicator services
<code>version</code>	Show the replicator version number and build

These commands can be executed on the current or a specified host. Because these commands operate for replicators irrespective of the service configuration, selecting or specifying a service is not required.

9.16.2.1. `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> repctl 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> repctl kill -y
Sending Kill command to replicator
Replicator appears to be stopped
```

9.16.2.2. `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> repctl services
Processing services command...
NAME          VALUE
----          -----
appliedLastSeqno: 2541
appliedLatency : 0.48
role          : master
serviceName   : alpha
serviceType   : local
started       : true
state         : ONLINE
Finished services command...
```

For more information on the fields displayed, see [Section E.2, “Generated Field Reference”](#).

For a replicator with multiple services, the information is output for each configured service:

```
shell> repctl services
Processing services command...
NAME          VALUE
----          -----
appliedLastSeqno: 44
appliedLatency : 0.692
role          : master
serviceName   : alpha
serviceType   : local
started       : true
state         : ONLINE
NAME          VALUE
----          -----
appliedLastSeqno: 40
appliedLatency : 0.57
role          : slave
serviceName   : beta
serviceType   : remote
started       : true
state         : ONLINE
NAME          VALUE
----          -----
```

```
appliedLastSeqno: 41
appliedLatency : 0.06
role           : slave
serviceName    : gamma
serviceType   : remote
started        : true
state          : ONLINE
Finished services command...
```

The information can be reported in JSON format by using the `-json` option to the command:

```
shell> trepctl services -json
[
  {
    "serviceType" : "local",
    "appliedLatency" : "0.48",
    "serviceName" : "alpha",
    "appliedLastSeqno" : "2541",
    "started" : "true",
    "role" : "master",
    "state" : "ONLINE"
  }
]
```

The information is output as an array of objects, one object for each service identified.

If the `-full` option is added, the JSON output includes full details of the service, similar to that output by the `trepctl status` command, but for each configured service:

```
shell> trepctl services -json -full
[
  {
    "masterConnectUri" : "",
    "rmiPort" : "10000",
    "clusterName" : "default",
    "currentTimeMillis" : "1370256230198",
    "state" : "ONLINE",
    "maximumStoredSeqNo" : "2541",
    "minimumStoredSeqNo" : "0",
    "pendingErrorCode" : "NONE",
    "masterListenUri" : "thl://host1:2112/",
    "pendingErrorSeqno" : "-1",
    "pipelineSource" : "jdbc:mysql:thin://host1:3306",
    "serviceName" : "alpha",
    "pendingErrorEventId" : "NONE",
    "appliedLatency" : "0.48",
    "transitioningTo" : "",
    "relativeLatency" : "245804.198",
    "role" : "master",
    "siteName" : "default",
    "pendingError" : "NONE",
    "uptimeSeconds" : "246023.627",
    "latestEpochNumber" : "2537",
    "extensions" : "",
    "dataServerHost" : "host1",
    "resourcePrecedence" : "99",
    "pendingExceptionMessage" : "NONE",
    "simpleServiceName" : "alpha",
    "sourceId" : "host1",
    "offlineRequests" : "NONE",
    "channels" : "1",
    "version" : "Tungsten Replicator 5.0.1 build 136",
    "seqnoType" : "java.lang.Long",
    "serviceType" : "local",
    "currentEventId" : "mysql-bin.000007:0000000000001033",
    "appliedLastEventId" : "mysql-bin.000007:0000000000001033;0",
    "timeInStateSeconds" : "245803.753",
    "appliedLastSeqno" : "2541",
    "started" : "true"
  }
]
```

For more information on the fields displayed, see [Section E.2, “Generated Field Reference”](#).

9.16.2.3. `trepctl version` Command

The `trepctl version` command outputs the version number of the specified replicator service.

```
trepctl version
```

```
shell> trepctl version
Tungsten Replicator 5.0.1 build 136
```

The system can also be used to obtain remote version:

```
shell> trepctl -host host2 version
Tungsten Replicator 5.0.1 build 136
```

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.

9.16.3. 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 9.21. `trepctl` Service Commands

Option	Description
<code>backup</code>	Backup database
<code>capabilities</code>	List the configured replicator capabilities
<code>check</code>	Generate consistency check
<code>clear</code>	Clear one or all dynamic variables
<code>clients</code>	List clients connected to this replicator
<code>flush</code>	Synchronize transaction history log to database
<code>heartbeat</code>	Insert a heartbeat event with optional name
<code>load</code>	Load the replication service
<code>offline</code>	Set replicator to OFFLINE state
<code>offline-deferred</code>	Set replicator OFFLINE at a future point in the replication stream
<code>online</code>	Set Replicator to ONLINE with start and stop points
<code>properties</code>	Display a list of all internal properties
<code>purge</code>	Purge non-Tungsten logins on database
<code>reset</code>	Deletes the replicator service
<code>restore</code>	Restore database on specified host
<code>setrole</code>	Set replicator role
<code>shard</code>	List, add, update, and delete shards
<code>status</code>	Print replicator status information
<code>unload</code>	Unload the replication service
<code>wait</code>	Wait for the replicator to reach a specific state, time or applied sequence number

The following sections detail each command individually, with specific options, operations and information.

9.16.3.1. trepctl backup Command

Tungsten Replication 5.0.0. This feature has been deprecated in Tungsten Replication 5.0.0 and will be removed in a future release.

The `trepctl backup` command performs a backup of the corresponding database for the selected service.

```
trepctl backup [ -backup agent ] [ -limit s ] [ -storage agent ]
```

Where:

Table 9.22. `trepctl backup` Command Options

Option	Description
<code>-backup agent</code> [269]	Select the backup agent
<code>-limit s</code> [269]	The period to wait before returning after the backup request

Option	Description
<code>-storage agent [269]</code>	Select the storage agent

Without specifying any options, the backup uses the default configured backup and storage system, and will wait indefinitely until the backup process has been completed:

```
shell> trepctl backup
Backup completed successfully; URI=storage://file-system/store-0000000002.properties
```

The return information gives the URI of the backup properties file. This information can be used when performing a restore operation as the source of the backup. See [Section 9.16.3.15, “trepctl restore Command”](#). Different backup solutions may require that the replicator be placed into the [OFFLINE \[207\]](#) 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. [mysql-dump.log](#)).

If multiple backup agents have been configured, the backup agent can be selected on the command-line:

```
shell> trepctl backup -backup mysqldump
```

If multiple storage agents have been configured, the storage agent can be selected using the `-storage [269]` option:

```
shell> trepctl backup -storage file
```

A backup will always be attempted, but the timeout to wait for the backup to be started during the command-line session can be specified using the `-limit [269]` 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 [269]` option specifies how long `trepctl` should wait before returning.

For example, to wait five seconds before returning:

```
shell> trepctl -service alpha backup -limit 5
Backup is pending; check log for status
```

The backup request has been received, but not completed within the allocated time limit. The command will return. Checking the logs shows the timeout:

```
... management.OpenReplicatorManager Backup request timed out: seconds=5
```

Followed by the successful completion of the backup, indicated by the URI provided in the log showing where the backup file has been stored.

```
... backup.BackupTask Storing backup result...
... backup.FileSystemStorageAgent Allocated backup location: »
uri =storage://file-system/store-0000000003.properties
... backup.FileSystemStorageAgent Stored backup storage file: »
file=/opt/continuent/backups/store-0000000003-mysqldump_2013-07-15_18-14_11.sql.gz length=0
... backup.FileSystemStorageAgent Stored backup storage properties: »
file=/opt/continuent/backups/store-0000000003.properties length=314
... backup.BackupTask Backup completed normally: »
uri=storage://file-system/store-0000000003.properties
```

The URI can be used during a restore.

9.16.3.2. trepctl capabilities Command

The `capabilities` command outputs a list of the supported capabilities for this replicator instance.

```
trepctl capabilities
```

The information output will depend on the configuration and current role of the replicator service. Different services on the same host may have different capabilities. For example:

```
shell> trepctl capabilities
Replicator Capabilities
  Roles:           [master, slave]
  Replication Model: push
  Consistency Check: true
  Heartbeat:        true
  Flush:           true
```

The fields output are as follows:

- `Roles`

Indicates whether the replicator can be a [master](#) (in [Tungsten Clustering for MySQL 5.0 Manual]) or [slave](#) (in [Tungsten Clustering for MySQL 5.0 Manual]), 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 9.16.3.3, “treptctl check Command”](#).

- [Heartbeat](#)

Indicates whether the heartbeat service is supported. For more information see [Section 9.16.3.7, “treptctl heartbeat Command”](#).

- [Flush](#)

Indicates whether the `treptctl flush` operation is supported.

9.16.3.3. treptctl 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.

9.16.3.4. treptctl clear Command

The `treptctl clear` command deletes any dynamic properties configured within the replicator service.

```
treptctl clear
```

Dynamic properties include the current active role for the service. The dynamic information is stored internally within the replicator, and also stored within a properties file on disk so that the replicator can be restarted.

For example, the replicator role may be temporarily changed to receive information from a different host or to act as a master in place of a slave. The replicator can be returned to the initial configuration for the service by clearing this dynamic property:

```
shell> treptctl clear
```

9.16.3.5. treptctl clients Command

Outputs a list of the that have been connected to the master service since it went online. If a slave service goes offline or is stopped, it will still be reported by this command.

```
treptctl clients [-json]
```

Where:

Table 9.23. `treptctl clients` Command Options

Option	Description
<code>-json</code> [270]	Output the information as JSON

The command outputs the list of clients and the management port on which they can be reached:

```
shell> treptctl 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` [270] option:

```
shell> treptctl 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.

9.16.3.6. `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
```

9.16.3.7. `trepctl heartbeat` Command

Inserts a heartbeat into the replication stream, which can be used to identify replication points.

```
trepctl heartbeat [-name]
```

The heartbeat system is a way of inserting an identifiable event into the THL that is independent of the data being replicated. This can be useful when performing different operations on the data where specific checkpoints must be identified.

To insert a standard heartbeat:

```
shell> trepctl heartbeat
```

When performing specific operations, the heartbeat can be given an name:

```
shell> trepctl heartbeat -name dataload
```

Heartbeats insert a transaction into the THL using the transaction metadata and can be used to identify whether replication is operating between replicator hosts by checking that the sequence number has been replicated to the 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         : 0.704  
...  
offlineRequests        : Offline at heartbeat event: dataload
```

2. On the master:

```
master shell> mysql newdb < newdb.load
```

3. Once the data load has completed, insert the heartbeat on the master:

```
master shell> trepctl heartbeat -name dataload
```

The heartbeat will appear in the transaction history log after the data has been loaded and will identify the end of the load.

4. When the heartbeat is received, the slave will go into the offline state. Now a backup can be created with all of the loaded data replicated from the master. Because the slave is in the offline state, no further data or changes will be recorded on the slave

This method of identifying specific events and points within the transaction history log can be used for a variety of different purposes where the point within the replication stream without relying on the arbitrary event or sequence number.

Internal Implementation

Internally, the heartbeat system operates through a tag added to the metadata of the THL entry and through a dedicated `heartbeat` table within the schema created for the replicator service. The table contains the sequence number, event ID, timestamp and heartbeat name. The heartbeat information is written into a special record within the transaction history log. A sample THL entry can be seen in the output below:

```
SEQ# = 3629 / FRAG# = 0 (last frag)  
- TIME = 2013-07-19 12:14:57.0  
- EPOCH# = 3614  
- EVENTID = mysql-bin.000009:0000000000008681:0  
- SOURCEID = host1  
- METADATA = [mysql_server_id=1687011;dbms_type=mysql;is_metadata=true;service=alpha;  
            shard=tungsten_alpha;heartbeat=dataload]  
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent  
- OPTIONS = [##charset = UTF-8, autocommit = 1, sql_auto_is_null = 0,  
            foreign_key_checks = 1, unique_checks = 1, sql_mode = 'IGNORE_SPACE',  
            character_set_client = 33, collation_connection = 33, collation_server = 8]  
- SCHEMA = tungsten_alpha  
- SQL(0) = UPDATE tungsten_alpha.heartbeat SET source_tstamp= '2013-07-19 12:14:57',  
           salt= 9, name= 'dataload' WHERE id= 1
```

During replication, slaves identify the heartbeat and record this information into their own `heartbeat` table. Because the heartbeat is recorded into the transaction history log, the specific sequence number of the transaction, and the event itself can be easily identified.

9.16.3.8. `trepctl load` Command

Load the replicator service.

```
trepctl load
```

Load the replicator service. The service name must be specified on the command-line, even when only one service is configured:

```
shell> repctl load
Operation failed: You must specify a service name using -service
```

The service name can be specified using the `-service` option:

```
shell> repctl -service alpha load
Service loaded successfully: name=alpha
```

9.16.3.9. `trepctl offline` Command

The `trepctl offline` command puts the replicator into the offline state, stopping replication.

```
trepctl offline [ -immediate ]
```

To put the replicator offline:

```
shell> repctl offline
```

While offline:

- Transactions are not extracted from the source dataserver.
- Transactions are not applied to the destination dataserver.

Certain operations on the replicator, including updates to the operating system and dataserver should be performed while in the offline state.

By default, the replicator goes offline in deferred mode, allowing the current transactions being read from the binary log, or applied to the dataserver to complete, the sequence number table in the database is updated, and the replicator is placed offline, stopping replication.

To stop replication immediately, within the middle of an executing transaction, use the `-immediate` option:

```
shell> repctl offline -immediate
```

9.16.3.10. `trepctl offline-deferred` Command

The `trepctl offline-deferred` sets a future sequence, event or heartbeat as the trigger to put the replicator in the offline state.

```
trepctl offline-deferred [ -at-event event ] [ -at-heartbeat [heartbeat] ] [ -at-seqno seqno ] [ -at-time YYYY-MM-DD_hh:mm:ss ]
```

Where:

Table 9.24. `trepctl offline-deferred` Command Options

Option	Description
<code>-at-event event</code> [274]	Go offline at the specified event
<code>-at-heartbeat [heartbeat]</code> [274]	Go offline when the specified heartbeat is identified
<code>-at-seqno seqno</code> [273]	Go offline at the specified sequence number
<code>-at-time YYYY-MM-DD_hh:mm:ss</code> [274]	Go offline at the specified time

The `trepctl offline-deferred` command can be used to put the replicator into an offline state at some future point in the replication stream by identifying a specific trigger. The replicator must be online when the `trepctl offline-deferred` command is given; if the replicator is not online, the command is ignored.

The offline process performs a clean offline event, equivalent to executing `trepctl offline`. See Section 9.16.3.9, “`trepctl offline` Command”.

The supported triggers are:

- `-at-seqno` [273]

Specifies a transaction sequence number (GTID) where the replication will be stopped. For example:

```
shell> trepctl offline-deferred -at-seqno 3800
```

The replicator goes into offline at the end of the matching transaction. In the above example, sequence 3800 would be applied to the dataserver, then the replicator goes offline.

- [-at-event \[274\]](#)

Specifies the event where replication should stop:

```
shell> trepctl offline-deferred -at-event 'mysql-bin.000009:000000000088140;0'
```

Because there is not a one-to-one relationship between global transaction IDs and events, the replicator will go offline at a transaction that has an event ID higher than the deferred event ID. If the event specification is located within the middle of a THL transaction, the entire transaction is applied.

- [-at-heartbeat \[274\]](#)

Specifies the name of a specific heartbeat to look for when replication should be stopped.

- [-at-time \[274\]](#)

Specifies a time (using the format YYYY-MM-DD_hh:mm:ss) at which replication should be stopped. The time must be specified in full (date and time to the second).

```
shell> trepctl offline-deferred -at-time 2013-09-01_00:00:00
```

The transaction being executed at the time specified completes, then the replicator goes offline.

If any specified deferred point has already been reached, then the replicator will go offline anyway. For example, if the current sequence number is 3800 and the deferred sequence number specified is 3700, then the replicator will go offline immediately just as if the [trepctl offline](#) command has been used.

When a trigger is reached, For example if a sequence number is given, that sequence will be applied and then the replicator will go offline.

The status of the pending [trepctl offline-deferred](#) setting can be identified within the status output within the [offlineRequests](#) field:

```
shell> trepctl status
...
offlineRequests      : Offline at sequence number: 3810
```

Multiple [trepctl offline-deferred](#) commands can be given for each corresponding trigger type. For example, below three different triggers have been specified, sequence number, time and heartbeat event, with the status showing each deferred event separated by a semicolon:

```
shell> trepctl status
...
offlineRequests      : Offline at heartbeat event: dataloaded;Offline at »
sequence number: 3640;Offline at time: 2013-09-01 00:00:00 EDT
```

Offline deferred settings are cleared when the replicator is put into the offline state, either manually or automatically.

9.16.3.11. [trepctl online](#) Command

The [trepctl online](#) command puts the replicator into the online state. During the state change from offline to online various options can be used to control how the replicator goes back on line. For example, the replicator can be placed online, skipping one or more faulty transactions or disabling specific configurations.

```
trepctl online [-base-seqno x] [-force] [-from-event event] [-no-checksum] [-provision [SCN]] [-skip-seqno seqdef] [-until-event event] [-until-heartbeat [name]] [-until-seqno seqno] [-until-time YYYY-MM-DD_hh:mm:ss]
```

Where:

Table 9.25. [trepctl online](#) Command Options

Option	Description
<code>-base-seqno x</code>	On a master, restart replication using the specified sequence number
<code>-force</code>	Force the online state
<code>-from-event event</code>	Start replication from the specified event
<code>-no-checksum</code>	Disable checksums for all events when going online
<code>-provision [SCN]</code>	Start provisioning using the parallel extractor

Option	Description
<code>-skip-seqno seqdef</code>	Skip one, multiple, or ranges of sequence numbers before going online
<code>-until-event event</code>	Define an event when replication will stop
<code>-until-heartbeat [name]</code>	Define a heartbeat when replication will stop
<code>-until-seqno seqno</code>	Define a sequence no when replication will stop
<code>-until-time YYYY-MM-DD_hh:mm:ss</code>	Define a time when replication will stop

The `trepctl online` command attempts to switch replicator into the online state. The replicator may need to be put online because it has been placed offline for maintenance, or due to a failure.

To put the replicator online use the standard form of the command:

```
shell> trepctl online
```

Going online may fail if the reason for going offline was due to a fault in processing the THL, or in applying changes to the dataserver. The replicator will refuse to go online if there is a fault, but certain failures can be explicitly bypassed.

9.16.3.11.1. Going Online from Specific Transaction Points

If there is one, or more, event in the THL that could not be applied to the slave because of a mismatch in the data (for example, a duplicate key), the event or events can be skipped using the `-skip-seqno` option. For example, the status shows that a statement failed:

```
shell> trepctl status
...
pendingError      : Event application failed: seqno=5250 fragno=0 *
    message=java.sql.SQLException: Statement failed on slave but succeeded on master
...
```

To skip the single sequence number, `5250`, shown:

```
shell> trepctl online -skip-seqno 5250
```

The sequence number specification can be specified according to the following rules:

- A single sequence number:

```
shell> trepctl online -skip-seqno 5250
```

- A sequence range:

```
shell> trepctl online -skip-seqno 5250-5260
```

- A comma-separated list of individual sequence numbers and/or ranges:

```
shell> trepctl online -skip-seqno 5250,5251,5253-5260
```

9.16.3.11.2. Going Online from a Base Sequence Number

Warning

To set the position of the replicator, the `dsctl` command should 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` (in [Tungsten Clustering for MySQL 5.0 Manual]) role only. Use on slaves may lead to duplication or corruption of data.

9.16.3.11.3. Going Online from a Specific Event

Warning

To set the position of the replicator, the `dsctl` command should 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> treptcl online -from-event 'mysql-bin.000011:000000000002552;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.

9.16.3.11.4. 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> treptcl status
...
minimumStoredSeqNo      : 0
offlineRequests          : Offline at sequence number: 5262;Offline at time: 2014-01-01 00:00:00 EST
pendingError              : NONE
...
```

If the replicator goes offline for any reason before the deferred offline state is reached, the deferred settings are lost.

9.16.3.11.4.1. Going Online Until Specified Sequence Number

To go online until a specific transaction ID, use *-until-seqno*:

```
shell> treptcl online -until-seqno 5260
```

This will process all transactions up to, and including, sequence 5260, at which point the replicator will go offline.

9.16.3.11.4.2. Going Online Until Specified Event

To go online until a specific event ID:

```
shell> treptcl online -until-event 'mysql-bin.000011:000000000003057;0'
```

Replication will go offline when the event ID up to the specified event has been processed.

9.16.3.11.4.3. Going Online Until Heartbeat

To go online until a heartbeat event:

```
shell> treptcl 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> treptcl online -until-heartbeat load-finished
```

9.16.3.11.4.4. Going Online Until Specified Time

To go online until a specific date and time:

```
shell> treptcl online -until-time 2014-01-01_00:00:00
```

Replication will go offline once the transaction being processed at the time specified has completed.

9.16.3.11.5. 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> treptcl online -force
```

9.16.3.11.6. Going Online and Starting Provisioning

The replicator can be configured to go online and start populating the THL using information generated by the [Section 8.10, “Using the Parallel Extractor”](#) process. This generates THL information, inserting the data generated by the parallel extractor before starting to extract data from the source database. This effectively populates the THL (and downstream slave databases) with existing data from the source database.

To start provisioning, extracting all data:

```
shell> treptcl online -provision
```

To start provisioning, extracting data from the existing database using a specific reference point, such as the Oracle System Change Number (SCN), append the number to the example:

```
shell> treptcl online -provision 45987459
```

9.16.3.11.7. Going Online without Validating Checksum

In the event of a checksum problem in the THL, checksums can be disabled using the `-no-checksum` option:

```
shell> treptcl 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.

9.16.3.12. treptcl properties Command

Display a list of all the internal properties. The list can be filtered.

```
treptcl properties [-filter name] [-values]
```

The list of properties can be used to determine the current configuration:

```
shell> treptcl properties
{
  "replicator.store.thl.log_file_retention": "7d",
  "replicator.filter.bidiSlave.allowBidiUnsafe": "false",
  "replicator.extractor.dbms.binlog_file_pattern": "mysql-bin",
  "replicator.filter.pkey.url": ">
    \"jdbc:mysql:thin://host2:3306/tungsten_alpha?createDB=true",
  ...
}
```

Note

Passwords are not displayed in the output.

The information is output as a JSON object with key/value pairs for each property and corresponding value.

The list can be filtered using the `-filter` option:

```
shell> treptcl 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> treptcl properties -filter site.name -values
default
```

If a filter that would select multiple values is specified, all the values are listed without field names:

```
shell> trepctl properties -filter shard -values
com.continuent.tungsten.replicator.shard.ShardFilter
com.continuent.tungsten.replicator.filter.JavaScriptFilter
1000
false
../../../../tungsten-replicator//samples/extensions/javascript/shardbyseqno.js
error
../../../../tungsten-replicator//samples/extensions/javascript/shardbytable.js
true
false
stringent
com.continuent.tungsten.replicator.filter.JavaScriptFilter
false
error
```

9.16.3.13. `trepctl purge` Command

Forces all logins on the attached database, other than those directly related to Tungsten Replication, 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]
```

Where:

Table 9.26. `trepctl purge` Command Options

Option	Description
<code>-limit s</code> [278]	Specify the waiting time for the operation

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` [278] option:

```
shell> trepctl purge -y
Directing replicator to purge non-Tungsten sessions
Number of sessions purged: 0
```

An optional parameter, `-wait` [278], defines the period of time that the operation will wait before returning to the command-line.

An optional parameter, `-limit` [278], defines the period of time that the operation will wait before returning to the command-line.

9.16.3.14. `trepctl reset` Command

The `trepctl reset` command resets an existing replicator service, performing the following operations:

- Deleting the local THL and relay directories
- Removes the Tungsten schema from the dataserver
- Removes any dynamic properties that have previously been set

The service name must be specified, using `-service`.

```
trepctl reset [-all] [-db] [-relay] [-thl] [-y]
```

Where:

Table 9.27. `trepctl reset` Command Options

Option	Description
<code>-all</code>	Deletes the thl directory, relay logs directory and tungsten database for the service. Same as specifying <code>-thl -relay -db</code>

Option	Description
<code>-db</code>	Deletes the tungsten_{service_name} database for the service
<code>-relay</code>	Deletes the relay directory for the service
<code>-thl</code>	Deletes the thl directory for the service
<code>-y [279]</code>	Indicates that the command should continue without interactive confirmation

To reset a replication service, the replication service must be offline and the service name must be specified:

```
shell> trepctl offline
```

Execute the `trepctl reset` command:

```
shell> trepctl -service alpha reset
Do you really want to delete replication service alpha completely? [yes/NO]
```

You will be prompted to confirm the deletion. To ignore the interactive prompt, use the `-y [279]` option:

```
shell> trepctl -service alpha reset -y
```

Then put the replicator back online again:

```
shell> trepctl online
```

9.16.3.15. `trepctl restore` Command

Tungsten Replication 5.0.0. This feature has been deprecated in Tungsten Replication 5.0.0 and will be removed in a future release.

Restores the database on a host from a previous backup.

`trepctl` capabilities

Once the restore has been completed, the node will remain in the `OFFLINE [207]` state. The datasource should be switched `ONLINE [207]` 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.

9.16.3.16. `trepctl setrole` Command

The `trepctl setrole` command changes the role of the replicator service. This command can be used to change a configured host between slave and master roles, for example during switchover.

`trepctl setrole [-role master|relay|slave] [-uri]`

Where:

Table 9.28. `trepctl setrole` Command Options

Option	Description
<code>-role</code>	Replicator role
<code>-uri [279]</code>	URI of the master

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/
```

9.16.3.17. `trepctl shard` Command

The `trepctl shard` command provides an interface to the replicator shard system definition system.

`trepctl shard [-delete shard] [-insert shard] [-list] [-update shard]`

Where:

Table 9.29. **trepctl shard** Command Options

Option	Description
<code>-delete shard</code>	Delete a shard definition
<code>-insert shard</code>	Add a new shard definition
<code>-list</code>	List configured shards
<code>-update shard</code>	Update a shard definition

The replicator shard system is used during multi-site replication configurations to control where information is replicated.

9.16.3.17.1. Listing Current Shards

To obtain a list of the currently configured shards:

```
shell> trepctl shard -list
shard_id master critical
alpha     sales   true
```

The shard map information can also be captured and then edited to update existing configurations:

```
shell> trepctl shard -list>shard.map
```

9.16.3.17.2. Inserting a New Shard Configuration

To add a new shard map definition, either enter the information interactively:

```
shell> trepctl shard -insert
Reading from standard input
...
1 new shard inserted
```

Or import from a file:

```
shell> trepctl shard -insert < shard.map
Reading from standard input
1 new shard inserted
```

9.16.3.17.3. Updating an Existing Shard Configuration

To update a definition:

```
shell> trepctl shard -update < shard.map
Reading from standard input
1 shard updated
```

9.16.3.17.4. Deleting a Shard Configuration

To delete a single shard definition, specify the shard name:

```
shell> trepctl shard -delete alpha
```

9.16.3.18. **trepctl status** Command

The **trepctl status** command provides status information about the selected data service. The status information by default is a generic status report containing the key fields of status information. More detailed service information can be obtained by specifying the status name with the `-name` parameter.

The format of the command is:

```
trepctl status [ -json ] [ -namechannel-assignmentsservicesshardsstagesstorestaskswatches ]
```

Where:

Table 9.30. **trepctl status** Command Options

Option	Description
<code>-json</code>	Output the information in JSON format

Option	Description
<code>-name</code>	Select a specific group of status information

For example, to get the basic status information:

```
shell> treptl status
Processing status command...
NAME      VALUE
----      -----
appliedLastEventId : mysql-bin.000007:0000000000001353:0
appliedLastSeqno : 2504
appliedLatency : 0.53
channels : 1
clusterName : default
currentEventId : mysql-bin.000007:0000000000001353
currentTimeMillis : 1369233160014
dataServerHost : host1
extensions :
latestEpochNumber : 2500
masterConnectUri :
masterListenUri : thl://host1:2112/
maximumStoredSeqNo : 2504
minimumStoredSeqNo : 0
offlineRequests : NONE
pendingError : NONE
pendingErrorCode : NONE
pendingErrorEventId : NONE
pendingErrorSeqno : -1
pendingExceptionMessage: NONE
pipelineSource : jdbc:mysql:thin://host1:3306/
relativeLatency : 1875.013
resourcePrecedence : 99
rmiPort : 10000
role : master
segnoType : java.lang.Long
serviceName : alpha
serviceType : local
simpleServiceName : alpha
siteName : default
sourceId : host1
state : ONLINE
timeInStateSeconds : 1874.512
transitioningTo :
uptimeSeconds : 1877.823
version : Tungsten Replicator 5.0.1 build 136
Finished status command...
```

For more information on the field information output, see [Section E.2, “Generated Field Reference”](#).

9.16.3.18.1. Getting Detailed Status

More detailed information about selected areas of the replicator status can be obtained by using the `-name` option.

9.16.3.18.1.1. Detailed Status: Channel Assignments

When using a single threaded replicator service, the `treptl status -name channel-assignments` will output an empty status. In parallel replication deployments, the `treptl 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> treptl 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)...
```

9.16.3.18.1.2. Detailed Status: Services

The `treptl status -name services` status output shows a list of the currently configure internal services that are defined within the replicator.

```
shell> treptl 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)...
```

9.16.3.18.1.3. Detailed Status: Shards

9.16.3.18.1.4. Detailed Status: Stages

The `trepctl status -name stages` status output lists the individual stages configured within the replicator, showing each stage, configuration, filters and other parameters applied at each replicator stage:

```
shell> trepctl status -name stages
Processing status command (stages)...
NAME          VALUE
----          -----
applier.class   : com.continuent.tungsten.replicator.thl.THLStoreApplier
applier.name    : thl-applier
blockCommitRowCount: 1
committedMinSeqno : 15
extractor.class : com.continuent.tungsten.replicator.thl.RemoteTHLExtractor
extractor.name   : thl-remote
name          : remote-to-thl
processedMinSeqno : -1
taskCount      : 1
NAME          VALUE
----          -----
applier.class   : com.continuent.tungsten.replicator.thl.THLParallelQueueApplier
applier.name    : parallel-q-applier
blockCommitRowCount: 10
committedMinSeqno : 15
extractor.class : com.continuent.tungsten.replicator.thl.THLStoreExtractor
extractor.name   : thl-extractor
name          : thl-to-q
processedMinSeqno : -1
taskCount      : 1
NAME          VALUE
----          -----
applier.class   : com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier
applier.name    : dbms
blockCommitRowCount: 10
committedMinSeqno : 15
extractor.class : com.continuent.tungsten.replicator.thl.THLParallelQueueExtractor
extractor.name   : parallel-q-extractor
filter.0.class  : com.continuent.tungsten.replicator.filter.TimeDelayFilter
filter.0.name    : delay
filter.1.class  : com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter
filter.1.name    : mysqlsessions
filter.2.class  : com.continuent.tungsten.replicator.filter.PrimaryKeyFilter
filter.2.name    : pkey
name          : q-to-dbms
processedMinSeqno : -1
taskCount      : 5
Finished status command (stages)...
```

9.16.3.18.1.5. 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
name          : binlog-to-q
```

```

processedMinSeqno : 224
taskCount : 1
NAME      VALUE
-----
applier.class : com.continuent.tungsten.replicator.thl.THLStoreApplier
applier.name : autoflush-thl-applier
blockCommitRowCount: 10
committedMinSeqno : 224
extractor.class : com.continuent.tungsten.replicator.storage.InMemoryQueueAdapter
extractor.name : queue
name : q-to-thl
processedMinSeqno : 224
taskCount : 1
Finished status command (stages)...

```

When running parallel replication, the output shows the store name, sequence number and status information for each parallel replication channel:

```

shell> trepctl status -name stores
Processing status command (stores)...
NAME      VALUE
-----
activeSeqno : 15
doChecksum : false
flushIntervalMillis : 0
fsyncOnFlush : false
logConnectionTimeout : 28800
logDir : /opt/continuent/thl/alpha
logFileRetainMillis : 604800000
logFileSize : 1000000000
maximumStoredSeqNo : 16
minimumStoredSeqNo : 0
name : thl
readOnly : false
storeClass : com.continuent.tungsten.replicator.thl.THL
timeoutMillis : 2147483647
NAME      VALUE
-----
criticalPartition : -1
discardCount : 0
estimatedOfflineInterval: 0.0
eventCount : 1
headSeqno : 16
intervalGuard : AtomicIntervalGuard (array is empty)
maxDelayInterval : 60
maxOfflineInterval : 5
maxSize : 10
name : parallel-queue
queues : 5
serializationCount : 0
serialized : false
stopRequested : false
store.0 : THLParallelReadTask task_id=0 thread_name=store-thl-0 hi_seqno=16 lo_seqno=16 read=1 accepted=1 discarded=0
store.1 : THLParallelReadTask task_id=1 thread_name=store-thl-1 hi_seqno=16 lo_seqno=16 read=1 accepted=0 discarded=0
store.2 : THLParallelReadTask task_id=2 thread_name=store-thl-2 hi_seqno=16 lo_seqno=16 read=1 accepted=0 discarded=0
store.3 : THLParallelReadTask task_id=3 thread_name=store-thl-3 hi_seqno=16 lo_seqno=16 read=1 accepted=0 discarded=0
store.4 : THLParallelReadTask task_id=4 thread_name=store-thl-4 hi_seqno=16 lo_seqno=16 read=1 accepted=0 discarded=0
storeClass : com.continuent.tungsten.replicator.thl.THLParallelQueue
syncInterval : 10000
Finished status command (stores)...

```

9.16.3.18.1.6. Detailed Status: Tasks

The `trepctl status -name tasks` command outputs the current list of active tasks within a given service, with one block for each stage within the replicator service.

```

shell> trepctl status -name tasks
Processing status command (tasks)...
NAME      VALUE
-----
appliedLastEventId : mysql-bin.000015:0000000000001117:0
appliedLastSeqno : 5271
appliedLatency : 4656.176
applyTime : 0.017
averageBlockSize : 0.500
cancelled : false
commits : 10
currentBlockSize : 0
currentLastEventId : mysql-bin.000015:0000000000001117:0
currentLastFragno : 0
currentLastSeqno : 5271
eventCount : 5

```

```

extractTime      : 0.385
filterTime       : 0.0
lastCommittedBlockSize: 1
lastCommittedBlockTime: 0.017
otherTime        : 0.004
stage            : remote-to-thl
state            : extract
taskId           : 0
NAME             VALUE
----  

appliedLastEventId : mysql-bin.000015:0000000000001117;0
appliedLastSegno   : 5271
appliedLatency     : 4656.188
applyTime         : 0.0
averageBlockSize   : 0.500
cancelled          : false
commits           : 10
currentBlockSize   : 0
currentLastEventId : mysql-bin.000015:0000000000001117;0
currentLastFragno  : 0
currentLastSegno   : 5271
eventCount         : 5
extractTime        : 0.406
filterTime         : 0.0
lastCommittedBlockSize: 1
lastCommittedBlockTime: 0.009
otherTime          : 0.0
stage              : thl-to-q
state              : extract
taskId             : 0
NAME               VALUE
----  

appliedLastEventId : mysql-bin.000015:0000000000001117;0
appliedLastSegno   : 5271
appliedLatency     : 4656.231
applyTime          : 0.066
averageBlockSize   : 0.500
cancelled          : false
commits            : 10
currentBlockSize   : 0
currentLastEventId : mysql-bin.000015:0000000000001117;0
currentLastFragno  : 0
currentLastSegno   : 5271
eventCount          : 5
extractTime         : 0.394
filterTime          : 0.017
lastCommittedBlockSize: 1
lastCommittedBlockTime: 0.033
otherTime           : 0.001
stage               : q-to-dbms
state               : extract
taskId              : 0
Finished status command (tasks)...

```

The list of tasks and information provided depends on the role of the host, the number of stages, and whether parallel apply is enabled.

9.16.3.18.1.7. Detailed Status: Watches

9.16.3.18.2. 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"](#).

treptl status JSON Output

```
{
"uptimeSeconds": "2128.682",
"masterListenUri": "thl://host1:2112/",
"clusterName": "default",
"pendingExceptionMessage": "NONE",
"appliedLastEventId": "mysql-bin.000007:0000000000001353:0",
"pendingError": "NONE",
"resourcePrecedence": "99",
"transitioningTo": "",
"offlineRequests": "NONE",
"state": "ONLINE",
"simpleServiceName": "alpha",
}
```

```

"extensions": "",
"pendingErrorEventId": "NONE",
"sourceId": "host1",
"serviceName": "alpha",
"version": "Tungsten Replicator 5.0.1 build 136",
"role": "master",
"currentTimeMillis": "1369233410874",
"masterConnectUri": "",
"rmiPort": "10000",
"siteName": "default",
"pendingErrorSeqno": "-1",
"appliedLatency": "0.53",
"pipelineSource": "jdbc:mysql:thin://host1:3306/",
"pendingErrorCode": "NONE",
"maximumStoredSeqNo": "2504",
"latestEpochNumber": "2500",
"channels": "1",
"appliedLastSeqno": "2504",
"serviceType": "local",
"seqnoType": "java.lang.Long",
"currentEventId": "mysql-bin.000007:0000000000001353",
"relativeLatency": "2125.873",
"minimumStoredSeqNo": "0",
"timeInStateSeconds": "2125.372",
"dataServerHost": "host1"
}

```

9.16.3.18.2.1. Detailed Status: Channel Assignments JSON Output

```

shell> treptctl status -name channel-assignments -json
[
  {
    "channel" : "0",
    "shard_id" : "cheffy"
  },
  {
    "channel" : "0",
    "shard_id" : "tungsten_alpha"
  }
]

```

9.16.3.18.2.2. Detailed Status: Services JSON Output

```

shell> treptctl status -name services -json
[
  {
    "totalAssignments" : "2",
    "accessFailures" : "0",
    "storeClass" : "com.continuent.tungsten.replicator.channel.ChannelAssignmentService",
    "name" : "channel-assignment",
    "maxChannel" : "0"
  }
]

```

9.16.3.18.2.3. Detailed Status: Shards JSON Output

```

shell> treptctl status -name shards -json
[
  {
    "stage" : "q-to-dbms",
    "appliedLastEventId" : "mysql-bin.000007:000000007224342:0",
    "appliedLatency" : "63.099",
    "appliedLastSeqno" : "2514",
    "eventCount" : "16",
    "shardId" : "cheffy"
  }
]

```

9.16.3.18.2.4. Detailed Status: Stages JSON Output

```

shell> treptctl status -name stages -json
[
  {
    "applier.name" : "thl-applier",
    "applier.class" : "com.continuent.tungsten.replicator.thl.THLStoreApplier",
    "name" : "remote-to-thl",
    "extractor.name" : "thl-remote",
    "taskCount" : "1",
    "committedMinSeqno" : "2504",
    "blockCommitRowCount" : "1",
    "processedMinSeqno" : "-1",
    "extractor.class" : "com.continuent.tungsten.replicator.thl.RemoteTHLExtractor"
  }
]

```

```

},
{
  "applier.name" : "parallel-q-applier",
  "applier.class" : "com.continuent.tungsten.replicator.storage.InMemoryQueueAdapter",
  "name" : "thl-to-q",
  "extractor.name" : "thl-extractor",
  "taskCount" : "1",
  "committedMinSeqno" : "2504",
  "blockCommitRowCount" : "10",
  "processedMinSeqno" : "-1",
  "extractor.class" : "com.continuent.tungsten.replicator.thl.THLStoreExtractor"
},
{
  "applier.name" : "dbms",
  "applier.class" : "com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier",
  "filter.2.name" : "bidiSlave",
  "name" : "q-to-dbms",
  "extractor.name" : "parallel-q-extractor",
  "filter.1.name" : "pkey",
  "taskCount" : "1",
  "committedMinSeqno" : "2504",
  "filter.2.class" : "com.continuent.tungsten.replicator.filter.BidiRemoteSlaveFilter",
  "filter.1.class" : "com.continuent.tungsten.replicator.filter.PrimaryKeyFilter",
  "filter.0.class" : "com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter",
  "blockCommitRowCount" : "10",
  "filter.0.name" : "mysqlsessions",
  "processedMinSeqno" : "-1",
  "extractor.class" : "com.continuent.tungsten.replicator.storage.InMemoryQueueAdapter"
}
]

```

9.16.3.18.2.5. Detailed Status: Stores JSON Output

```

shell> treptctl status -name stores -json
[
  {
    "logConnectionTimeout" : "28800",
    "doChecksum" : "false",
    "name" : "thl",
    "flushIntervalMillis" : "0",
    "logFileSize" : "100000000",
    "logDir" : "/opt/continuent/thl/alpha",
    "activeSeqno" : "2561",
    "readOnly" : "false",
    "timeoutMillis" : "2147483647",
    "storeClass" : "com.continuent.tungsten.replicator.thl.THL",
    "logFileRetainMillis" : "604800000",
    "maximumStoredSeqNo" : "2565",
    "minimumStoredSeqNo" : "2047",
    "fsyncOnFlush" : "false"
  },
  {
    "storeClass" : "com.continuent.tungsten.replicator.storage.InMemoryQueueStore",
    "maxSize" : "10",
    "storeSize" : "7",
    "name" : "parallel-queue",
    "eventCount" : "119"
  }
]

```

9.16.3.18.2.6. Detailed Status: Tasks JSON Output

```

shell> treptctl status -name tasks -json
[
  {
    "filterTime" : "0.0",
    "stage" : "remote-to-thl",
    "currentLastFragno" : "1",
    "taskId" : "0",
    "currentLastSeqno" : "2615",
    "state" : "extract",
    "extractTime" : "604.297",
    "applyTime" : "16.708",
    "averageBlockSize" : "0.982",
    "otherTime" : "0.017",
    "appliedLastEventId" : "mysql-bin.000007:0000000111424440;0",
    "appliedLatency" : "63.787",
    "currentLastEventId" : "mysql-bin.000007:0000000111424440;0",
    "eventCount" : "219",
    "appliedLastSeqno" : "2615",
    "cancelled" : "false"
  },
  {

```

```

    "filterTime" : "0.0",
    "stage" : "thl-to-q",
    "currentLastFragno" : "1",
    "taskId" : "0",
    "currentLastSeqno" : "2615",
    "state" : "extract",
    "extractTime" : "620.715",
    "applyTime" : "0.344",
    "averageBlockSize" : "1.904      ",
    "otherTime" : "0.006",
    "appliedLastEventId" : "mysql-bin.000007:000000111424369;0",
    "appliedLatency" : "63.834",
    "currentLastEventId" : "mysql-bin.000007:000000111424440;0",
    "eventCount" : "219",
    "appliedLastSeqno" : "2615",
    "cancelled" : "false"
},
{
    "filterTime" : "0.263",
    "stage" : "q-to-dbms",
    "currentLastFragno" : "1",
    "taskId" : "0",
    "currentLastSeqno" : "2614",
    "state" : "apply",
    "extractTime" : "533.471",
    "applyTime" : "61.618",
    "averageBlockSize" : "1.160      ",
    "otherTime" : "24.052",
    "appliedLastEventId" : "mysql-bin.000007:000000110392640;0",
    "appliedLatency" : "63.178",
    "currentLastEventId" : "mysql-bin.000007:000000110392711;0",
    "eventCount" : "217",
    "appliedLastSeqno" : "2614",
    "cancelled" : "false"
}
]

```

9.16.3.18.2.7. Detailed Status: Tasks JSON Output

```
shell> treptl status -name watches -json
```

9.16.3.19. treptl unload Command

Unload the replicator service.

```
treptl unload [-y]
```

Unload the replicator service entirely. An interactive prompt is provided to confirm the shutdown:

```
shell> treptl unload
Do you really want to unload replication service alpha? [yes/NO]
```

To disable the prompt, use the `-y` [287] option:

```
shell> treptl unload -y
Service unloaded successfully: name=alpha
```

The name of the service unloaded is provided for confirmation.

9.16.3.20. treptl wait Command

The `treptl wait` command waits for the replicator to enter a specific state, or for a specific sequence number to be applied to the dataserver.

```
treptl wait [-applied seqno] [-limit s] [-state st]
```

Where:

Table 9.31. `treptl wait` Command Options

Option	Description
<code>-applied seqno</code> [288]	Specify the sequence number to be waited for
<code>-limit s</code> [288]	Specify the number of seconds to wait for the operation to complete
<code>-state st</code> [288]	Specify a state to be waited for

The command will wait for the specified occurrence, of either a change in the replicator status (i.e. `ONLINE` [207]), or for a specific sequence number to be applied. For example, to wait for the replicator to go into the `ONLINE` [207] state:

```
shell> trepctl wait -state ONLINE
```

This can be useful in scripts when the state maybe changed (for example during a backup or restore operation), allowing for an operation to take place once the requested state has been reached. Once reached, `trepctl` returns with exit status 0.

To wait a specific sequence number to be applied:

```
shell> trepctl wait -applied 2000
```

This can be useful when performing bulk loads where the sequence number where the bulk load completed is known, or when waiting for a specific sequence number from the 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 bee successfully applied.

If the optional `-limit` [288] option is used, then `trepctl` waits for the specified number of seconds for the request event to occur. For example, to wait for 10 seconds for the replicator to go online:

```
shell> trepctl wait -state ONLINE -limit 10
Wait timed out!
```

If the requested event does not take place before the specified time limit expires, then `trepctl` returns with the message 'Wait timed out!', and an exit status of 1.

9.17. The `tpasswd` Command

Table 9.32. `tpasswd` Common Options

Option	Description
<code>--create, -c</code>	Creates a new user/password
<code>-delete, -d</code>	Delete a user/password combination
<code>-e, --encrypted.password</code>	Encrypt the password
<code>--file, -f</code>	Specify the location of the security.properties file

9.18. 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](#) [207], but the command can place the replicator offline automatically as part of the provisioning process.

Once these prerequisites have been met, the basic method of executing the command is to specify the location of the Tungsten Replicator tarball, MySQL tarball and the databases that you want to provision:

```
shell> tungsten_provision_thl \
    --tungsten-replicator-package=/home/tungsten/tungsten-replicator-3.0.0-254.tar.gz \
    --mysql-package=/home/tungsten/mysql-5.6.20-linux-glibc2.5-x86_64.tar.gz \
    --schemas=test
NOTE >>The THL has been provisioned to mysql-bin.000025:493 on host1:3306
```

The command reports the MySQL binary log point and host on which the THL has been provisioned. Put the Tungsten Replicator back online from the reported position:

```
shell> treptctl 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.

9.18.1. 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` [290] 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 `treptctl 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` [290] and `--extract-from-host` [290] 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
NOTE >>The THL has been provisioned to mysql-bin.000025:493 on rds-host1:3306
# Run `CALL mysql.rds_start_replication();` on the RDS Read Replica
```

9.18.2. `tungsten_provision_thl` Reference

The format of the command is:

```
tungsten_provision_thl [ --cleanup-on-failure ] [ --clear-logs ] [ --directory ] [ --extract-from=mysql-native-slaverdsrds-read-replicatingstenslave ] [ --extract-from-host ] [ --extract-from-port ] [ --help, -h ] [ --info, -i ] [ --java-file-encoding ] [ --json ] [ --mysql-
```

```
package] [--net-ssh-option] [--notice,-n] [--offline] [--quiet,-q] [--sandbox-directory] [--sandbox-mysql-port] [--sandbox-password] [--sandbox-rmi-port] [--sandbox-user] [--schemas] [--service] [--tungsten-replicator-package] [--validate] [--verbose,-v]
```

Where:

[--cleanup-on-failure \[290\]](#)

Option	--cleanup-on-failure [290]
Description	Cleanup the sandbox installations when the provision process fails
Value Type	boolean
Default	false

[--clear-logs \[290\]](#)

Option	--clear-logs [290]
Description	Delete all THL and relay logs for the service
Value Type	boolean
Default	false

[--directory \[290\]](#)

Option	--directory [290]
Description	Use this installed Tungsten directory as the base for all operations
Value Type	string

[--extract-from \[290\]](#)

Option	--extract-from [290]	
Description	The type of server you are going to extract from	
Value Type	string	
Valid Values	mysql-native-slave	A MySQL native slave with binary logging enabled
	rds	An Amazon RDS instance
	rds-read-replica	An Amazon RDS read replica instance
	tungsten-slave	An instance with Tungsten Replication already installed with generated THL

[--extract-from-host \[290\]](#)

Option	--extract-from-host [290]
Description	The hostname of a different MySQL server that will be used as the source for mysqldump
Value Type	string

The hostname of a different MySQL server that will be used as the source for [mysqldump](#). When given, the script will use [SHOW SLAVE STATUS](#) to determine the binary log position on the master server. You must run [STOP SLAVE](#) prior to executing [tungsten_provision_thl](#).

[--extract-from-port \[290\]](#)

Option	--extract-from-port [290]
Description	The listening port of a different MySQL server that will be used as the source for mysqldump
Value Type	numeric

[--help \[290\]](#)

Option	--help [290]
Aliases	-h [290]
Description	Display the help message

Value Type	string
--info [291]	
Option	--info [291]
Aliases	<code>-i</code> [291]
Description	Provide information-level messages
Value Type	string
--java-file-encoding [291]	
Option	--java-file-encoding [291]
Description	Java platform charset
Value Type	string
--json [291]	
Option	--json [291]
Description	Provide return code and logging messages as a JSON object after the script finishes
Value Type	string
--mysql-package [291]	
Option	--mysql-package [291]
Description	The location of a the MySQL tar.gz package
Value Type	string
--net-ssh-option [291]	
Option	--net-ssh-option [291]
Description	Sets additional options for SSH usage by the system, such as port numbers and passwords.
Value Type	string
Default	default
Sets options for the <code>Net::SSH</code> 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 <code>--net-ssh-option=port=80</code> [291] will use port 80 for SSH communication in place of the default port 22.	
For more information on the options, see http://net-ssh.github.com/ssh/v2/api/classes/Net/SSH.html#M000002 .	
--notice [291]	
Option	--notice [291]
Aliases	<code>-n</code> [291]
Description	Provide notice-level messages
Value Type	string
--offline [291]	
Option	--offline [291]
Description	Put required replication services offline before processing
Value Type	boolean
Default	false
--online [291]	
Option	--online [291]
Description	Put required replication services online after successful processing

Value Type	boolean
Default	false

[--quiet \[292\]](#)

Option	--quiet [292]
Aliases	<code>-q</code> [292]
Description	Execute with the minimum of output
Value Type	string

[--sandbox-directory \[292\]](#)

Option	--sandbox-directory [292]
Description	The location to use for storing the temporary replicator and MySQL server
Value Type	string

[--sandbox-mysql-port \[292\]](#)

Option	--sandbox-mysql-port [292]
Description	The listening port for the MySQL Sandbox
Value Type	string
Default	3307

[--sandbox-password \[292\]](#)

Option	--sandbox-password [292]
Description	The password for the MySQL sandbox user
Value Type	string
Default	secret

[--sandbox-rmi-port \[292\]](#)

Option	--sandbox-rmi-port [292]
Description	The listening port for the temporary Tungsten Replicator
Value Type	string
Default	10002

[--sandbox-user \[292\]](#)

Option	--sandbox-user [292]
Description	The MySQL user to create and use in the MySQL Sandbox
Value Type	string
Default	tungsten

[--schemas \[292\]](#)

Option	--schemas [292]
Description	The provision process will be limited to these schemas
Value Type	string

[--service \[292\]](#)

Option	--service [292]
Description	Replication service to read information from
Value Type	string
Default	alpha

`--tungsten-replicator-package [293]`

Option	<code>--tungsten-replicator-package [293]</code>
Description	The location of a fresh Tungsten Replicator tar.gz package
Value Type	string

`--validate [293]`

Option	<code>--validate [293]</code>
Description	Run the script validation for the provided options and files
Value Type	boolean
Default	false

`--verbose [293]`

Option	<code>--verbose [293]</code>
Aliases	<code>-v [293]</code>
Description	Provide verbose-level error messages
Value Type	string

9.19. The `tungsten_provision_slave` Script

The script was added in Tungsten Replicator 2.2.0. It cannot be backported to older versions.

The `tungsten_provision_slave` script allows you to easily provision, or reprovision, a database server using information from a remote host. It implements the Tungsten Script Interface as well as these additional options.

```
tungsten_provision_slave [ --clear-logs ] [ --direct ] [ --directory ] [ -f, --force ] [ --help, -h ] [ --info, -i ] [ --json ] [ --mysqldump ] [ --net-ssh-option ] [ --notice, -n ] [ --offline ] [ --offline-timeout ] [ --online ] [ --service ] [ --source ] [ --source-directory ] [ --validate ] [ --verbose, -v ] [ --xtrabackup ]
```

Where:

Table 9.33. `tungsten_provision_slave` Command-line Options

Option	Description
<code>--clear-logs</code>	Delete all THL and relay logs for the service
<code>--direct</code>	Use the MySQL data directory for staging and preparation
<code>--directory</code>	The \$CONTINUENT_ROOT directory to use for running this command. It will default to the directory you use to run the script.
<code>--force, -f</code>	Continue operation even if script validation fails
<code>--help, -h</code>	Show help text
<code>--info, -i</code>	Display info, notice, warning, and error messages
<code>--json</code>	Output all messages and the return code as a JSON object
<code>--mysqldump</code>	Use mysqldump for generating the information
<code>--net-ssh-option</code>	Provide custom SSH options to use for SSH communication to other hosts.
<code>--notice, -n</code>	Display notice, warning, and error messages
<code>--offline</code>	Put required replication services offline before processing
<code>--offline-timeout</code>	Put required replication services offline before processing
<code>--online</code>	Put required replication services online after successful processing
<code>--service</code>	Replication service to read information from
<code>--source</code>	Server to use as a source for the backup
<code>--source-directory</code>	Directory on --source to find installed software
<code>--validate</code>	Only run script validation

Option	Description
<code>--verbose, -v</code>	Show verbose information during processing
<code>--xtrabackup</code>	Use xtrabackup for generating the information

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](#) [207] or [OFFLINE: NORMAL](#) [207] states. This can be overridden by using the `-f` or `--force` options.

When provisioning masters, for example in [fan-in](#), or when recovering a failed master in a standard master-slave topology, the service must be reset with the [trepcctl reset](#) after the command is finished. The service must also be reset on all slaves.

The `--service` argument is used to determine which database server should be provisioned. If there are multiple services defined in the replicator and one of those is a master, the master service must be specified.

If the installation directory on `--source` is different from the target, specify `--source-directory` to specify where it can be found. This option should point to an installation that is running the `--service` replication service. The `--source-directory` option is not required if the software is installed to the same directory on both servers.

Using `xtrabackup`

The script will use Xtrabackup by default. It will run validation prior to starting to make sure the needed scripts are available. The provision process will run Xtrabackup on the source server and stream the contents to the server you are provisioning. Passing the `--direct` option will empty the MySQL data directory prior to doing the backup and place the streaming backup there. After taking the backup, the script will prepare the directory and restart the MySQL server.

Using `mysqldump`

If you have a small dataset or don't have Xtrabackup, you may pass the `--mysqldump` option to use it. It implements the Tungsten Script Interface as well as these additional options.

Compatibility

The script only works with MySQL at this time.

9.20. The `tungsten_read_master_events` Script

The script was added in Tungsten Replicator 2.2.0. It cannot be backported to older versions.

The `tungsten_read_master_events` displays the raw contents of the master datasource for the given THL records. It implements the Tungsten Script Interface as well as these additional options.

```
tungsten_read_master_events [ --directory ] [ --force ] [ --help, -h ] [ --high ] [ --info, -i ] [ --json ] [ --low ] [ --net-ssh-option ] [ --notice, -n ] [ --service ] [ --source ] [ --validate ] [ --verbose, -v ]
```

Where:

Table 9.34. `tungsten_read_master_events` Command-line Options

Option	Description
<code>--directory</code>	The \$CONTINUENT_ROOT directory to use for running this command. It will default to the directory you use to run the script.
<code>--force</code>	Continue operation even if script validation fails
<code>--help, -h</code>	Show help text
<code>--high</code>	Display events ending with this sequence number
<code>--info, -i</code>	Display info, notice, warning, and error messages
<code>--json</code>	Output all messages and the return code as a JSON object
<code>--low</code>	Display events starting with this sequence number
<code>--net-ssh-option</code>	Provide custom SSH options to use for SSH communication to other hosts.
<code>--notice, -n</code>	Display notice, warning, and error messages

Option	Description
--service	Replication service to read information from
--source	Determine metadata for the --after, --low, --high statements from this host
--validate	Only run script validation
--verbose, -v	Show verbose information during processing

Display all information after a specific sequence number

This may be used when you have had a master failover or would like to see everything that happened after a certain event. It will read the start position from the sequence number passed and allow you to see all events, even if they were not extracted by the replication service.

```
shell> tungsten_read_master_events --after=1792
```

If you provide the --source option, the script will SSH to the host in question and read its THL information.

Display information between two sequence numbers

This will show the raw master data between the two sequence numbers. It is inclusive so the information for the --low option will be included. This will only work if the sourceld for both sequence numbers is the same.

```
shell> tungsten_read_master_events --low=4582 --high=4725
```

Compatibility

The script only works with MySQL at this time.

The script was added in Continuent Tungsten 2.0.1 and Tungsten Replicator 2.2.0. It cannot be backported to older versions.

9.21. The `tungsten_set_position` Script

The script was added in Tungsten Replicator 2.2.0. It cannot be backported to older versions.

The `tungsten_set_position` updates the `trep_commit_seqno` table to reflect the given THL sequence number or provided information. It implements the Tungsten Script Interface as well as these additional options.

```
tungsten_set_position [ --clear-logs ] [ --epoch ] [ --event-id ] [ --high ] [ --low ] [ --offline ] [ --offline-timeout ] [ --online ] [ --replicate-statements ] [ --seqno ] [ --service ] [ --source ] [ --source-directory ] [ --source-id ] [ --sql ]
```

Where:

Table 9.35. `tungsten_set_position` Command-line Options

Option	Description
--clear-logs	Delete all THL and relay logs for the service
--epoch	The epoch number to use for updating the <code>trep_commit_seqno</code> table
--event-id	The event id to use for updating the <code>trep_commit_seqno</code> table
--high	Display events ending with this sequence number
--low	Display events starting with this sequence number
--offline	Put required replication services offline before processing
--offline-timeout	Put required replication services offline before processing
--online	Put required replication services online after successful processing
--replicate-statements	Execute the events so they will be replicated if the service is a master
--seqno	The sequence number to use for updating the <code>trep_commit_seqno</code> table
--service	Replication service to read information from
--source	Determine metadata for the --after, --low, --high statements from this host
--source-directory	Directory on --source to find installed software

Option	Description
--source-id	The source id to use for updating the trep_commit_seqno table
--sql	Only output the SQL statements needed to update the schema

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 deleted from the server following provisioning. This ensures that any corrupted or inconsistent THL records are removed prior to replication coming back online.

The `--service` argument is used to determine which database server should be provisioned.

If the installation directory on `--source` is different from the target, specify `--source-directory` to specify where it can be found. This option should point to an installation that is running the `--service` replication service. The `--source-directory` option is not required if the software is installed to the same directory on both servers.

This command will fail if there is more than one record in the `trep_commit_seqno` table. This may happen if parallel replication does not stop cleanly. You may bypass that error with the `--force` option.

Update trep_commit_seqno with information from a THL event

This will read the THL information from the host specified as `--source`.

```
shell> tungsten_set_position --seqno=5273 --source=db1
```

Update trep_commit_seqno with specific information

The script will also accept specific values to update the `trep_commit_seqno` table. This may be used when bringing a new master service online or when the THL event is no longer available.

```
shell> tungsten_set_position --seqno=5273 --epoch=5264  
--source-id=db1
```

```
shell> tungsten_set_position --seqno=5273 --epoch=5264  
--source-id=db1 --event-id=mysql-bin.000025:0000000000000421
```

Compatibility

The script only works with MySQL at this time.

9.22. The undeployall Command

The `undeployall` command removes 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/trepli...  
/etc/rc0.d/K80trepli...  
/etc/rc1.d/K80trepli...  
/etc/rc2.d/S80trepli...  
/etc/rc3.d/S80trepli...  
/etc/rc4.d/S80trepli...  
/etc/rc5.d/S80trepli...  
/etc/rc6.d/K80trepli...
```

To enable the scripts on the system, use `deployall`.

9.23. The updateCDC.sh Command

The `updateCDC.sh` script updates 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 9.17, “`setupCDC.conf` Configuration Options”](#).

To use, supply the name of the configuration file:

```
shell> ./updateCDC.sh sample.conf
```

9.24. The vmrr Command

Chapter 10. The tpm Deployment Command

tpm, or the Tungsten Package Manager, is a complete configuration, installation and deployment tool for Tungsten Replication. 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 Replication and the configuration, and optionally automatically starts the services on each host. **tpm** manages the entire deployment, configuration and upgrade procedure.
- **tpm INI** configuration — **tpm** uses an **INI** file to configure the service on the local host. The **INI** file must be created on each host that will run Tungsten Replication. **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 10.1, “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 Replication.
- 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 Replication. 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 Replication 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 Replication 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 Replication 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 10.5, “tpm Commands”](#); configuration options, which determine the parameters that configure individual services, which are detailed in [Section 10.7, “tpm Configuration Options”](#); and the options that alter the way **tpm** operates, covered in [Section 10.3, “tpm Staging Configuration”](#).

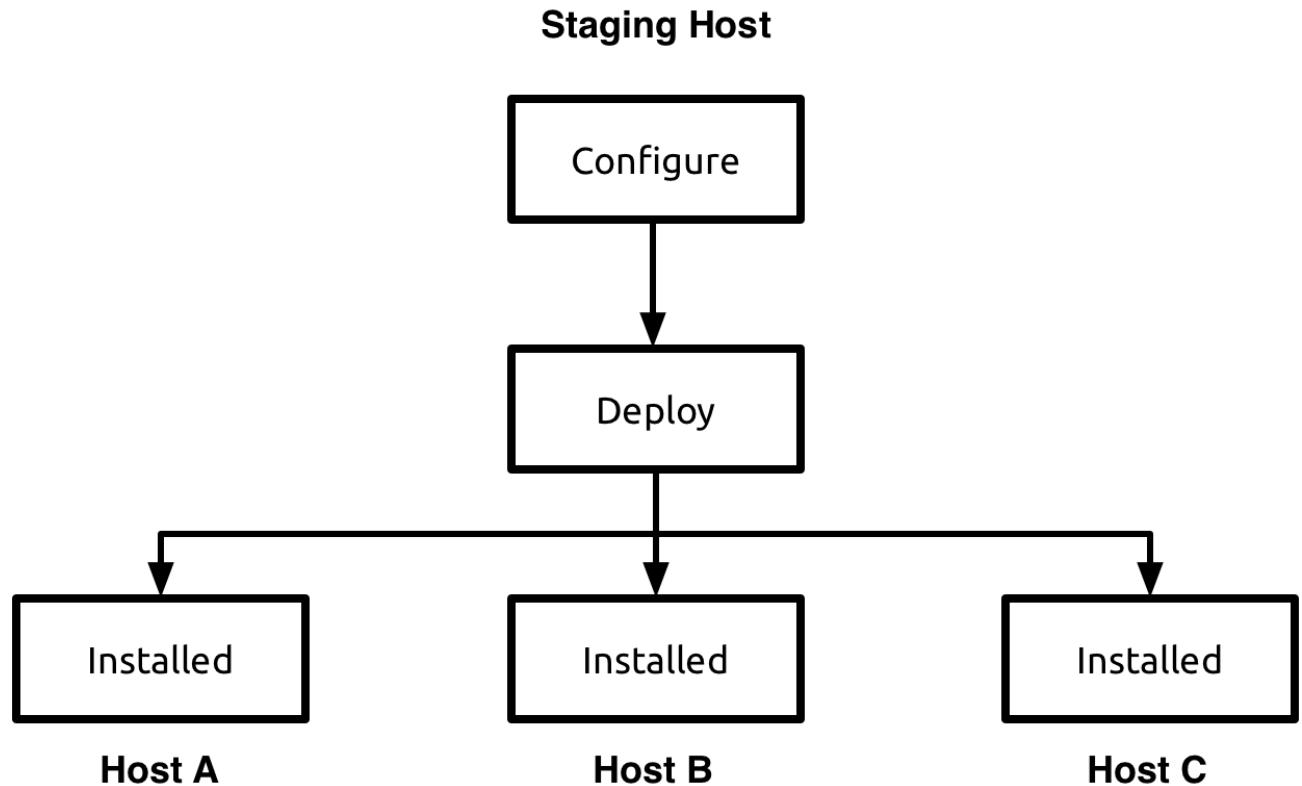
10.1. Comparing Staging and `ini` `tpm` Methods

`tpm` supports two different deployment methodologies. Both configure one or more Tungsten Replication 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 Replication 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 10.1. `tpm` Staging Based Deployment



- `ini` File

When using the `ini` file method, configuration for each service must be made individually using an `ini` configuration file on each host. This is ideal for deployments where you have a configuration management system (e.g. Puppet and Chef) to manage the `ini` file. It also works very well for deployments where the configuration for each system is different from the others.

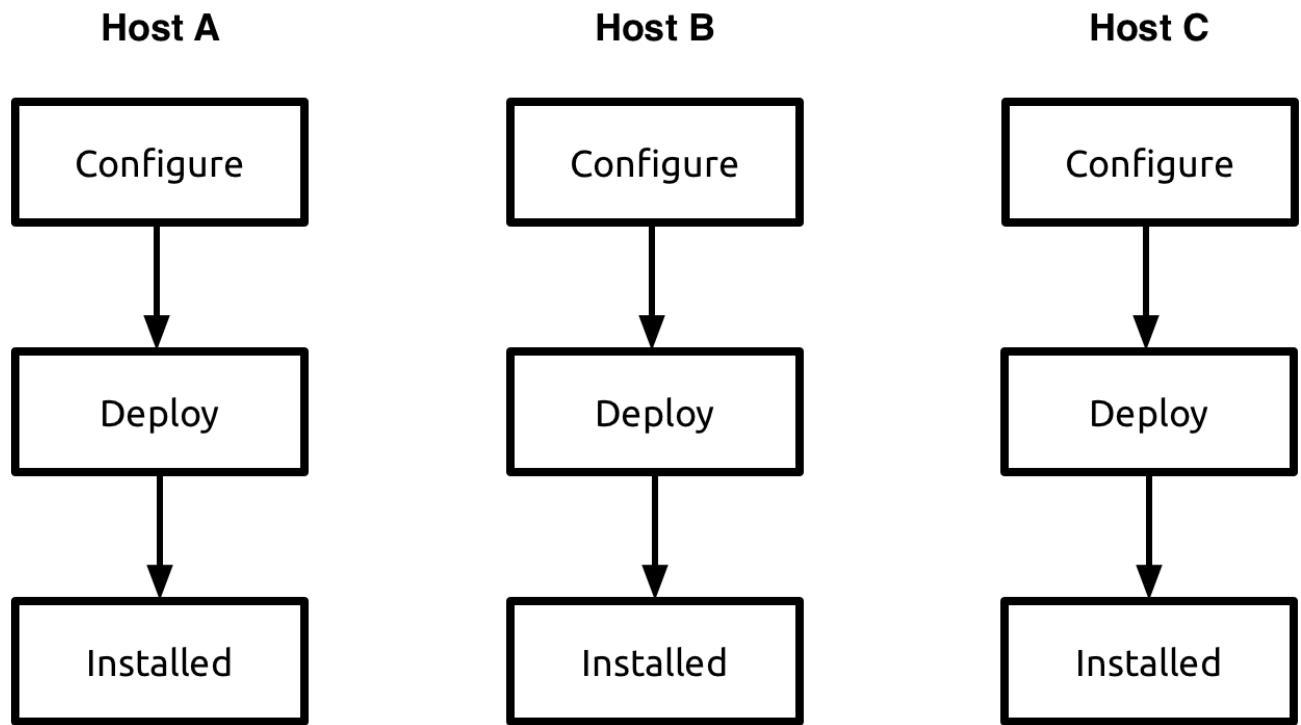
Figure 10.2. `tpm` INI Based Deployment

Table 10.1. TPM Deployment Methods

Feature	Staging Directory	INI File
Deploy Multiple Services	Yes	Yes
Deploy to Multiple Hosts	Yes	No
Individual Host-based Configuration	Yes	Yes
Single-Step Upgrade	Yes	No
Requires SSH Configuration	Yes	No
RPM/PKG Support	Yes	Yes

Note

Check the output of `tpm query staging` to determine which method your current installation uses. The output for an installation from a staging directory will start with `# Installed from tungsten@staging-host:/opt/continuent/software/tungsten-replicator-5.0.1-136`. 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 Replication 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 Replication.
 - b. Complete all the [Appendix B, Prerequisites](#).

- c. Create the `INI` file containing your configuration.
- d. Execute the `tpm install` command to deploy the service.

When using the staging method, upgrades and updates to the configuration must be made using `tpm` from the staging directory. Configuration methods can be swapped from staging to `INI` only by manually recreating the `INI` file with the new configuration and running `tpm update`.

10.2. 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 Replication and deployment files to each server

During this stage part of the Tungsten Replication 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 Replication and write configuration files

If validation is successful, we will move on to deploying Tungsten Replication and writing the actual configuration files. The `tpm` command uses a JSON file that summarizes the configuration. The Tungsten Replication 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 Replication 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 Replication 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 Replication services

After Tungsten Replication 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` [365] or `--start-and-report` [365] were provided in the configuration
2. Start the Tungsten Replicator and Tungsten Manager on all hosts
3. Wait for the Tungsten Manager to become responsive
4. Start the Tungsten Connector on all hosts

- **Upgrade**

1. Put all dataservices into MAINTENANCE mode
2. Stop the Tungsten Replicator on all nodes

10.3. **tpm** Staging Configuration

Before installing your hosts, you must provide the desired configuration. This will be done with one or more calls to **tpm configure** as seen in the [Chapter 2, Deployment](#). These calls place the given parameters into a staging configuration file that will be used during installation. This is done for dataservices, composite dataservices and replication services.

Instead of a subcommand, **tpm configure** accepts a service name or the word `defaults` as a subcommand. This identifies what you are configuring.

```
shell> tpm configure [service_name|defaults] [tpm options] [service configuration options]
```

In addition to the [Section 10.7, “tpm Configuration Options”](#), the common options in [Table 10.4, “tpm Common Options”](#) may be given.

The **tpm** command will store the staging configuration in the staging directory that you run it from. This behavior is changed if you have `$CONTINUENT_PROFILES` or `$REPLICATOR_PROFILES` defined in the environment. If present, **tpm** will store the staging configuration in that directory. Doing this will allow you to upgrade to a new version of the software without having to run the **tpm fetch** command.

If you are running Tungsten Replicator, the **tpm** command will use `$REPLICATOR_PROFILES` if it is available, before using `$CONTINUENT_PROFILES`.

10.3.1. 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**.

10.3.2. Configuring a single service

```
shell> ./tools/tpm configure alpha \
--master=host1 \
--members=host1,host2,host3 \
--home-directory=/opt/continuent \
--user=tungsten
```

The configuration options provided following the service name will be associated with the 'alpha' dataservice. These options will override any given with **tpm configure defaults**.

Relationship of `--members` [351], `--slaves` [365] and `--master` [351]

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` [351] and `--slaves` [365]; `--members` [351] will be calculated as the unique join of both values.

If you specify `--master` [351] and `--members` [351]; `--slaves` [365] will be calculated as the unique difference of both values.

10.3.3. Configuring a single host

```
shell> ./tools/tpm configure alpha --hosts=host3 \
    --backup-method=xtrabackup-incremental
```

This will apply the `--repl-backup-method` [330] option to just the host3 server. Multiple hosts may be given as a comma-separated list. The names used in the `--members` [351], `--slaves` [365], `--master` [351], options should be used when calling `--hosts` [347]. These values will override any given in `tpm configure defaults` or `tpm configure alpha`.

10.3.4. Reviewing the current configuration

You may run the `tpm reverse` command to review the list of configuration options. This will run in the staging directory and in your installation directory. It is a good idea to run this command prior to installation and upgrades to validate the current settings.

```
# Installed from tungsten@host1:/home/tungsten/tungsten-replicator-5.0.1-136
# Options for the alpha data service
tools/tpm configure alpha \
--enable-thl-ssl=true \
--install-directory=/opt/continuent \
--java-keystore-password=password \
--java-truststore-password=password \
--master=host1 \
--members=host1,host2,host3 \
--replication-password=password \
--replication-user=tungsten \
--start=true \
--topology=master-slave
```

The output includes all of the `tpm configure` commands necessary to rebuild the configuration. It includes all default, dataservice and host specific configuration settings. Review this output and make changes as needed until you are satisfied.

10.3.5. Installation

After you have prepared the configuration file, it is time to install.

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

This will install all services defined in configuration. The installation will be done as explained in Section 10.2, “Processing Installs and Upgrades”. This will include the full set of `--members` [351], `--slaves` [365], and `--master` [351].

10.3.5.1. Installing a set of specific services

```
shell> ./tools/tpm install alpha,bravo
```

All hosts included in the alpha and bravo services will be installed. The installation will be done as explained in Section 10.2, “Processing Installs and Upgrades”.

10.3.5.2. Installing a set of specific hosts

```
shell> ./tools/tpm install --hosts=host1,host2
```

Only `host1` and `host2` will be installed. The installation will be done as explained in Section 10.2, “Processing Installs and Upgrades”.

10.3.6. Upgrades from a Staging Directory

This process must be run from the staging directory in order to run properly. Determine where the current software was installed from.

```
shell> tpm query staging
tungsten@staging-host:/opt/continuent/software/continuent-tungsten-2.0.3-519
```

This outputs the hostname and directory where the software was installed from. Make your way to that host and the parent directory before proceeding. Unpack the new software into the `/opt/continuent/software` directory and make it your current directory.

```
shell> tar zxf tungsten-replicator-5.0.1-136.tar.gz
shell> cd tungsten-replicator-5.0.1-136
```

Before any update, the current configuration must be known. If the `$CONTINUENT_PROFILES` or `$REPLICATOR_PROFILES` environment variables were used in the original deployment, these can be set to the directory location where the configuration was stored.

Alternatively, the update can be performed by fetching the existing configuration from the deployed directory by using the `tpm fetch` command:

```
shell> ./tools/tpm fetch --reset --directory=/opt/continuent \
    --hosts=host1,autodetect
```

This will load the configuration into the local staging directory. Review the current configuration before making any configuration changes or deploying the new software.

```
shell> ./tools/tpm reverse
```

This will output the current configuration of all services defined in the staging directory. You can then make changes using [tpm configure](#) before pushing out the upgrade. Run [tpm reverse](#) again before [tpm update](#) to confirm your changes were loaded correctly.

```
shell> ./tools/tpm configure service_name ...
shell> ./tools/tpm update
```

This will update the configuration file and then push the updates to all hosts. No additional arguments are needed for the [tpm update](#) command since the configuration has already been loaded.

10.3.7. Configuration Changes from a Staging Directory

Where, and how, you make configuration changes depends on where you want the changes to be applied.

Making Configuration Changes to the Current Host

You may make changes to a specific host from the [/opt/continuent/tungsten](#) directory.

```
shell> ./tools/tpm update service_name --th1-log-retention=14d
```

This will update the local configuration with the new settings and restart the replicator. You can use the [tpm help update](#) command to see which components will be restarted.

```
shell> ./tools/tpm help update | grep th1-log-retention
--th1-log-retention How long do you want to keep TH1 files?
```

If you make changes in this way then you must be sure to run [tpm fetch](#) from your staging directory prior to any further changes. Skipping this step may result in you pushing an old configuration from the staging directory.

Making Configuration Changes to all hosts

This process must be run from the staging directory in order to run properly. Determine where the current software was installed from.

```
shell> tpm query staging
tungsten@staging-host:/opt/continuent/software/continuent-tungsten-2.0.3-519
```

This outputs the hostname and directory where the software was installed from. Make your way to that host and directory before proceeding.

```
shell> ./tools/tpm fetch --reset --directory=/opt/continuent \
    --hosts=host1,autodetect
```

This will load the configuration into the local staging directory. Review the current configuration before making any configuration changes or deploying the new software.

```
shell> ./tools/tpm reverse
```

This will output the current configuration of all services defined in the staging directory. You can then make changes using [tpm configure](#) before pushing out the upgrade. Run [tpm reverse](#) again before [tpm update](#) to confirm your changes were loaded correctly.

```
shell> ./tools/tpm configure service_name ...
shell> ./tools/tpm update
```

This will update the configuration file and then push the updates to all hosts. No additional arguments are needed for the [tpm update](#) command since the configuration has already been loaded.

10.3.8. Converting from INI to Staging

If you currently use the INI installation method and wish to convert to using the Staging method, there is currently no easy way to do that. The procedure involves uninstalling fully on each node, then reinstalling from scratch.

If you still wish to convert from the INI installation method to using the Staging method, use the following procedure:

1. On the staging node, extract the software into [/opt/continuent/software/{extracted_dir}](#)

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-replicator-5.0.1-136.tar.gz
```

2. Create the text file `config.sh` based on the output from `tpm reverse`:

```
shell> cd tungsten-replicator-5.0.1-136
shell> tpm reverse > config.sh
```

Review the new `config.sh` script to confirm everything is correct, making any needed edits. When ready, create the new configuration:

```
shell> sh config.sh
```

Review the new configuration:

```
shell> tools/tpm reverse
```

See [Section 10.3, “`tpm` Staging Configuration”](#) for more information.

3. On all nodes, uninstall the Tungsten software:

Warning

Executing this step WILL cause an interruption of service.

```
shell> tpm uninstall --i-am-sure
```

4. On all nodes, rename the `tungsten.ini` file:

```
shell> mv /etc/tungsten/tungsten.ini /etc/tungsten/tungsten.ini.old
```

5. On the staging node only, change to the extracted directory and execute the `tpm install` command:

```
shell> cd /opt/continuent/software/tungsten-replicator-5.0.1-136
shell> ./tools/tpm install
```

10.4. `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.

10.4.1. Creating an INI file

When using an INI configuration, installation and updates will still be done using the `tpm` command. Instead of providing configuration information on the command line, the `tpm` command will look for an INI file at `/etc/tungsten.ini` or `/etc/tungsten/tungsten.ini`. The file must be readable by the `tungsten` system user.

In Tungsten Replicator 3.0 and later, `tpm` will automatically search all INI files within the `/etc/tungsten` and current directories. An alternative directory can be searched using `--ini` option to `tpm`.

Here is an example of a `tungsten.ini` file that would setup a simple dataservice.

```
[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

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

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.

10.4.2. Installation with INI File

Once you have created the `tungsten.ini` file, the `tpm` command will recognize it and use it for configuration. Unpack the software into `/opt/continuent/software` and run the `tpm install` command.

```
shell> cd /opt/continuent/software/tungsten-replicator-5.0.1-136
shell> ./tools/tpm install
```

The `tpm` command will read the `tungsten.ini` file and setup all dataservices on the current server.

10.4.3. Upgrades with an INI File

Use the `tpm update` command to upgrade to the latest version.

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-replicator-5.0.1-136.tar.gz
shell> cd tungsten-replicator-5.0.1-136
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.

10.4.4. Configuration Changes with an INI file

The `tpm update` also allows you to apply any configuration changes. Start by making any necessary changes to the `tungsten.ini` file. Then proceed to running `tpm update`.

```
shell> cd /opt/continuent/tungsten
shell> ./tools/tpm update
```

This will read the `tungsten.ini` file and apply the settings. The `tpm` command will identify what services likely need to be restarted and will just restart those. You can manually restart the desired services if you are unsure if the new configuration has been applied.

10.4.5. 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.

1. Create the text file `/etc/tungsten/tungsten.ini` on each node. They will normally all be the same.

```
shell> sudo mkdir /etc/tungsten
shell> sudo chown -R tungsten: /etc/tungsten
shell> chmod 700 /etc/tungsten
shell> touch /etc/tungsten/tungsten.ini
shell> chmod 600 /etc/tungsten/tungsten.ini
```

Each section in the INI file replaces a single `tpm configure` call. The section name inside of [square brackets] is used as the service name. In the case of the [defaults] section, this will act like the `tpm configure defaults` command. The property names in the INI file are the same as what is used on the command line. Simply remove the leading -- characters and add it to the proper section.

For example, to seed the `tungsten.ini` file, use the output of `tpm reverse`:

```
shell> tpm reverse > /etc/tungsten/tungsten.ini
```

Edit the new ini file and clean it up as per the rules above. For example, using vim:

```
shell> vim /etc/tungsten/tungsten.ini
:is/tools\[^tpm configure \]/g
:%s/^---//g
:%s/\s*\$\$/g
```

Important

In the above example, you MUST manually add the trailing square bracket] to the end of the defaults tag and to the end of every service name section. Just search for the opening square bracket [and make sure there is a matching closing square bracket for every one.

See [Section 10.4.1, “Creating an INI file”](#) for more information.

2. On every node, extract the software into `/opt/continuent/software/{extracted_dir}`

Warning

Make sure you have the same release that is currently installed.

```
shell> cd /opt/continuent/software  
shell> tar zxf tungsten-replicator-5.0.1-136.tar.gz
```

3. On each node, change to the extracted directory and execute the **tpm** command:

```
shell> cd /opt/continuent/software/tungsten-replicator-5.0.1-136  
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.

10.5. **tpm** Commands

All calls to **tpm** will follow a similar structure, made up of the **command**, which defines the type of operation, and one or more options.

```
shell> tpm command [sub command] [tpm options] [command options]
```

The command options will vary for each command. The core **tpm** options are:

Table 10.2. **tpm** Core Options

Option	Description
--force [346], -f [346]	Do not display confirmation prompts or stop the configure process for errors
--help [346], -h [346]	Displays help message
--info [347], -i [347]	Display info, notice, warning and error messages
--notice [356], -n [356]	Display notice, warning and error messages
--preview [360], -p [360]	Displays the help message and preview the effect of the command line options
--profile file [361]	Sets name of config file
--quiet [362], -q [362]	Only display warning and error messages
--verbose [370], -v [370]	Display debug, info, notice, warning and error messages

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 [347]` 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 [351]`, `--slaves [365]`, `--master [351]` arguments should be used when calling `--hosts [347]`.
2. When you are using an INI configuration file (see [Section 10.4, “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 Replication will ultimately be placed but may be a sub-directory. In most cases we will install to `/opt/continuent` but use `/opt/continuent/software` as a staging directory. The release package should be unpacked in the staging directory before proceeding. See the [Section B.2, “Staging Host Configuration”](#) for instructions on selecting a staging directory.

Table 10.3. **tpm** Commands

Option	Description
<code>configure</code>	Configure a data service within the global configuration
<code>diag</code>	Obtain diagnostic information
<code>fetch</code>	Fetch configuration information from a running service

Option	Description
<code>firewall</code>	Display firewall information for the configured services
<code>help</code>	Show command help information
<code>install</code>	Install a data service based on the existing and runtime parameters
<code>mysql</code>	Open a connection to the configured MySQL server
<code>query</code>	Query the active configuration for information
<code>reset</code>	Reset the cluster on each host
<code>reset-thl</code>	Reset the THL for a host
<code>restart</code>	Restart the services on specified or added hosts
<code>ssh-copy-cert</code>	Executes the commands required to generate authorized SSH certificate and keys required for tpm
<code>start</code>	Start services on specified or added hosts
<code>stop</code>	Stop services on specified or added hosts
<code>upgrade, update</code>	Update an existing configuration or software version
<code>validate</code>	Validate the current configuration
<code>validate-update</code>	Validate the current configuration and update

10.5.1. **tpm configure** Command

The **configure** command to **tpm** creates a configuration file within the current profiles directory

10.5.2. **tpm diag** Command

The **tpm diag** command will create a ZIP file including log files and current dataservice status. It will connect to all servers listed in the **tpm reverse** output attempting to collect information.

```
shell> tpm diag
NOTE  >> host1 >> Diagnostic information written to /home/tungsten/tungsten-diag-2013-10-09-21-04-23.zip
```

The information collected depends on the installation type:

- Within a staging directory installation, all the hosts configured within the cluster will be contacted, and all the information across all hosts will be incorporated into the Zip file that is created.
- Within an INI installation, the other hosts in the cluster will be contacted if **ssh** has been configured and the other hosts can be reached. If **ssh** is not available, a warning will be printed, and each host will need to be accessed individually to run **tpm diag**.

The structure of the created file will depend on the configured hosts, but will include all the logs for each accessible host configured. For example:

```
Archive: tungsten-diag-2013-10-17-15-37-56.zip  22465 bytes  13 files
drwxr-xr-x  5.2 unx      0 t- defN 17-Oct-13 15:37 tungsten-diag-2013-10-17-15-37-56/
drwxr-xr-x  5.2 unx      0 t- defN 17-Oct-13 15:37 tungsten-diag-2013-10-17-15-37-56/host1/
-rw-r--r--  5.2 unx     80 t- defN 17-Oct-13 15:37 tungsten-diag-2013-10-17-15-37-56/host1/thl.txt
-rw-r--r--  5.2 unx   1428 t- defN 17-Oct-13 15:37 tungsten-diag-2013-10-17-15-37-56/host1/trepctl.txt
-rw-r--r--  5.2 unx  106415 t- defN 17-Oct-13 15:37 tungsten-diag-2013-10-17-15-37-56/host1/trepsvc.log
drwxr-xr-x  5.2 unx      0 t- defN 17-Oct-13 15:37 tungsten-diag-2013-10-17-15-37-56/host2/
-rw-r--r--  5.2 unx     82 t- defN 17-Oct-13 15:37 tungsten-diag-2013-10-17-15-37-56/host2/thl.txt
-rw-r--r--  5.2 unx   1365 t- defN 17-Oct-13 15:37 tungsten-diag-2013-10-17-15-37-56/host2/trepctl.txt
-rw-r--r--  5.2 unx  44128 t- defN 17-Oct-13 15:37 tungsten-diag-2013-10-17-15-37-56/host2/trepsvc.log
drwxr-xr-x  5.2 unx      0 t- defN 17-Oct-13 15:37 tungsten-diag-2013-10-17-15-37-56/host3/
-rw-r--r--  5.2 unx     82 t- defN 17-Oct-13 15:37 tungsten-diag-2013-10-17-15-37-56/host3/thl.txt
-rw-r--r--  5.2 unx   1365 t- defN 17-Oct-13 15:37 tungsten-diag-2013-10-17-15-37-56/host3/trepctl.txt
-rw-r--r--  5.2 unx  44156 t- defN 17-Oct-13 15:37 tungsten-diag-2013-10-17-15-37-56/host3/trepsvc.log
```

10.5.3. **tpm fetch** Command

There are some cases where you would like to review the configuration or make changes prior to the upgrade. In these cases it is possible to fetch the configuration and process the upgrade as different steps.

```
shell> ./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** [347]

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** [370]

The username to be used when logging in to other hosts.

- **--directory**

The installation directory of the current Tungsten Replication installation. If **autodetect** is specified, then **tpm** will look for the installation directory by checking any running Tungsten Replication processes.

10.5.4. **tpm firewall** Command

The **tpm firewall** command displays port information required to configured a firewall. When used, the information shown is for the current host:

```
shell> tpm firewall
To host1
-----
From application servers
From connector servers      13306
From database servers       2112, 13306
```

The information shows which ports, on which hosts, should be opened to enable communication.

10.5.5. **tpm help** Command

The **tpm help** command outputs the help information for **tpm** showing the list of supported commands and options.

```
shell> tpm help
Usage: tpm help [commands,config-file,template-file] [general-options] [command-options]
-----
General options:
-f, --force          Do not display confirmation prompts or stop the configure »
                     process for errors
-h, --help            Displays help message
--profile file       Sets name of config file (default: tungsten.cfg)
-p, --preview         Displays the help message and preview the effect of the »
                     command line options
-q, --quiet           Only display warning and error messages
-n, --notice          Display notice, warning and error messages
-i, --info             Display info, notice, warning and error messages
-v, --verbose          Display debug, info, notice, warning and error messages
...
```

To get a list of available configuration options, use the **config-file** subcommand:

```
shell> tpm help config-file
#####
# Config File Options
#####
config_target_basename      [tungsten-replicator-5.0.1-136_pid10926]
deployment_command           Current command being run
remote_package_path          Path on the server to use for running tpm commands
deploy_current_package       Deploy the current Tungsten package
deploy_package_uri            URL for the Tungsten package to deploy
deployment_host               Host alias for the host to be deployed here
staging_host                 Host being used to install
...
```

10.5.6. **tpm install** Command

The **tpm install** command performs an installation based on the current configuration (if one has been previously created), or using the configuration information provided on the command-line.

For example:

```
shell> ./tools/tpm install alpha \
    --topology=master-slave \
    --master=host1 \
    --replication-user=tungsten \
    --replication-password=password \
    --home-directory=/opt/continuent \
    --members=host1,host2,host3 \
    --start
```

Installs a service using the command-line configuration.

```
shell> ./tools/tpm configure alpha \
    --topology=master-slave \
    --master=host1 \
    --replication-user=tungsten \
    --replication-password=password \
    --home-directory=/opt/continuent \
    --members=host1,host2,host3
shell> ./tools/tpm install alpha
```

Configures the service first, then performs the installation steps.

During installation, **tpm** checks for any host configuration problems and issues, copies the Tungsten Replication 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.

10.5.7. **tpm mysql** Command

This will open a MySQL CLI connection to the local MySQL server using the current values for [--replication-user \[363\]](#), [--replication-password \[363\]](#) and [--replication-port \[363\]](#).

```
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.

10.5.8. **tpm query** Command

The **query** command provides information about the current **tpm** installation. There are a number of subcommands to query specific information:

- **tpm query config** — return the full configuration values
- **tpm query dataservices** — return the list of dataservices
- **tpm query default** — return the list of configured default values
- **tpm query deployments** — return the configuration of all deployed hosts
- **tpm query manifest** — get the manifest information
- **tpm query modified-files** — return the list of files modified since installation by **tpm**
- **tpm query staging** — return the staging directory from where Tungsten Replication was installed
- **tpm query values** — return the list of configured values
- **tpm query version** — get the version of the current installation

10.5.8.1. **tpm query config**

Returns a list of all of the configuration values, both user-specified and implied within the current configuration. The information is returned in the form a JSON value:

```
shell> tpm query config
{
  "__system_defaults_will_be_overwritten__": {
  ...
  "staging_directory": "/home/tungsten/tungsten-replicator-5.0.1-136",
  "staging_host": "tr-ms1",
  "staging_user": "tungsten"
}
```

10.5.8.2. **tpm query dataservices**

Returns the list of configured dataservices that have, or will be, installed:

```
shell> tpm query dataservices
alpha      : PHYSICAL
```

10.5.8.3. **tpm query deployments**

Returns a list of all the individual deployment hosts and configuration information, returned in the form of a JSON object for each installation host:

```
shell> tpm query deployments
{
  "config_target_basename": "tungsten-replicator-5.0.1-136_pid22729",
  "dataservice_host_options": {
    "alpha": {
      "start": "true"
    }
  ...
  "staging_directory": "/home/tungsten/tungsten-replicator-5.0.1-136",
  "staging_host": "tr-ms1",
  "staging_user": "tungsten"
}
```

10.5.8.4. **tpm query manifest**

Returns the manifest information for the identified release of Tungsten Replication, including the build, source and component versions, returned in the form of a JSON value:

```
shell> tpm query manifest
{
  "SVN": {
    "bristlecone": {
      "URL": "http://bristlecone.googlecode.com/svn/trunk/bristlecone",
      "revision": 170
    },
    "commons": {
      "URL": "https://tungsten-replicator.googlecode.com/svn/trunk/commons",
      "revision": 1983
    },
    "cookbook": {
      "URL": "https://tungsten-toolbox.googlecode.com/svn/trunk/cookbook",
      "revision": 230
    },
    "replicator": {
      "URL": "https://tungsten-replicator.googlecode.com/svn/trunk/replicator",
      "revision": 1983
    }
  },
  "date": "Wed Jan 8 18:11:08 UTC 2014",
  "host": "ip-10-250-35-16",
  "hudson": {
    "SVNRevision": null,
    "URL": "http://cc.aws.continuent.com/",
    "buildId": 28,
    "buildNumber": 28,
    "buildTag": "jenkins-Base_Replicator_JUnit-28",
    "jobName": "Base_Replicator_JUnit"
  },
  "product": "Tungsten Replicator",
  "userAccount": "jenkins",
  "version": {
    "major": 2,
    "minor": 2,
    "revision": 1
  }
}
```

10.5.8.5. **tpm query modified-files**

Shows the list of configuration files that have been modified since the installation was completed. Modified configuration files cannot be overwritten during an upgrade process, using this command enables you identify which files contain changes so that these modifications can be manually migrated to the new installation. To restore or replace files with their original installation, copy the `.filename.orig` file.

10.5.8.6. **tpm query staging**

Returns the host and directory from which the current installation was created:

```
shell> tpm query staging
tungsten@host1:/home/tungsten/tungsten-replicator-5.0.1-136
```

This can be useful when the installation host and directory from which the original configuration was made need to be updated or modified.

10.5.8.7. **tpm query version**

Returns the version for the identified version of Tungsten Replication:

```
shell> tpm query version
5.0.1-136
```

10.5.9. **tpm reset** Command

This command will clear the current state for all Tungsten services:

- Management metadata
- Replication metadata
- THL files
- Relay log files
- Replication position

If you run the command from an installed directory, it will only apply to the current server. If you run it from a staging directory, it will apply to all servers unless you specify the [--hosts \[347\]](#) option.

```
shell> ./tools/tpm reset
```

10.5.10. **tpm reset-thl** Command

This command will clear the current replication state for the Tungsten Replicator:

- THL files
- Relay log files
- Replication position

If you run the command from an installed directory, it will only apply to the current server. If you run it from a staging directory, it will apply to all servers unless you specify the [--hosts \[347\]](#) option.

```
shell> ./tools/tpm reset-thl
```

10.5.11. **tpm restart** Command

The **tpm restart** command contacts the currently configured services on the current host and restarts each service. On a running system this will result in an interruption to service as the services are restarted.

The **restart** command can be useful in situations where services may not have started properly, or after a reboot services failed. For more information on explicitly starting components, see [Section 2.6, “Starting and Stopping Tungsten Replicator”](#). For information on how to configure services to start during a reboot, see [Section 2.7, “Configuring Startup on Boot”](#).

10.5.12. **tpm reverse** Command

The **tpm reverse** command will show you the commands required to rebuild the configuration for the current directory. This is useful for doing an upgrade or when copying the deployment to another server.

```
shell> ./tools/tpm reverse
# Defaults for all data services and hosts
tools/tpm configure defaults \
--application-password=secret \
--application-port=3306 \
--application-user=app \
```

```
--replication-password=secret \
--replication-port=13306 \
--replication-user=tungsten \
--start-and-report=true \
--user=tungsten
# Options for the alpha data service
tools/tpm configure alpha \
--connectors=host1,host2,host3 \
--master=host1 \
--members=host1,host2,host3
```

The **tpm reverse** command supports the following arguments:

- **--public**

Hide passwords in the command output

- **--ini-format**

Display output in ini format for use in </etc/tungsten/tungsten.ini> and similar configuration files

10.5.13. **tpm ssh-copy-cert** Command

The **tpm ssh-copy-cert** command executes all the required commands to generate the required **ssh** certificates required for SSH operation by **tpm**. Executing the command should generate the required directory, certificate and add that information to the required SSH files, then ensure that the directory permissions and ownership on [~/.ssh](#) are set correctly.

For example, executing the command outputs the stages and progress:

```
shell> ./tools/tpm ssh-copy-cert
mkdir -p ~/.ssh
echo "-----BEGIN RSA PRIVATE KEY-----"
MIEowIBAAKCAQEAnMSRTwB2Ik6FOT2YXQkXg1FivniLSRxcNw73UDVEGxPtsdN
p5qzXH+ktSlyFHIPHkjh8jEnoWpzjpUmrhgqUUYg6zssxL515w8UK5NUDmWRxV
lAE0uJ2TyNm8uAVWGwokPHmg0zOsYjg314UAwx6WhFtiiKtf96jLAQfethTQU
eRKZjIC17Hm2GLXNutzfgfTgKKWsfrLQjm4WZEHqmZCBy3fRjnAnyeJPJcr8gPPl
ato000mJ66rdUT53TN91FwEWwC+vIacypKyFkbqwFHDC60Vb0kMaQd/T4Y35E7s
wfEOmrjmSgs70/a1NuSjrs5CcgeezOx1NN8nQIDAQABAoIBAHx+idrQHjpmd+6R
0qUhIMRg3o5AZUJuN3xmGVBapR12ulMvsVaRvzCM2XSjQ2pDLgbxhAQ/yN1qgUTp
KD1gUZgbmrViCaKe52RpTf36e/Pnw1Yv7zIrRv/5e5w813B3Tdw7gHc1YVTL/bZ0
WLqvBmI93j8eJHBtN1OIrV+jGYmIdlHjb+I2VcpQMfbAgxZVDNy1OMe7+YZk0hj3
414etqTgUMOnF/tkw81uPbfUGV0nM9a8er4wJLxbjbF7Y00jGOOSFhWNgrrMMCrKz
gy0gW6pWYAh2iId095Q/Lgt3Yk77Dld8By6tgHa74IZwgUQb/iCTcbTaPHRErXL
vfhuQtUcgYEAzCX7VQMc2YJh0j/OEOBwMmIzccCIC1GuFI0kqaNauCm8aL/ydUdR
chZGzbWzIMd6VJ3ud7rwevFzymgGcyrmRig98D56TkCOHN+UmMO30efzRGwEz5
FnwT2wM4P8bKcVKrotDae3uruEV6mAV2kGU8fnHqS0lNE0cGQW+sCgYEaxJXW
JrkZX4W8QtpIXZcywXem9SnOK6Q2RxOcStFspbxKPz62730E1RpeIiz76Wm33s81
06dkVWrhSKh7K1IXte4Koq0jJ2S2gCc4cqxxuS0na+H90xSHIsrgUp1tmeUr05
X9zqfgw041665/cKV8BmuzqXZW9+QryJBCtVIsCgYPAiBtym9Vlxlg1QnYDv0UI
IiEJVE14sYMX8uVzTR56J3q8AOkolgR8iZDHQs1OoH9yf0g3Zpb3fA00nY4JbtN
VP8UotnoRNQbzOOrfvDxYOAkaw7BdQhcsd77pQONxzy1u+v5uujzLL16/g/DJN8b
sqFp/o3B16PoxjYpsJAA3Q0CgYBxeBs4FrcUjAjxMSNpMhC14x6XfB3oyswZkpQu
uVc5Gsmw76v1XWom601d10jiv/V8V5Y2KPSc6Shq9GaKd9uyAsnmpFD/kaL1+lyT
Z6/dob0vF1YM+Xus2VoWJizUoqBMFDj3vIeTYfbTmUPBCLMSiMdt9T/V40kKhypq
7raXqQKBgGrBGo/FoUduJFfadVwr66vsg1b+3q/GX4adnL3bn1C7QxigzXHPH1vf9
z2c/P9Tw8M41JX2hOKCyGgxIbZ+fNP0sB8prdhbc/Zl1d4tUcZFtSCAjk3pwDmm
2NDp3ddCh/scfm8o2dxblKFsjUtaBsk6ApN49AWa8W5GckKG+or
-----END RSA PRIVATE KEY----- > ~/.ssh/id_rsa
echo "ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCCxJFPAAHYiToU5NljfCRcEUWK+eItJHFw3DvdQNUQbE+2x02nmrNcf6S2yXIUcg8ceQmGzyMSehanO0lSau
touch ~/.ssh/authorized_keys
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 700 ~/.ssh
chmod 600 ~/.ssh/*
```

10.5.14. **tpm start** Command

The **tpm start** command starts configured services on the current host. This can be useful in situations where you have installed services but not configured them to be started.

```
shell> tpm start
...
Getting replication status on host1
Processing services command...
NAME          VALUE
----          -----
appliedLastSeqno: 610
appliedLatency : 0.95
role          : master
```

```
serviceName      : alpha
serviceType     : local
started        : true
state          : ONLINE
Finished services command...

NOTE  >> tr_ss11 >> Command successfully completed
```

The **tpm start** can also be provided with the name of a service, which will start all the processes for that service on the current host.

See also the **tpm restart** command, [Section 2.6, “Starting and Stopping Tungsten Replicator”](#), and [Section 2.7, “Configuring Startup on Boot”](#).

10.5.15. **tpm stop** Command

The **tpm stop** command contacts all configured services on the current host and stops them if they are running.

```
shell> tpm stop
NOTE  >> host1 >> Command successfully completed
```

See also the **tpm restart** command, [Section 2.6, “Starting and Stopping Tungsten Replicator”](#), and [Section 2.7, “Configuring Startup on Boot”](#).

10.5.16. **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 10.2, “Processing Installs and Upgrades”](#). The behavior of **tpm update** is dependent on two factors.

1. Are you upgrading to a new version or applying configuration changes to the current version?
2. The installation method used during deployment.

Note

Check the output of **tpm query staging** to determine which method your current installation uses. The output for an installation from a staging directory will start with `# Installed from tungsten@staging-host:/opt/continuous/software/tungsten-replicator-5.0.1-136`. 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 10.3.6, “Upgrades from a Staging Directory”](#).

If an INI file was used; see [Section 10.4.3, “Upgrades with an INI File”](#)

Applying configuration changes to the current version

If a staging directory was used; see [Section 10.3.7, “Configuration Changes from a Staging Directory”](#).

If an INI file was used; see [Section 10.4.4, “Configuration Changes with an INI file”](#).

10.5.17. **tpm validate** Command

The **tpm validate** command validates the current configuration before installation. The validation checks all prerequisites that apply before an installation, and assumes that the configured hosts are currently not configured for any Tungsten services, and no Tungsten services are currently running.

```
shell> ./tools/tpm validate
.....
...
#####
# Validation failed
#####
...
```

The command can be run after performing a **tpm configure** and before a **tpm install** to ensure that any prerequisite or configuration issues are addressed before installation occurs.

10.5.18. **tpm validate-update** Command

The **tpm validate-update** command checks whether the configured hosts are ready to be updated. By checking the prerequisites and configuration of the dataserver and hosts, the same checks as made by **tpm** during a **tpm install** operation. Since there may have been changes to the requirements or required configuration, this check can be useful before attempting an update.

Using **tpm validate-update** is different from **tpm validate** in that it checks the environment based on the updated configuration, including the status of any existing services.

```
shell> ./tools/tpm validate-update
....
WARN >> host1 >> The process limit is set to 7812, we suggest a value»
of at least 8096. Add 'tungsten      -      nproc 8096' to your »
/etc/security/limits.conf and restart Tungsten processes. (ProcessLimitCheck)

WARN >> host2 >> The process limit is set to 7812, we suggest a value»
of at least 8096. Add 'tungsten      -      nproc 8096' to your »
/etc/security/limits.conf and restart Tungsten processes. (ProcessLimitCheck)

WARN >> host3 >> The process limit is set to 7812, we suggest a value »
of at least 8096. Add 'tungsten      -      nproc 8096' to your »
/etc/security/limits.conf and restart Tungsten processes. (ProcessLimitCheck)
.WARN >> host3 >> MyISAM tables exist within this instance - These »
tables are not crash safe and may lead to data loss in a failover »
(MySQLMyISAMCheck)

NOTE >> Command successfully completed
```

Any problems noted should be addressed before you perform the update using **tpm update**.

10.6. **tpm** Common Options

tpm accepts these options along with those in [Section 10.7, “**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 10.4. **tpm** Common Options

Option	Description
<code>--enable-validation-check String [345]</code>	Remove a corresponding --skip-validation-check argument
<code>--enable-validation-warnings String [346]</code>	Remove a corresponding --skip-validation-warnings argument
<code>--net-ssh-option=key=value</code>	Set the Net::SSH option for remote system calls
<code>--property=key~/match/replace/, --property=key+=value, --property=key=value</code>	Modify the value for key in any file that the configure script touches; key=value - Set key to value without evaluating template values or other rules; key+=value - Evaluate template values and then append value to the end of the line; key~/=match/replace/ - Evaluate template values then execute the specified Ruby regex with sub. For example --property=replicator.key~/=(.+)/somevalue,\1 will prepend 'somevalue' before the template value for 'replicator.key'
<code>--remove-property=key</code>	Remove a corresponding --property argument.
<code>--skip-validation-check String [365]</code>	Do not run the specified validation check. Validation checks are identified by the string included in the error they output.
<code>--skip-validation-warnings String [365]</code>	Do not display warnings for the specified validation check. Validation checks are identified by the string included in the warning they output.

10.7. **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-th1-log-retention=3d [369]`
- In an INI file, without the double-dash prefix, i.e. `repl-th1-log-retention=3d [369]`

A full list of all the available options supported is provided in [Table 10.5, “**tpm** Configuration Options”](#).

Table 10.5. **tpm** Configuration Options

INI File Option	CmdLine Option	Description
--allow-bidi-unsafe [327], --repl-allow-bidi-unsafe [327]	allow-bidi-unsafe [327], repl-allow-bidi-unsafe [327]	Allow unsafe SQL from remote service
--api [327], --repl-api [327]	api [327], repl-api [327]	Enable the replication API
--api-host [327], --repl-api-host [327]	api-host [327], repl-api-host [327]	Hostname that the replication API should listen on
--api-password [328], --repl-api-password [328]	api-password [328], repl-api-password [328]	HTTP basic auth password for the replication API
--api-port [328], --repl-api-port [328]	api-port [328], repl-api-port [328]	Port that the replication API should bind to
--api-user [328], --repl-api-user [328]	api-user [328], repl-api-user [328]	HTTP basic auth username for the replication API
--application-password [328], --connector-password [328]	application-password [328], connector-password [328]	Database password for the connector
--application-port [328], --connector-listen-port [328]	application-port [328], connector-listen-port [328]	Port for the connector to listen on
--application-readonly-port [328], --connector-readonly-listen-port [328]	application-readonly-port [328], connector-readonly-listen-port [328]	Port for the connector to listen for read-only connections on
--application-user [329], --connector-user [329]	application-user [329], connector-user [329]	Database username for the connector
--auto-enable [329], --repl-auto-enable [329]	auto-enable [329], repl-auto-enable [329]	Auto-enable services after start-up
--auto-recovery-delay-interval [329], --repl-auto-recovery-delay-interval [329]	auto-recovery-delay-interval [329], repl-auto-recovery-delay-interval [329]	Delay between going OFFLINE and attempting to go ONLINE
--auto-recovery-max-attempts [329], --repl-auto-recovery-max-attempts [329]	auto-recovery-max-attempts [329], repl-auto-recovery-max-attempts [329]	Maximum number of attempts at automatic recovery
--auto-recovery-reset-interval [329], --repl-auto-recovery-reset-interval [329]	auto-recovery-reset-interval [329], repl-auto-recovery-reset-interval [329]	Delay before autorecovery is deemed to have succeeded
--backup-directory [330], --repl-backup-directory [330]	backup-directory [330], repl-backup-directory [330]	Permanent backup storage directory
--backup-dump-directory [330], --repl-backup-dump-directory [330]	backup-dump-directory [330], repl-backup-dump-directory [330]	Backup temporary dump directory
--backup-method [330], --repl-backup-method [330]	backup-method [330], repl-backup-method [330]	Database backup method
--backup-online [330], --repl-backup-online [330]	backup-online [330], repl-backup-online [330]	Does the backup script support backing up a datasource while it is ONLINE
--backup-retention [330], --repl-backup-retention [330]	backup-retention [330], repl-backup-retention [330]	Number of backups to retain
--backup-script [331], --repl-backup-script [331]	backup-script [331], repl-backup-script [331]	What is the path to the backup script
--batch-enabled [331]	batch-enabled [331]	Should the replicator service use a batch applier
--batch-load-language [331]	batch-load-language [331]	Which script language to use for batch loading
--batch-load-template [331]	batch-load-template [331]	Value for the loadBatchTemplate property
--buffer-size [331], --repl-buffer-size [331]	buffer-size [331], repl-buffer-size [331]	Replicator queue size between stages (min 1)

INI File Option	CmdLine Option	Description
--channels [331], --repl-channels [331]	channels [331], repl-channels [331]	Number of replication channels to use for services
--cluster-slave-auto-recovery-delay-interval [331], --cluster-slave-repl-auto-recovery-delay-interval [331]	cluster-slave-auto-recovery-delay-interval [331], cluster-slave-repl-auto-recovery-delay-interval [331]	Default value for --auto-recovery-delay-interval when --topology=cluster-slave
--cluster-slave-auto-recovery-max-attempts [332], --cluster-slave-repl-auto-recovery-max-attempts [332]	cluster-slave-auto-recovery-max-attempts [332], cluster-slave-repl-auto-recovery-max-attempts [332]	Default value for --auto-recovery-max-attempts when --topology=cluster-slave
--cluster-slave-auto-recovery-reset-interval [332], --cluster-slave-repl-auto-recovery-reset-interval [332]	cluster-slave-auto-recovery-reset-interval [332], cluster-slave-repl-auto-recovery-reset-interval [332]	Default value for --auto-recovery-reset-interval when --topology=cluster-slave
--composite-datasources [332], --dataservice-composite-data-sources [332]	composite-datasources [332], dataservice-composite-data-sources [332]	Data services that should be added to this composite data service
--config-file-help [332]	config-file-help [332]	Display help information for content of the config file
--conn-java-enable-concurrent-gc [332]	conn-java-enable-concurrent-gc [332]	Connector Java uses concurrent garbage collection
--conn-java-mem-size [332]	conn-java-mem-size [332]	Connector Java heap memory size used to buffer data between clients and databases
--conn-round-robin-include-master [332]	conn-round-robin-include-master [332]	Should the Connector include the master in round-robin load balancing
--connector-affinity [333]	connector-affinity [333]	The default affinity for all connections
--connector-autoreconnect [333]	connector-autoreconnect [333]	Enable auto-reconnect in the connector
--connector-autoreconnect-killed-connections [333]	connector-autoreconnect-killed-connections [333]	Enable autoreconnect for connections killed within the connector
--connector-bridge-mode [333], --enable-connector-bridge-mode [333]	connector-bridge-mode [333], enable-connector-bridge-mode [333]	Enable the Tungsten Connector bridge mode
--connector-default-schema [333], --connector-forced-schema [333]	connector-default-schema [333], connector-forced-schema [333]	Default schema for the connector to use
--connector-delete-user-map [333]	connector-delete-user-map [333]	Overwrite an existing user.map file
--connector-disable-connection-warnings [334]	connector-disable-connection-warnings [334]	Hide Connector warnings in log files
--connector-disconnect-timeout [334]	connector-disconnect-timeout [334]	Time (in seconds) to wait for active connection to disconnect before forcing them closed [default: 5]
--connector-drop-after-max-connections [334]	connector-drop-after-max-connections [334]	Instantly drop connections that arrive after --connector-max-connections has been reached
--connector-listen-interface [334]	connector-listen-interface [334]	Listen interface to use for the connector
--connector-max-connections [334]	connector-max-connections [334]	The maximum number of connections the connector should allow at any time
--connector-max-slave-latency [334], --connector-max-applied-latency [334]	connector-max-applied-latency [334], connector-max-slave-latency [334]	The maximum applied latency for slave connections
--connector-readonly [334], --enable-connector-readonly [334]	connector-readonly [334], enable-connector-readonly [334]	Enable the Tungsten Connector read-only mode
--connector-ro-addresses [334]	connector-ro-addresses [334]	Connector addresses that should receive a r/o connection

INI File Option	CmdLine Option	Description
--connector-rw-addresses [335]	connector-rw-addresses [335]	Connector addresses that should receive a r/w connection
--connector-rwsplitting [335]	connector-rwsplitting [335]	Enable DirectReads R/W splitting in the connector
--connector-smartscale [335]	connector-smartscale [335]	Enable SmartScale R/W splitting in the connector
--connector-smartscale-sessionid [335]	connector-smartscale-sessionid [335]	The default session ID to use with smart scale
--connectors [335], --dataservice-connectors [335]	connectors [335], dataservice-connectors [335]	Hostnames for the dataservice connectors
--consistency-policy [335], --repl-consistency-policy [335]	consistency-policy [335], repl-consistency-policy [335]	Should the replicator stop or warn if a consistency check fails?
--dataservice-name [335]	dataservice-name [335]	Limit the command to the hosts in this dataservice Multiple data services may be specified by providing a comma separated list
--dataservice-relay-enabled [336]	dataservice-relay-enabled [336]	Make this dataservice the slave of another
--dataservice-schema [336]	dataservice-schema [336]	The db schema to hold dataservice details
--dataservice-thl-port [336]	dataservice-thl-port [336]	Port to use for THL operations
--dataservice-use-relative-latency [336], --use-relative-latency [336]	dataservice-use-relative-latency [336], use-relative-latency [336]	Enable the cluster to operate on relative latency
--dataservice-vip-enabled [336]	dataservice-vip-enabled [336]	Is VIP management enabled?
--dataservice-vip-ip-address [336]	dataservice-vip-ip-address [336]	VIP IP address
--dataservice-vip-netmask [336]	dataservice-vip-netmask [336]	VIP netmask
--datasource-boot-script [336], --repl-datasource-boot-script [336]	datasource-boot-script [336], repl-datasource-boot-script [336]	Database start script
--datasource-enable-ssl [337], --repl-datasource-enable-ssl [337]	datasource-enable-ssl [337], repl-datasource-enable-ssl [337]	Enable SSL connection to DBMS server
--datasource-log-directory [337], --repl-datasource-log-directory [337]	datasource-log-directory [337], repl-datasource-log-directory [337]	Master log directory
--datasource-log-pattern [337], --repl-datasource-log-pattern [337]	datasource-log-pattern [337], repl-datasource-log-pattern [337]	Master log filename pattern
--datasource-mysql-conf [337], --repl-datasource-mysql-conf [337]	datasource-mysql-conf [337], repl-datasource-mysql-conf [337]	MySQL config file
--datasource-mysql-data-directory [337], --repl-datasource-mysql-data-directory [337]	datasource-mysql-data-directory [337], repl-datasource-mysql-data-directory [337]	MySQL data directory
--datasource-mysql-ibdata-directory [337], --repl-datasource-mysql-ibdata-directory [337]	datasource-mysql-ibdata-directory [337], repl-datasource-mysql-ibdata-directory [337]	MySQL InnoDB data directory
--datasource-mysql-iblog-directory [338], --repl-datasource-mysql-iblog-directory [338]	datasource-mysql-iblog-directory [338], repl-datasource-mysql-iblog-directory [338]	MySQL InnoDB log directory
--datasource-mysql-ssl-ca [338], --repl-datasource-mysql-ssl-ca [338]	datasource-mysql-ssl-ca [338], repl-datasource-mysql-ssl-ca [338]	MySQL SSL CA file

INI File Option	CmdLine Option	Description
--datasource=mysql-ssl-cert [338], --repl-data-source=mysql-ssl-cert [338]	datasource=mysql-ssl-cert [338], repl-data-source=mysql-ssl-cert [338]	MySQL SSL certificate file
--datasource=mysql-ssl-key [338], --repl-data-source=mysql-ssl-key [338]	datasource=mysql-ssl-key [338], repl-data-source=mysql-ssl-key [338]	MySQL SSL key file
--datasource=oracle-cle-scan [338], --repl-data-source=oracle-scan [338]	datasource=oracle-scan [338], repl-data-source=oracle-scan [338]	Oracle SCAN
--datasource=oracle-service [338], --repl-data-source=oracle-service [338]	datasource=oracle-service [338], repl-data-source=oracle-service [338]	Oracle Service
--datasource=oracle-service-group [338], --repl-data-source=oracle-service-group [338]	datasource=oracle-service-group [338], repl-data-source=oracle-service-group [338]	Oracle service system group
--datasource=oracle-service-user [339], --repl-data-source=oracle-service-user [339]	datasource=oracle-service-user [339], repl-data-source=oracle-service-user [339]	Oracle service system user
--datasource=oracle-sid [339], --repl-data-source=oracle-sid [339]	datasource=oracle-sid [339], repl-data-source=oracle-sid [339]	Oracle Service ID for older Oracle installations (Oracle 10)
--datasource=pg-archive [339], --repl-data-source=pg-archive [339]	datasource=pg-archive [339], repl-data-source=pg-archive [339]	PostgreSQL archive location
--datasource=pg-conf [339], --repl-data-source=pg-conf [339]	datasource=pg-conf [339], repl-data-source=pg-conf [339]	Location of postgresql.conf
--datasource=pg-home [339], --repl-data-source=pg-home [339]	datasource=pg-home [339], repl-data-source=pg-home [339]	PostgreSQL data directory
--datasource=pg-root [339], --repl-data-source=pg-root [339]	datasource=pg-root [339], repl-data-source=pg-root [339]	Root directory for postgresql installation
--datasource=systemctl-service [339], --repl-data-source=systemctl-service [339]	datasource=systemctl-service [339], repl-data-source=systemctl-service [339]	Database systemctl script
--datasource-type [340], --repl-data-source-type [340]	datasource-type [340], repl-data-source-type [340]	Database type
--delete [340]	delete [340]	Delete the named data service from the configuration Data Service options:
--direct-datasource-log-directory [340], --repl-direct-datasource-log-directory [340]	direct-datasource-log-directory [340], repl-direct-datasource-log-directory [340]	Master log directory
--direct-datasource-log-pattern [340], --repl-direct-datasource-log-pattern [340]	direct-datasource-log-pattern [340], repl-direct-datasource-log-pattern [340]	Master log filename pattern
--direct-datasource=oracle-scan [340], --repl-direct-datasource=oracle-scan [340]	direct-datasource=oracle-scan [340], repl-direct-datasource=oracle-scan [340]	Oracle SCAN
--direct-datasource=oracle-service [341], --repl-direct-datasource=oracle-service [341]	direct-datasource=oracle-service [341], repl-direct-datasource=oracle-service [341]	Oracle Service

INI File Option	CmdLine Option	Description
--direct-datasource-oracle-sid [341], --repl-direct-datasource-oracle-sid [341]	direct-datasource-oracle-sid [341], repl-direct-datasource-oracle-sid [341]	Oracle SID
--direct-data-source-type [341], --repl-direct-datasource-type [341]	direct-datasource-type [341], repl-direct-data-source-type [341]	Database type
--direct-replica-tion-host [341], --di-rect-datasource-host [341], --repl-direct-data-source-host [341]	direct-datasource-host [341], direct-replication-host [341], repl-direct-data-source-host [341]	Database server hostname
--direct-replication-password [341], --direct-datasource-password [341], --repl-direct-datasource-password [341]	direct-datasource-password [341], direct-replication-password [341], repl-direct-datasource-password [341]	Database password
--direct-replica-tion-port [341], --di-rect-datasource-port [341], --repl-direct-data-source-port [341]	direct-datasource-port [341], direct-replication-port [341], repl-direct-data-source-port [341]	Database server port
--direct-replica-tion-user [342], --di-rect-datasource-user [342], --repl-direct-data-source-user [342]	direct-datasource-user [342], direct-replication-user [342], repl-direct-data-source-user [342]	Database login for Tungsten
--disable-relay-logs [342], --repl-disable-relay-logs [342]	disable-relay-logs [342], repl-disable-relay-logs [342]	Disable the use of relay-logs?
--disable-security-controls [342]	disable-security-controls [342]	Disables all forms of security, including SSL, TLS and authentication
--disable-slave-extractor [342], --repl-disable-slave-extractor [342]	disable-slave-extractor [342], repl-disable-slave-extractor [342]	Should slave servers support the master role?
--drop-static-columns-in-updates [342]	drop-static-columns-in-updates [342]	This will modify UPDATE transactions in row-based replication and eliminate any columns that were not modified.
--enable-active-witnesses [342], --active-witnesses [342]	active-witnesses [342], enable-active-witnesses [342]	Enable active witness hosts
--enable-batch-master [342]	enable-batch-master [342]	Enable batch operation for the master
--enable-batch-service [343]	enable-batch-service [343]	Enables batch mode for a service
--enable-batch-slave [343]	enable-batch-slave [343]	Enable batch operation for the slave
--enable-connector-client-ssl [343], --connector-client-ssl [343]	connector-client-ssl [343], enable-connector-client-ssl [343]	Enable SSL encryption of traffic from the client to the connector
--enable-connector-server-ssl [343], --connec-tor-server-ssl [343]	connector-server-ssl [343], enable-connector-server-ssl [343]	Enable SSL encryption of traffic from the connector to the database
--enable-connector-ssl [343], --connector-ssl [343]	connector-ssl [343], enable-connector-ssl [343]	Enable SSL encryption of connector traffic to the database
--enable-heterogeneous-mas-ter [344], --enable-heteroge-neous-master [344]	enable-heterogeneous-mas-ter [344], enable-heteroge-neous-master [344]	Enable heterogeneous operation for the master
--enable-heterogeneous-ser-vice [344], --enable-heteroge-neous-service [344]	enable-heterogeneous-ser-vice [344], enable-heteroge-neous-service [344]	Enable heterogeneous operation

INI File Option	CmdLine Option	Description
--enable-heterogeneous-slave [344], --enable-heterogenous-slave [344]	enable-heterogeneous-slave [344], enable-heterogenous-slave [344]	Enable heterogeneous operation for the slave
--enable-jgroups-ssl [344], --jgroups-ssl [344]	enable-jgroups-ssl [344], jgroups-ssl [344]	Enable SSL encryption of JGroups communication on this host
--enable-rmi-authentication [344], --rmi-authentication [344]	enable-rmi-authentication [344], rmi-authentication [344]	Enable RMI authentication for the services running on this host
--enable-rmi-ssl [345], --rmi-ssl [345]	enable-rmi-ssl [345], rmi-ssl [345]	Enable SSL encryption of RMI communication on this host
--enable-role-change [345], --repl-enable-role-change [345]	enable-role-change [345], repl-enable-role-change [345]	For Oracle installations, ensures that a master and slave pipeline are configured regardless of the default role.
--enable-slave-thl-listener [345], --repl-enable-slave-thl-listener [345]	enable-slave-thl-listener [345], repl-enable-slave-thl-listener [345]	Should this service allow THL connections?
--enable-sudo-access [345], --root-command-prefix [345]	enable-sudo-access [345], root-command-prefix [345]	Run root commands using sudo
--enable-thl-ssl [345], --repl-enable-thl-ssl [345], --thl-ssl [345]	enable-thl-ssl [345], repl-enable-thl-ssl [345], thl-ssl [345]	Enable SSL encryption of THL communication for this service
--enable-validation-check String [345]		Remove a corresponding --skip-validation-check argument
--enable-validation-warnings String [346]		Remove a corresponding --skip-validation-warnings argument
--executable-prefix [346]	executable-prefix [346]	Adds a prefix to command aliases
--file-protection-level [346]	file-protection-level [346]	Protection level for Continuent files
--file-protection-umask [346]	file-protection-umask [346]	Protection umask for Continuent files
--force [346], -f [346]		Do not display confirmation prompts or stop the configure process for errors
--help [346], -h [346]		Displays help message
--host-name [347]	host-name [347]	DNS hostname
--hosts [347]	hosts [347]	Limit the command to the hosts listed You must use the hostname as it appears in the configuration.
--hub [347], --dataservice-hub-service-hub-host [347]	dataservice-hub-host [347], hub [347]	What is the hub host for this all-masters dataservice?
--hub-service [347], --dataservice-hub-service [347]	dataservice-hub-service [347], hub-service [347]	The data service to use for the hub of a star topology
--info [347], -i [347]		Display info, notice, warning and error messages
--install [347]	install [347]	Install service start scripts
--install-directory [347], --home-directory [347]	home-directory [347], install-directory [347]	Installation directory
--install-vmware-redo-reader [348], --repl-install-vmware-redo-reader [348]	install-vmware-redo-reader [348], repl-install-vmware-redo-reader [348]	Install or upgrade the VMware Redo Reader
--java-connector-key-store-password [348]	java-connector-keystore-password [348]	The password for unlocking the tungsten_connector_keystore.jks file in the security directory
--java-connector-key-store-path [348]	java-connector-key-store-path [348]	Local path to the Java Connector Keystore file.
--java-connector-trust-store-password [348]	java-connector-trust-store-password [348]	The password for unlocking the tungsten_connector_truststore.jks file in the security directory
--java-connector-trust-store-path [348]	java-connector-trust-store-path [348]	Local path to the Java Connector Truststore file.

INI File Option	CmdLine Option	Description
--java-enable-concurrent-gc [348], --repl-java-enable-concurrent-gc [348]	java-enable-concurrent-gc [348], repl-java-enable-concurrent-gc [348]	Replicator Java uses concurrent garbage collection
--java-external-lib-dir [348], --repl-java-external-lib-dir [348]	java-external-lib-dir [348], repl-java-external-lib-dir [348]	Directory for 3rd party Jar files required by replicator
--java-file-encoding [348], --repl-java-file-encoding [348]	java-file-encoding [348], repl-java-file-encoding [348]	Java platform charset (esp. for heterogeneous replication)
--java-jgroups-key [349]	java-jgroups-key [349]	The alias to use for the JGroups TLS key in the keystore.
--java-jgroups-key-store-path [349]	java-jgroups-key-store-path [349]	Local path to the JGroups Java Keystore file.
--java-jmxremote-access-path [349]	java-jmxremote-access-path [349]	Local path to the Java JMX Remote Access file.
--java-keystore-password [349]	java-keystore-password [349]	The password for unlocking the tungsten_keystore.jks file in the security directory
--java-keystore-path [349]	java-keystore-path [349]	Local path to the Java Keystore file.
--java-mem-size [349], --repl-java-mem-size [349]	java-mem-size [349], repl-java-mem-size [349]	Replicator Java heap memory size in Mb (min 128)
--java-password-store-path [349]	java-passwordstore-path [349]	Local path to the Java Password Store file.
--java-tls-alias [350]	java-tls-alias [350]	The alias to use for the TLS key/certificate in the keystore and truststore.
--java-tls-key-lifetime [350]	java-tls-key-lifetime [350]	Lifetime for the Java TLS key
--java-tls-key-store-path [350]	java-tls-keystore-path [350]	The keystore holding a certificate to use for all Continuent TLS encryption.
--java-truststore-password [350]	java-truststore-pass-word [350]	The password for unlocking the tungsten_truststore.jks file in the security directory
--java-truststore-path [350]	java-truststore-path [350]	Local path to the Java Truststore file.
--java-user-timezone [350], --repl-java-user-timezone [350]	java-user-timezone [350], repl-java-user-timezone [350]	Java VM Timezone (esp. for cross-site replication)
--log [350]	log [350]	Write all messages, visible and hidden, to this file. You may specify a filename, 'pid' or 'timestamp'.
--log-slave-updates [350]	log-slave-updates [350]	Should slaves log updates to binlog
--master [351], --dataservice-master-host [351], --masters [351], --relay [351]	dataservice-master-host [351], master [351], masters [351], relay [351]	What is the master host for this dataservice?
--master-preferred-role [351], --repl-master-preferred-role [351]	master-preferred-role [351], repl-master-preferred-role [351]	Preferred role for master THL when connecting as a slave (master, slave, etc.)
--master-services [351], --dataservice-master-services [351]	dataservice-master-services [351], master-services [351]	Data service names that should be used on each master
--members [351], --dataservice-hosts [351]	dataservice-hosts [351], members [351]	Hostnames for the dataservice members
--metadata-directory [351], --repl-metadata-directory [351]	metadata-directory [351], repl-metadata-directory [351]	Replicator metadata directory
--mgr-api [351]	mgr-api [351]	Enable the Manager API
--mgr-api-address [352]	mgr-api-address [352]	Address for the Manager API
--mgr-api-full-access [352]	mgr-api-full-access [352]	Enable all Manager API commands. Only the status command will be enabled without it.
--mgr-api-port [352]	mgr-api-port [352]	Port for the Manager API

INI File Option	CmdLine Option	Description
--mgr-group-communication-port [352]	mgr-group-communication-port [352]	Port to use for manager group communication
--mgr-heap-threshold [352]	mgr-heap-threshold [352]	Java memory usage (MB) that will force a Manager restart
--mgr-java-enable-concurrent-gc [352]	mgr-java-enable-concurrent-gc [352]	Manager Java uses concurrent garbage collection
--mgr-java-mem-size [352]	mgr-java-mem-size [352]	Manager Java heap memory size in Mb (min 128)
--mgr-listen-interface [352]	mgr-listen-interface [352]	Listen interface to use for the manager
--mgr-ping-method [353]	mgr-ping-method [353]	Mechanism to use when identifying the liveness of other data-sources (ping, echo)
--mgr-policy-mode [353]	mgr-policy-mode [353]	Manager policy mode
--mgr-rmi-port [353]	mgr-rmi-port [353]	Port to use for the manager RMI server
--mgr-rmi-remote-port [353]	mgr-rmi-remote-port [353]	Port to use for calling the remote manager RMI server
--mgr-ro-slave [353]	mgr-ro-slave [353]	Make slaves read-only
--mgr-vip-arp-path [353]	mgr-vip-arp-path [353]	Path to the arp binary
--mgr-vip-device [353]	mgr-vip-device [353]	VIP network device
--mgr-vip-ifconfig-path [353]	mgr-vip-ifconfig-path [353]	Path to the ifconfig binary
--mgr-wait-for-members [354]	mgr-wait-for-members [354]	Wait for all datasources to be available before completing installation
--mysql-allow-intensive-checks [354]	mysql-allow-intensive-checks [354]	For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility.
--mysql-connectorj-path [354]	mysql-connectorj-path [354]	Path to MySQL Connector/J
--mysql-driver [354]	mysql-driver [354]	MySQL Driver Vendor
--mysql-enable-ansi-quotes [354], --repl-mysql-enable-ansi-quotes [354]	mysql-enable-ansi-quotes [354], repl-mysql-enable-ansi-quotes [354]	Enables ANSI_QUOTES mode for incoming events?
--mysql-enable-noonlykeywords [354], --repl-mysql-enable-noonlykeywords [354]	mysql-enable-noonlykeywords [354], repl-mysql-enable-noonlykeywords [354]	Translates DELETE FROM ONLY } DELETE FROM and UPDATE ONLY } UPDATE.
--mysql-enable-set-tostring [354], --repl-mysql-enable-settostring [354]	mysql-enable-set-tostring [354], repl-mysql-enable-settostring [354]	Decode SET values into their text values?
--mysql-ro-slave [355], --repl-mysql-ro-slave [355]	mysql-ro-slave [355], repl-mysql-ro-slave [355]	Slaves are read-only?
--mysql-server-id [355], --repl-mysql-server-id [355]	mysql-server-id [355], repl-mysql-server-id [355]	Explicitly set the MySQL server ID
--mysql-use-bytes-for-string [355], --repl-mysql-use-bytes-for-string [355]	mysql-use-bytes-for-string [355], repl-mysql-use-bytes-for-string [355]	Transfer strings as their byte representation?
--mysql-xtrabackup-dir [355], --repl-mysql-xtrabackup-dir [355]	mysql-xtrabackup-dir [355], repl-mysql-xtrabackup-dir [355]	Directory to use for storing xtrabackup full & incremental backups
--native-slave-takeover [355], --repl-native-slave-takeover [355]	native-slave-takeover [355], repl-native-slave-takeover [355]	Takeover native replication
--net-ssh-option=key=value		Set the Net::SSH option for remote system calls
--no-deployment [356]	no-deployment [356]	Skip deployment steps that create the install directory
--no-validation [356]	no-validation [356]	Skip validation checks that run on each host
--notice [356], -n [356]		Display notice, warning and error messages
--optimize-row-events [356]	optimize-row-events [356]	Enables or disables optimized row updates

INI File Option	CmdLine Option	Description
--oracle-extractor-method [356], --repl-oracle-extractor-method [356]	oracle-extractor-method [356], repl-oracle-extractor-method [356]	Oracle extractor method
--oracle-home [356], --repl-oracle-home [356]	oracle-home [356], repl-oracle-home [356]	Oracle Home
--oracle-redo-fetcher-host [357], --repl-oracle-redo-fetcher-host [357]	oracle-redo-fetcher-host [357], repl-oracle-redo-fetcher-host [357]	Oracle host that will run the VMRR fetcher process
--oracle-redo-miner-directory [357], --repl-oracle-redo-miner-directory [357]	oracle-redo-miner-directory [357], repl-oracle-redo-miner-directory [357]	Oracle Redo Miner Directory
--oracle-redo-miner-transaction-fragment-size [357], --repl-oracle-redo-miner-transaction-fragment-size [357]	oracle-redo-miner-transaction-fragment-size [357], repl-oracle-redo-miner-transaction-fragment-size [357]	Oracle Redo Miner Transaction Fragment Size
--oracle-redo-password [357], --repl-oracle-redo-password [357]	oracle-redo-password [357], repl-oracle-redo-password [357]	Password for --oracle-redo-user
--oracle-redo-replicate-tables [357], --repl-oracle-redo-replicate-tables [357]	oracle-redo-replicate-tables [357], repl-oracle-redo-replicate-tables [357]	Oracle schema or tables to include in replication
--oracle-redo-tablespace [357], --repl-oracle-redo-tablespace [357]	oracle-redo-tablespace [357], repl-oracle-redo-tablespace [357]	Oracle tablespace to use with the redo reader
--oracle-redo-user [357], --repl-oracle-redo-user [357]	oracle-redo-user [357], repl-oracle-redo-user [357]	Oracle user to create for the redo reader
--oracle-sys-user-password [358], --repl-oracle-sys-user-password [358]	oracle-sys-user-password [358], repl-oracle-sys-user-password [358]	Password for the Oracle SYS user
--oracle-system-user-password [358], --repl-oracle-system-user-password [358]	oracle-system-user-password [358], repl-oracle-system-user-password [358]	Password for the Oracle SYSTEM user
--pg-archive-timeout [358], --repl-pg-archive-timeout [358]	pg-archive-timeout [358], repl-pg-archive-timeout [358]	Timeout for sending unfilled WAL buffers (data loss window)
--pg-ctl [358], --repl-pg-ctl [358]	pg-ctl [358], repl-pg-ctl [358]	Path to the pg_ctl script
--pg-method [358], --repl-pg-method [358]	pg-method [358], repl-pg-method [358]	Postgres Replication method
--pg-standby [358], --repl-pg-standby [358]	pg-standby [358], repl-pg-standby [358]	Path to the pg_standby script
--postgresql-dbname [359], --repl-postgresql-dbname [359]	postgresql-dbname [359], repl-postgresql-dbname [359]	Name of the database to replicate
--postgresql-enable-mysql2pgsql [359], --repl-postgresql-enable-mysql2pgsql [359]	postgresql-enable-mysql2pgsql [359], repl-postgresql-enable-mysql2pgsql [359]	Enable MySQL to PostgreSQL DDL dialect converting filter placeholder
--postgresql-slonek [359], --repl-postgresql-slonek [359]	postgresql-slonek [359], repl-postgresql-slonek [359]	Path to the slonek executable
--postgresql-tables [359], --repl-postgresql-tables [359]	postgresql-tables [359], repl-postgresql-tables [359]	Tables to replicate in form: schema1.table1,schema2.table2,...
--preferred-path [359]	preferred-path [359]	Additional command path
--prefetch-enabled [359]	prefetch-enabled [359]	Should the replicator service be setup as a prefetch applier
--prefetch-max-time-ahead [360]	prefetch-max-time-ahead [360]	Maximum number of seconds that the prefetch applier can get in front of the standard applier

INI File Option	CmdLine Option	Description
--prefetch-min-time-ahead [360]	prefetch-min-time-ahead [360]	Minimum number of seconds that the prefetch applier must be in front of the standard applier
--prefetch-schema [360]	prefetch-schema [360]	Schema to watch for timing prefetch progress
--prefetch-sleep-time [360]	prefetch-sleep-time [360]	How long to wait when the prefetch applier gets too far ahead
--preview [360], -p [360]		Displays the help message and preview the effect of the command line options
--privileged-master [360]	privileged-master [360]	Does the login for the master database service have superuser privileges
--privileged-slave [360]	privileged-slave [360]	Does the login for the slave database service have superuser privileges
--profile file [361]		Sets name of config file
--profile-script [361]	profile-script [361]	Append commands to include env.sh in this profile script
--property=key~=/match/re-place/, --property=key+=value, --property=key=value		Modify the value for key in any file that the configure script touches; key=value - Set key to value without evaluating template values or other rules; key+=value - Evaluate template values and then append value to the end of the line; key~=/match/re-place/ - Evaluate template values then execute the specified Ruby regex with sub. For example --property=replicator.key~=/(.*)/somevalue,\1/ will prepend 'somevalue' before the template value for 'replicator.key'
--protect-configuration-files [361]	protect-configuration-files [361]	When enabled, configuration files are protected to be only readable and updatable by the configured user
--quiet [362], -q [362]		Only display warning and error messages
--redshift-database [362], --replica-redshift-database [362]	redshift-database [362], replica-redshift-database [362]	NAme of the Redshift database to replicate into
--relay-directory [362], --replica-relay-directory [362]	relay-directory [362], replica-relay-directory [362]	Directory for logs transferred from the master
--relay-enabled [362]	relay-enabled [362]	Should the replicator service be setup as a relay master
--relay-source [362], --dataservice-relay-source [362], --master-dataservice [362]	dataservice-relay-source [362], master-dataservice [362], relay-source [362]	Dataservice name to use as a relay source
--remove-property=key		Remove a corresponding --property argument.
--replication-host [363], --datasource-host [363], --replica-datasource-host [363]	datasource-host [363], replica-datasource-host [363], replication-host [363]	Database server hostname
--replication-password [363], --datasource-password [363], --replica-datasource-password [363]	datasource-password [363], replica-datasource-password [363], replication-password [363]	Database password
--replication-port [363], --datasource-port [363], --replica-datasource-port [363]	datasource-port [363], replica-datasource-port [363], replication-port [363]	Database server port
--replication-user [363], --datasource-user [363], --replica-datasource-user [363]	datasource-user [363], replica-datasource-user [363], replication-user [363]	Database login for Tungsten
--reset [363]	reset [363]	Clear the current configuration before processing any arguments
--rmi-port [363], --replica-rmi-port [363]	replica-rmi-port [363], rmi-port [363]	Replication RMI listen port
--rmi-user [363]	rmi-user [363]	The username for RMI authentication
--role [364], --replica-role [364]	replica-role [364], role [364]	What is the replication role for this service?
--router-gateway-port [364]	router-gateway-port [364]	The router gateway port
--router-jmx-port [364]	router-jmx-port [364]	The router jmx port

INI File Option	CmdLine Option	Description
--security-directory [364]	security-directory [364]	Storage directory for the Java security/encryption files
--service-alias [364], --dataservice-service-alias [364]	dataservice-service-alias [364], service-alias [364]	Replication alias of this dataservice
--service-type [364], --reppl-service-type [364]	repl-service-type [364], service-type [364]	What is the replication service type?
--skip-statemap [365]	skip-statemap [365]	Do not copy the cluster-home/conf/statemap.properties from the previous install
--skip-validation-check String [365]		Do not run the specified validation check. Validation checks are identified by the string included in the error they output.
--skip-validation-warnings String [365]		Do not display warnings for the specified validation check. Validation checks are identified by the string included in the warning they output.
--slaves [365], --dataservice-slaves [365]	dataservice-slaves [365], slaves [365]	What are the slaves for this dataservice?
--start [365]	start [365]	Start the services after configuration
--start-and-report [365]	start-and-report [365]	Start the services and report out the status after configuration
--svc-allow-any-remote-service [365], --reppl-svc-allow-any-remote-service [365]	repl-svc-allow-any-remote-service [365], svc-allow-any-remote-service [365]	Replicate from any service
--svc-applier-block-commit-interval [366], --reppl-svc-applier-block-commit-interval [366]	repl-svc-applier-block-commit-interval [366], svc-applier-block-commit-interval [366]	Minimum interval between commits
--svc-applier-block-commit-size [366], --reppl-svc-applier-block-commit-size [366]	repl-svc-applier-block-commit-size [366], svc-applier-block-commit-size [366]	Applier block commit size (min 1)
--svc-applier-filters [366], --reppl-svc-applier-filters [366]	repl-svc-applier-filters [366], svc-applier-filters [366]	Replication service applier filters
--svc-extractor-filters [366], --reppl-svc-extractor-filters [366]	repl-svc-extractor-filters [366], svc-extractor-filters [366]	Replication service extractor filters
--svc-fail-on-zero-row-update [366], --reppl-svc-fail-on-zero-row-update [366]	repl-svc-fail-on-zero-row-update [366], svc-fail-on-zero-row-update [366]	How should the replicator behave when a Row-Based Replication UPDATE does not affect any rows.
--svc-parallelization-type [366], --reppl-svc-parallelization-type [366]	repl-svc-parallelization-type [366], svc-parallelization-type [366]	Method for implementing parallel apply
--svc-remote-filters [367], --reppl-svc-remote-filters [367]	repl-svc-remote-filters [367], svc-remote-filters [367]	Replication service remote download filters
--svc-reposition-on-source-id-change [367], --reppl-svc-reposition-on-source-id-change [367]	repl-svc-reposition-on-source-id-change [367], svc-reposition-on-source-id-change [367]	The master will come ONLINE from the current position if the stored source_id does not match the value in the static properties
--svc-shard-default-db [367], --reppl-svc-shard-default-db [367]	repl-svc-shard-default-db [367], svc-shard-default-db [367]	Mode for setting the shard ID from the default db
--svc-table-engine [367], --reppl-svc-table-engine [367]	repl-svc-table-engine [367], svc-table-engine [367]	Replication service table engine
--svc-thl-filters [367], --reppl-svc-thl-filters [367]	repl-svc-thl-filters [367], svc-thl-filters [367]	Replication service THL filters
--target-dataservice [368], --slave-dataservice [368]	slave-dataservice [368], target-dataservice [368]	Dataservice to use to determine the value of host configuration
--temp-directory [368]	temp-directory [368]	Temporary Directory

INI File Option	CmdLine Option	Description
--template-file-help [368]	template-file-help [368]	Display the keys that may be used in configuration template files
--template-search-path [368]	template-search-path [368]	Adds a new template search path for configuration file generation
--thl-directory [368], --repl-thl-directory [368]	repl-thl-directory [368], thl-directory [368]	Replicator log directory
--thl-do-checksum [368], --repl-thl-do-checksum [368]	repl-thl-do-checksum [368], thl-do-checksum [368]	Execute checksum operations on THL log files
--thl-interface [368], --repl-thl-interface [368]	repl-thl-interface [368], thl-interface [368]	Listen interface to use for THL operations
--thl-log-connection-timeout [369], --repl-thl-log-connection-timeout [369]	repl-thl-log-connection-timeout [369], thl-log-connection-timeout [369]	Number of seconds to wait for a connection to the THL log
--thl-log-file-size [369], --repl-thl-log-file-size [369]	repl-thl-log-file-size [369], thl-log-file-size [369]	File size in bytes for THL disk logs
--thl-log-fsync [369], --repl-thl-log-fsync [369]	repl-thl-log-fsync [369], thl-log-fsync [369]	Fsync THL records on commit. More reliable operation but adds latency to replication when using low-performance storage
--thl-log-retention [369], --repl-thl-log-retention [369]	repl-thl-log-retention [369], thl-log-retention [369]	How long do you want to keep THL files.
--thl-protocol [369], --repl-thl-protocol [369]	repl-thl-protocol [369], thl-protocol [369]	Protocol to use for THL communication with this service
--topology [369], --dataservice-topology [369]	dataservice-topology [369], topology [369]	Replication topology for the dataservice Valid values are star,cluster-slave,master-slave,fan-in,clustered,cluster-alias,all-masters,direct
--track-schema-changes [370]	track-schema-changes [370]	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.
--user [370]	user [370]	System User
--verbose [370], -v [370]		Display debug, info, notice, warning and error messages
--vertica-dbname [370], --repl-vertica-dbname [370]	repl-vertica-dbname [370], vertica-dbname [370]	Name of the database to replicate into
--witnesses [370], --dataservice-witnesses [370]	dataservice-witnesses [370], witnesses [370]	Witness hosts for the dataservice

10.7.1. A **tpm** Options

--allow-bidi-unsafe

Option	--allow-bidi-unsafe [327]
Aliases	--repl-allow-bidi-unsafe [327]
Config File Options	allow-bidi-unsafe [327], repl-allow-bidi-unsafe [327]
Description	Allow unsafe SQL from remote service
Value Type	boolean
Valid Values	false
	true

--api

Option	--api [327]
Aliases	--repl-api [327]
Config File Options	api [327], repl-api [327]
Description	Enable the replication API
Value Type	string

--api-host

Option	<code>--api-host [327]</code>
Aliases	<code>--repl-api-host [327]</code>
Config File Options	<code>api-host [327], repl-api-host [327]</code>
Description	Hostname that the replication API should listen on
Value Type	string

`--api-password`

Option	<code>--api-password [328]</code>
Aliases	<code>--repl-api-password [328]</code>
Config File Options	<code>api-password [328], repl-api-password [328]</code>
Description	HTTP basic auth password for the replication API
Value Type	string

`--api-port`

Option	<code>--api-port [328]</code>
Aliases	<code>--repl-api-port [328]</code>
Config File Options	<code>api-port [328], repl-api-port [328]</code>
Description	Port that the replication API should bind to
Value Type	string

`--api-user`

Option	<code>--api-user [328]</code>
Aliases	<code>--repl-api-user [328]</code>
Config File Options	<code>api-user [328], repl-api-user [328]</code>
Description	HTTP basic auth username for the replication API
Value Type	string

`--application-password`

Option	<code>--application-password [328]</code>
Aliases	<code>--connector-password [328]</code>
Config File Options	<code>application-password [328], connector-password [328]</code>
Description	Database password for the connector
Value Type	string

`--application-port`

Option	<code>--application-port [328]</code>
Aliases	<code>--connector-listen-port [328]</code>
Config File Options	<code>application-port [328], connector-listen-port [328]</code>
Description	Port for the connector to listen on
Value Type	string

`--application-readonly-port`

Option	<code>--application-readonly-port [328]</code>
Aliases	<code>--connector-readonly-listen-port [328]</code>
Config File Options	<code>application-readonly-port [328], connector-readonly-listen-port [328]</code>
Description	Port for the connector to listen for read-only connections on
Value Type	string

--application-user

Option	--application-user [329]
Aliases	--connector-user [329]
Config File Options	application-user [329], connector-user [329]
Description	Database username for the connector
Value Type	string

--auto-enable

Option	--auto-enable [329]
Aliases	--repl-auto-enable [329]
Config File Options	auto-enable [329], repl-auto-enable [329]
Description	Auto-enable services after start-up
Value Type	string

--auto-recovery-delay-interval

Option	--auto-recovery-delay-interval [329]
Aliases	--repl-auto-recovery-delay-interval [329]
Config File Options	auto-recovery-delay-interval [329], repl-auto-recovery-delay-interval [329]
Description	Delay between going OFFLINE and attempting to go ONLINE
Value Type	string
Valid Values	5

The delay between the replicator identifying that autorecovery is needed, and autorecovery being attempted. For busy MySQL installations, larger numbers may be needed to allow time for MySQL servers to restart or recover from their failure.

--auto-recovery-max-attempts

Option	--auto-recovery-max-attempts [329]
Aliases	--repl-auto-recovery-max-attempts [329]
Config File Options	auto-recovery-max-attempts [329], repl-auto-recovery-max-attempts [329]
Description	Maximum number of attempts at automatic recovery
Value Type	numeric
Valid Values	0

Specifies the number of attempts the replicator will make to go back online. When the number of attempts has been reached, the replicator will remain in the `OFFLINE` [207] state.

Autorecovery is not enabled until the value of this parameter is set to a non-zero value. The state of autorecovery can be determined using the `autoRecoveryEnabled` status parameter. The number of attempts made to autorecover can be tracked using the `autoRecoveryTotal` status parameter.

--auto-recovery-reset-interval

Option	--auto-recovery-reset-interval [329]
Aliases	--repl-auto-recovery-reset-interval [329]
Config File Options	auto-recovery-reset-interval [329], repl-auto-recovery-reset-interval [329]
Description	Delay before autorecovery is deemed to have succeeded
Value Type	numeric
Valid Values	5

The time in `ONLINE` [207] 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.

10.7.2. B `tpm` Options

--backup-directory

Option	--backup-directory [330]
Aliases	--repl-backup-directory [330]
Config File Options	backup-directory [330], repl-backup-directory [330]
Description	Permanent backup storage directory
Value Type	string
Default	{home directory}/backups
Valid Values	{home directory}/backups
	{home directory}/backups

--backup-dump-directory

Option	--backup-dump-directory [330]
Aliases	--repl-backup-dump-directory [330]
Config File Options	backup-dump-directory [330], repl-backup-dump-directory [330]
Description	Backup temporary dump directory
Value Type	string

--backup-method

Option	--backup-method [330]
Aliases	--repl-backup-method [330]
Config File Options	backup-method [330], repl-backup-method [330]
Description	Database backup method
Value Type	string
Valid Values	ebs-snapshot
	file-copy-snapshot
	mysqldump
	none
	script
	xtrabackup
	xtrabackup-full
	xtrabackup-incremental

--backup-online

Option	--backup-online [330]
Aliases	--repl-backup-online [330]
Config File Options	backup-online [330], repl-backup-online [330]
Description	Does the backup script support backing up a datasource while it is ONLINE
Value Type	string

--backup-retention

Option	--backup-retention [330]
Aliases	--repl-backup-retention [330]
Config File Options	backup-retention [330], repl-backup-retention [330]
Description	Number of backups to retain
Value Type	numeric

--backup-script

Option	<code>--backup-script [331]</code>
Aliases	<code>--repl-backup-script [331]</code>
Config File Options	<code>backup-script [331], repl-backup-script [331]</code>
Description	What is the path to the backup script
Value Type	filename

--batch-enabled

Option	<code>--batch-enabled [331]</code>
Config File Options	<code>batch-enabled [331]</code>
Description	Should the replicator service use a batch applier
Value Type	string

--batch-load-language

Option	<code>--batch-load-language [331]</code>				
Config File Options	<code>batch-load-language [331]</code>				
Description	Which script language to use for batch loading				
Value Type	string				
Valid Values	<table border="1"><tr><td>js</td><td>JavaScript</td></tr><tr><td>sql</td><td>SQL</td></tr></table>	js	JavaScript	sql	SQL
js	JavaScript				
sql	SQL				

--batch-load-template

Option	<code>--batch-load-template [331]</code>
Config File Options	<code>batch-load-template [331]</code>
Description	Value for the loadBatchTemplate property
Value Type	string

--buffer-size

Option	<code>--buffer-size [331]</code>
Aliases	<code>--repl-buffer-size [331]</code>
Config File Options	<code>buffer-size [331], repl-buffer-size [331]</code>
Description	Replicator queue size between stages (min 1)
Value Type	numeric

10.7.3. C `tpm` Options

--channels

Option	<code>--channels [331]</code>
Aliases	<code>--repl-channels [331]</code>
Config File Options	<code>channels [331], repl-channels [331]</code>
Description	Number of replication channels to use for services
Value Type	numeric

--cluster-slave-auto-recovery-delay-interval

Option	<code>--cluster-slave-auto-recovery-delay-interval [331]</code>
Aliases	<code>--cluster-slave-repl-auto-recovery-delay-interval [331]</code>
Config File Options	<code>cluster-slave-auto-recovery-delay-interval [331], cluster-slave-repl-auto-recovery-delay-interval [331]</code>

Description	Default value for --auto-recovery-delay-interval when --topology=cluster-slave
Value Type	string

--cluster-slave-auto-recovery-max-attempts

Option	<code>--cluster-slave-auto-recovery-max-attempts</code> [332]
Aliases	<code>--cluster-slave-repl-auto-recovery-max-attempts</code> [332]
Config File Options	<code>cluster-slave-auto-recovery-max-attempts</code> [332], <code>cluster-slave-repl-auto-recovery-max-attempts</code> [332]
Description	Default value for --auto-recovery-max-attempts when --topology=cluster-slave

Value Type string

--cluster-slave-auto-recovery-reset-interval

Option	<code>--cluster-slave-auto-recovery-reset-interval</code> [332]
Aliases	<code>--cluster-slave-repl-auto-recovery-reset-interval</code> [332]
Config File Options	<code>cluster-slave-auto-recovery-reset-interval</code> [332], <code>cluster-slave-repl-auto-recovery-reset-interval</code> [332]
Description	Default value for --auto-recovery-reset-interval when --topology=cluster-slave

Value Type string

--composite-datasources

Option	<code>--composite-datasources</code> [332]
Aliases	<code>--dataservice-composite-datasources</code> [332]
Config File Options	<code>composite-datasources</code> [332], <code>dataservice-composite-datasources</code> [332]
Description	Data services that should be added to this composite data service

Value Type string

--config-file-help

Option	<code>--config-file-help</code> [332]
Config File Options	<code>config-file-help</code> [332]
Description	Display help information for content of the config file
Value Type	string

--conn-java-enable-concurrent-gc

Option	<code>--conn-java-enable-concurrent-gc</code> [332]
Config File Options	<code>conn-java-enable-concurrent-gc</code> [332]
Description	Connector Java uses concurrent garbage collection
Value Type	string

--conn-java-mem-size

Option	<code>--conn-java-mem-size</code> [332]
Config File Options	<code>conn-java-mem-size</code> [332]
Description	Connector Java heap memory size used to buffer data between clients and databases
Value Type	numeric
Valid Values	256

The Connector allocates memory for each concurrent client connection, and may use up to the size of the configured MySQL `max_allowed_packet`. With multiple connections, the heap size should be configured to at least the combination of the number of concurrent connections multiplied by the maximum packet size.

--conn-round-robin-include-master

Option	<code>--conn-round-robin-include-master</code> [332]
--------	--

Config File Options	<code>conn-round-robin-include-master</code> [332]
Description	Should the Connector include the master in round-robin load balancing
Value Type	string

--connector-affinity

Option	<code>--connector-affinity</code> [333]
Config File Options	<code>connector-affinity</code> [333]
Description	The default affinity for all connections
Value Type	string

--connector-autoreconnect

Option	<code>--connector-autoreconnect</code> [333]
Config File Options	<code>connector-autoreconnect</code> [333]
Description	Enable auto-reconnect in the connector
Value Type	string

--connector-autoreconnect-killed-connections

Option	<code>--connector-autoreconnect-killed-connections</code> [333]	
Config File Options	<code>connector-autoreconnect-killed-connections</code> [333]	
Description	Enable autoreconnect for connections killed within the connector	
Value Type	string	
Valid Values	<code>false</code>	Do not retry killed connections
	<code>true</code>	

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` [333]. 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

Option	<code>--connector-bridge-mode</code> [333]
Aliases	<code>--enable-connector-bridge-mode</code> [333]
Config File Options	<code>connector-bridge-mode</code> [333], <code>enable-connector-bridge-mode</code> [333]
Description	Enable the Tungsten Connector bridge mode
Value Type	string

--connector-default-schema

Option	<code>--connector-default-schema</code> [333]
Aliases	<code>--connector-forced-schema</code> [333]
Config File Options	<code>connector-default-schema</code> [333], <code>connector-forced-schema</code> [333]
Description	Default schema for the connector to use
Value Type	string

--connector-delete-user-map

Option	<code>--connector-delete-user-map</code> [333]
Config File Options	<code>connector-delete-user-map</code> [333]

Description	Overwrite an existing user.map file
Value Type	string

--connector-disable-connection-warnings

Option	<code>--connector-disable-connection-warnings</code> [334]
Config File Options	<code>connector-disable-connection-warnings</code> [334]
Description	Hide Connector warnings in log files
Value Type	string

--connector-disconnect-timeout

Option	<code>--connector-disconnect-timeout</code> [334]
Config File Options	<code>connector-disconnect-timeout</code> [334]
Description	Time (in seconds) to wait for active connection to disconnect before forcing them closed [default: 5]
Value Type	boolean

--connector-drop-after-max-connections

Option	<code>--connector-drop-after-max-connections</code> [334]
Config File Options	<code>connector-drop-after-max-connections</code> [334]
Description	Instantly drop connections that arrive after --connector-max-connections has been reached
Value Type	boolean

--connector-listen-interface

Option	<code>--connector-listen-interface</code> [334]
Config File Options	<code>connector-listen-interface</code> [334]
Description	Listen interface to use for the connector
Value Type	string

--connector-max-connections

Option	<code>--connector-max-connections</code> [334]
Config File Options	<code>connector-max-connections</code> [334]
Description	The maximum number of connections the connector should allow at any time
Value Type	numeric

--connector-max-slave-latency

Option	<code>--connector-max-slave-latency</code> [334]
Aliases	<code>--connector-max-applied-latency</code> [334]
Config File Options	<code>connector-max-applied-latency</code> [334], <code>connector-max-slave-latency</code> [334]
Description	The maximum applied latency for slave connections
Value Type	string

--connector-readonly

Option	<code>--connector-readonly</code> [334]
Aliases	<code>--enable-connector-readonly</code> [334]
Config File Options	<code>connector-readonly</code> [334], <code>enable-connector-readonly</code> [334]
Description	Enable the Tungsten Connector read-only mode
Value Type	string

--connector-ro-addresses

Option	--connector-ro-addresses [334]
Config File Options	connector-ro-addresses [334]
Description	Connector addresses that should receive a r/o connection
Value Type	string

--connector-rw-addresses

Option	--connector-rw-addresses [335]
Config File Options	connector-rw-addresses [335]
Description	Connector addresses that should receive a r/w connection
Value Type	string

--connector-rwsplitting

Option	--connector-rwsplitting [335]
Config File Options	connector-rwsplitting [335]
Description	Enable DirectReads R/W splitting in the connector
Value Type	string

--connector-smartscale

Option	--connector-smartscale [335]
Config File Options	connector-smartscale [335]
Description	Enable SmartScale R/W splitting in the connector
Value Type	string

--connector-smartscale-sessionid

Option	--connector-smartscale-sessionid [335]
Config File Options	connector-smartscale-sessionid [335]
Description	The default session ID to use with smart scale
Value Type	string

--connectors

Option	--connectors [335]
Aliases	--dataservice-connectors [335]
Config File Options	connectors [335], dataservice-connectors [335]
Description	Hostnames for the dataservice connectors
Value Type	string

--consistency-policy

Option	--consistency-policy [335]
Aliases	--repl-consistency-policy [335]
Config File Options	consistency-policy [335], repl-consistency-policy [335]
Description	Should the replicator stop or warn if a consistency check fails?
Value Type	string

10.7.4. D `tpm` Options

--dataservice-name

Option	--dataservice-name [335]
Config File Options	dataservice-name [335]

Description	Limit the command to the hosts in this dataservice. Multiple data services may be specified by providing a comma separated list
Value Type	string

--dataservice-relay-enabled

Option	--dataservice-relay-enabled [336]
Config File Options	dataservice-relay-enabled [336]
Description	Make this dataservice the slave of another
Value Type	string

--dataservice-schema

Option	--dataservice-schema [336]
Config File Options	dataservice-schema [336]
Description	The db schema to hold dataservice details
Value Type	string

--dataservice-thl-port

Option	--dataservice-thl-port [336]
Config File Options	dataservice-thl-port [336]
Description	Port to use for THL operations
Value Type	string

--dataservice-use-relative-latency

Option	--dataservice-use-relative-latency [336]
Aliases	--use-relative-latency [336]
Config File Options	dataservice-use-relative-latency [336], use-relative-latency [336]
Description	Enable the cluster to operate on relative latency
Value Type	string

--dataservice-vip-enabled

Option	--dataservice-vip-enabled [336]
Config File Options	dataservice-vip-enabled [336]
Description	Is VIP management enabled?
Value Type	string

--dataservice-vip-ipaddress

Option	--dataservice-vip-ipaddress [336]
Config File Options	dataservice-vip-ipaddress [336]
Description	VIP IP address
Value Type	string

--dataservice-vip-netmask

Option	--dataservice-vip-netmask [336]
Config File Options	dataservice-vip-netmask [336]
Description	VIP netmask
Value Type	string

--datasource-boot-script

Option	--datasource-boot-script [336]
Aliases	--repl-datasource-boot-script [336]
Config File Options	datasource-boot-script [336], repl-datasource-boot-script [336]
Description	Database start script
Value Type	string

--datasource-enable-ssl

Option	--datasource-enable-ssl [337]
Aliases	--repl-datasource-enable-ssl [337]
Config File Options	datasource-enable-ssl [337], repl-datasource-enable-ssl [337]
Description	Enable SSL connection to DBMS server
Value Type	string

--datasource-log-directory

Option	--datasource-log-directory [337]
Aliases	--repl-datasource-log-directory [337]
Config File Options	datasource-log-directory [337], repl-datasource-log-directory [337]
Description	Master log directory
Value Type	string

--datasource-log-pattern

Option	--datasource-log-pattern [337]
Aliases	--repl-datasource-log-pattern [337]
Config File Options	datasource-log-pattern [337], repl-datasource-log-pattern [337]
Description	Master log filename pattern
Value Type	string

--datasource-mysql-conf

Option	--datasource-mysql-conf [337]
Aliases	--repl-datasource-mysql-conf [337]
Config File Options	datasource-mysql-conf [337], repl-datasource-mysql-conf [337]
Description	MySQL config file
Value Type	string

--datasource-mysql-data-directory

Option	--datasource-mysql-data-directory [337]
Aliases	--repl-datasource-mysql-data-directory [337]
Config File Options	datasource-mysql-data-directory [337], repl-datasource-mysql-data-directory [337]
Description	MySQL data directory
Value Type	string

--datasource-mysql-ibdata-directory

Option	--datasource-mysql-ibdata-directory [337]
Aliases	--repl-datasource-mysql-ibdata-directory [337]
Config File Options	datasource-mysql-ibdata-directory [337], repl-datasource-mysql-ibdata-directory [337]
Description	MySQL InnoDB data directory
Value Type	string

--datasource-mysql-iblog-directory

Option	--datasource-mysql-iblog-directory [338]
Aliases	--repl-datasource-mysql-iblog-directory [338]
Config File Options	datasource-mysql-iblog-directory [338], repl-datasource-mysql-iblog-directory [338]
Description	MySQL InnoDB log directory
Value Type	string

--datasource-mysql-ssl-ca

Option	--datasource-mysql-ssl-ca [338]
Aliases	--repl-datasource-mysql-ssl-ca [338]
Config File Options	datasource-mysql-ssl-ca [338], repl-datasource-mysql-ssl-ca [338]
Description	MySQL SSL CA file

--datasource-mysql-ssl-cert

Option	--datasource-mysql-ssl-cert [338]
Aliases	--repl-datasource-mysql-ssl-cert [338]
Config File Options	datasource-mysql-ssl-cert [338], repl-datasource-mysql-ssl-cert [338]
Description	MySQL SSL certificate file

--datasource-mysql-ssl-key

Option	--datasource-mysql-ssl-key [338]
Aliases	--repl-datasource-mysql-ssl-key [338]
Config File Options	datasource-mysql-ssl-key [338], repl-datasource-mysql-ssl-key [338]
Description	MySQL SSL key file

--datasource-oracle-scan

Option	--datasource-oracle-scan [338]
Aliases	--repl-datasource-oracle-scan [338]
Config File Options	datasource-oracle-scan [338], repl-datasource-oracle-scan [338]
Description	Oracle SCAN

--datasource-oracle-service

Option	--datasource-oracle-service [338]
Aliases	--repl-datasource-oracle-service [338]
Config File Options	datasource-oracle-service [338], repl-datasource-oracle-service [338]
Description	Oracle Service

--datasource-oracle-service-group

Option	--datasource-oracle-service-group [338]
Aliases	--repl-datasource-oracle-service-group [338]
Config File Options	datasource-oracle-service-group [338], repl-datasource-oracle-service-group [338]
Description	Oracle service system group

Value Type	string
-------------------	--------

--datasource-oracle-service-user

Option	--datasource-oracle-service-user [339]
Aliases	--repl-datasource-oracle-service-user [339]
Config File Options	datasource-oracle-service-user [339], repl-datasource-oracle-service-user [339]
Description	Oracle service system user
Value Type	string

--datasource-oracle-sid

Option	--datasource-oracle-sid [339]
Aliases	--repl-datasource-oracle-sid [339]
Config File Options	datasource-oracle-sid [339], repl-datasource-oracle-sid [339]
Description	Oracle Service ID for older Oracle installations (Oracle 10)
Value Type	string

--datasource-pg-archive

Option	--datasource-pg-archive [339]
Aliases	--repl-datasource-pg-archive [339]
Config File Options	datasource-pg-archive [339], repl-datasource-pg-archive [339]
Description	PostgreSQL archive location
Value Type	string

--datasource-pg-conf

Option	--datasource-pg-conf [339]
Aliases	--repl-datasource-pg-conf [339]
Config File Options	datasource-pg-conf [339], repl-datasource-pg-conf [339]
Description	Location of postgresql.conf
Value Type	string

--datasource-pg-home

Option	--datasource-pg-home [339]
Aliases	--repl-datasource-pg-home [339]
Config File Options	datasource-pg-home [339], repl-datasource-pg-home [339]
Description	PostgreSQL data directory
Value Type	string

--datasource-pg-root

Option	--datasource-pg-root [339]
Aliases	--repl-datasource-pg-root [339]
Config File Options	datasource-pg-root [339], repl-datasource-pg-root [339]
Description	Root directory for postgresql installation
Value Type	string

--datasource-systemctl-service

Option	--datasource-systemctl-service [339]
Aliases	--repl-datasource-systemctl-service [339]

Config File Options	datasource-systemctl-service [339] , repl-datasource-systemctl-service [339]
Description	Database systemctl script
Value Type	string

Specifies the command name or full path of the command that should be used to control the database service, including startup, shutdown and restart. This is used by the Tungsten Replication to control the underlying database service. By default, this will be configured to the service according to your environment if it has been found during installation. For example, the **services** command, or `/etc/init.d/mysql`.

--datasource-type

Option	--datasource-type [340]	
Aliases	--repl-datasource-type [340]	
Config File Options	datasource-type [340] , repl-datasource-type [340]	
Description	Database type	
Value Type	string	
Default	mysql	
Valid Values	file	File
	hdfs	HDFS (Hadoop)
	mongodb	MongoDB
	mysql	MySQL
	oracle	Oracle
	vertica	Vertica

--delete

Option	--delete [340]	
Config File Options	delete [340]	
Description	Delete the named data service from the configuration Data Service options:	
Value Type	string	

--direct-datasource-log-directory

Option	--direct-datasource-log-directory [340]	
Aliases	--repl-direct-datasource-log-directory [340]	
Config File Options	direct-datasource-log-directory [340] , repl-direct-datasource-log-directory [340]	
Description	Master log directory	

--direct-datasource-log-pattern

Option	--direct-datasource-log-pattern [340]	
Aliases	--repl-direct-datasource-log-pattern [340]	
Config File Options	direct-datasource-log-pattern [340] , repl-direct-datasource-log-pattern [340]	
Description	Master log filename pattern	

--direct-datasource-oracle-scan

Option	--direct-datasource-oracle-scan [340]	
Aliases	--repl-direct-datasource-oracle-scan [340]	
Config File Options	direct-datasource-oracle-scan [340] , repl-direct-datasource-oracle-scan [340]	
Description	Oracle SCAN	

--direct-datasource-oracle-service

Option	<code>--direct-datasource-oracle-service</code> [341]
Aliases	<code>--repl-direct-datasource-oracle-service</code> [341]
Config File Options	<code>direct-datasource-oracle-service</code> [341], <code>repl-direct-datasource-oracle-service</code> [341]
Description	Oracle Service
Value Type	string

--direct-datasource-oracle-sid

Option	<code>--direct-datasource-oracle-sid</code> [341]
Aliases	<code>--repl-direct-datasource-oracle-sid</code> [341]
Config File Options	<code>direct-datasource-oracle-sid</code> [341], <code>repl-direct-datasource-oracle-sid</code> [341]
Description	Oracle SID
Value Type	string

--direct-datasource-type

Option	<code>--direct-datasource-type</code> [341]	
Aliases	<code>--repl-direct-datasource-type</code> [341]	
Config File Options	<code>direct-datasource-type</code> [341], <code>repl-direct-datasource-type</code> [341]	
Description	Database type	
Value Type	string	
Default	mysql	
Valid Values	file	File
	hdfs	HDFS (Hadoop)
	mongodb	MongoDB
	mysql	
	mysql	MySQL
	oracle	Oracle
	vertica	Vertica

--direct-replication-host

Option	<code>--direct-replication-host</code> [341]	
Aliases	<code>--direct-datasource-host</code> [341], <code>--repl-direct-datasource-host</code> [341]	
Config File Options	<code>direct-datasource-host</code> [341], <code>direct-replication-host</code> [341], <code>repl-direct-datasource-host</code> [341]	
Description	Database server hostname	
Value Type	string	

--direct-replication-password

Option	<code>--direct-replication-password</code> [341]	
Aliases	<code>--direct-datasource-password</code> [341], <code>--repl-direct-datasource-password</code> [341]	
Config File Options	<code>direct-datasource-password</code> [341], <code>direct-replication-password</code> [341], <code>repl-direct-datasource-password</code> [341]	
Description	Database password	
Value Type	string	

--direct-replication-port

Option	<code>--direct-replication-port</code> [341]	
Aliases	<code>--direct-datasource-port</code> [341], <code>--repl-direct-datasource-port</code> [341]	

Config File Options	<code>direct-datasource-port</code> [341], <code>direct-replication-port</code> [341], <code>repl-direct-datasource-port</code> [341]
Description	Database server port
Value Type	string

`--direct-replication-user`

Option	<code>--direct-replication-user</code> [342]
Aliases	<code>--direct-datasource-user</code> [342], <code>--repl-direct-datasource-user</code> [342]
Config File Options	<code>direct-datasource-user</code> [342], <code>direct-replication-user</code> [342], <code>repl-direct-datasource-user</code> [342]
Description	Database login for Tungsten
Value Type	string

`--disable-relay-logs`

Option	<code>--disable-relay-logs</code> [342]
Aliases	<code>--repl-disable-relay-logs</code> [342]
Config File Options	<code>disable-relay-logs</code> [342], <code>repl-disable-relay-logs</code> [342]
Description	Disable the use of relay-logs?
Value Type	string

`--disable-security-controls`

Option	<code>--disable-security-controls</code> [342]
Config File Options	<code>disable-security-controls</code> [342]
Description	Disables all forms of security, including SSL, TLS and authentication
Value Type	string

`--disable-slave-extractor`

Option	<code>--disable-slave-extractor</code> [342]
Aliases	<code>--repl-disable-slave-extractor</code> [342]
Config File Options	<code>disable-slave-extractor</code> [342], <code>repl-disable-slave-extractor</code> [342]
Description	Should slave servers support the master role?
Value Type	string

`--drop-static-columns-in-updates`

Option	<code>--drop-static-columns-in-updates</code> [342]
Config File Options	<code>drop-static-columns-in-updates</code> [342]
Description	This will modify UPDATE transactions in row-based replication and eliminate any columns that were not modified.
Value Type	string

10.7.5. E `tpm` Options

`--enable-active-witnesses`

Option	<code>--enable-active-witnesses</code> [342]
Aliases	<code>--active-witnesses</code> [342]
Config File Options	<code>active-witnesses</code> [342], <code>enable-active-witnesses</code> [342]
Description	Enable active witness hosts
Value Type	string

`--enable-batch-master`

Option	--enable-batch-master [342]
Config File Options	enable-batch-master [342]
Description	Enable batch operation for the master
Value Type	string

--enable-batch-service

Option	--enable-batch-service [343]
Config File Options	enable-batch-service [343]
Description	Enables batch mode for a service
Value Type	string
Valid Values	false

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 [355]` 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 `addColumnsToDelete` 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 [355]` is set to true.
- `pkey` filter is enabled (`q-to-dbms` stage).

--enable-batch-slave

Option	--enable-batch-slave [343]
Config File Options	enable-batch-slave [343]
Description	Enable batch operation for the slave
Value Type	string

--enable-connector-client-ssl

Option	--enable-connector-client-ssl [343]
Aliases	--connector-client-ssl [343]
Config File Options	connector-client-ssl [343], enable-connector-client-ssl [343]
Description	Enable SSL encryption of traffic from the client to the connector
Value Type	string

--enable-connector-server-ssl

Option	--enable-connector-server-ssl [343]
Aliases	--connector-server-ssl [343]
Config File Options	connector-server-ssl [343], enable-connector-server-ssl [343]
Description	Enable SSL encryption of traffic from the connector to the database
Value Type	string

--enable-connector-ssl

Option	<code>--enable-connector-ssl [343]</code>
Aliases	<code>--connector-ssl [343]</code>
Config File Options	<code>connector-ssl [343], enable-connector-ssl [343]</code>
Description	Enable SSL encryption of connector traffic to the database
Value Type	string

`--enable-heterogeneous-master`

Option	<code>--enable-heterogeneous-master [344]</code>
Aliases	<code>--enable-heterogenous-master [344]</code>
Config File Options	<code>enable-heterogeneous-master [344], enable-heterogenous-master [344]</code>
Description	Enable heterogeneous operation for the master
Value Type	string

`--enable-heterogeneous-service`

Option	<code>--enable-heterogeneous-service [344]</code>
Aliases	<code>--enable-heterogenous-service [344]</code>
Config File Options	<code>enable-heterogeneous-service [344], enable-heterogenous-service [344]</code>
Description	Enable heterogeneous operation
Value Type	string

- **On a Master**

- `--mysql-use-bytes-for-string [355]` 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 [355]` is set to true.
- `pkey` filter is enabled (`q-to-dbms` stage).

`--enable-heterogeneous-slave`

Option	<code>--enable-heterogeneous-slave [344]</code>
Aliases	<code>--enable-heterogenous-slave [344]</code>
Config File Options	<code>enable-heterogeneous-slave [344], enable-heterogenous-slave [344]</code>
Description	Enable heterogeneous operation for the slave
Value Type	string

`--enable-jgroups-ssl`

Option	<code>--enable-jgroups-ssl [344]</code>
Aliases	<code>--jgroups-ssl [344]</code>
Config File Options	<code>enable-jgroups-ssl [344], jgroups-ssl [344]</code>
Description	Enable SSL encryption of JGroups communication on this host
Value Type	string

`--enable-rmi-authentication`

Option	--enable-rmi-authentication [344]
Aliases	--rmi-authentication [344]
Config File Options	enable-rmi-authentication [344], rmi-authentication [344]
Description	Enable RMI authentication for the services running on this host
Value Type	string

--enable-rmi-ssl

Option	--enable-rmi-ssl [345]
Aliases	--rmi-ssl [345]
Config File Options	enable-rmi-ssl [345], rmi-ssl [345]
Description	Enable SSL encryption of RMI communication on this host
Value Type	string

--enable-role-change

Option	--enable-role-change [345]
Aliases	--repl-enable-role-change [345]
Config File Options	enable-role-change [345], repl-enable-role-change [345]
Description	For Oracle installations, ensures that a master and slave pipeline are configured regardless of the default role.
Value Type	string

For Oracle installations, ensures that a master and slave pipeline are configured regardless of the default role. This ensures that in the event of a role change, for example when a manual switch has taken place, that the slave has the correct configuration to operate as a master, and vice versa. This option also ensures that the replication position can be updated correctly after the switch operation.

--enable-slave-thl-listener

Option	--enable-slave-thl-listener [345]
Aliases	--repl-enable-slave-thl-listener [345]
Config File Options	enable-slave-thl-listener [345], repl-enable-slave-thl-listener [345]
Description	Should this service allow THL connections?
Value Type	string

--enable-sudo-access

Option	--enable-sudo-access [345]
Aliases	--root-command-prefix [345]
Config File Options	enable-sudo-access [345], root-command-prefix [345]
Description	Run root commands using sudo
Value Type	string

--enable-thl-ssl

Option	--enable-thl-ssl [345]
Aliases	--repl-enable-thl-ssl [345], --thl-ssl [345]
Config File Options	enable-thl-ssl [345], repl-enable-thl-ssl [345], thl-ssl [345]
Description	Enable SSL encryption of THL communication for this service
Value Type	string

--enable-validation-check String

Option	--enable-validation-check String [345]
Config File Options	

Description	Remove a corresponding --skip-validation-check argument
Value Type	string

--enable-validation-warnings String

Option	<code>--enable-validation-warnings String</code> [346]
Config File Options	
Description	Remove a corresponding --skip-validation-warnings argument
Value Type	string

--executable-prefix

Option	<code>--executable-prefix</code> [346]
Config File Options	<code>executable-prefix</code> [346]
Description	Adds a prefix to command aliases
Value Type	string

When enabled, the supplied prefix is added to each command alias that is generated for a given installation. This enables multiple installations to co-exist and 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.

10.7.6. F tpm Options

--file-protection-level

Option	<code>--file-protection-level</code> [346]
Config File Options	<code>file-protection-level</code> [346]
Description	Protection level for Continuent files
Value Type	string

--file-protection-umask

Option	<code>--file-protection-umask</code> [346]
Config File Options	<code>file-protection-umask</code> [346]
Description	Protection umask for Continuent files
Value Type	string

--force

Option	<code>--force</code> [346]
Aliases	<code>-f</code> [346]
Config File Options	
Description	Do not display confirmation prompts or stop the configure process for errors
Value Type	string

10.7.7. H tpm Options

--help

Option	<code>--help</code> [346]
Aliases	<code>-h</code> [346]
Config File Options	
Description	Displays help message
Value Type	string

--host-name

Option	--host-name [347]
Config File Options	host-name [347]
Description	DNS hostname
Value Type	string

--hosts

Option	--hosts [347]
Config File Options	hosts [347]
Description	Limit the command to the hosts listed You must use the hostname as it appears in the configuration.
Value Type	string

--hub

Option	--hub [347]
Aliases	--dataservice-hub-host [347]
Config File Options	dataservice-hub-host [347], hub [347]
Description	What is the hub host for this all-masters dataservice?
Value Type	string

--hub-service

Option	--hub-service [347]
Aliases	--dataservice-hub-service [347]
Config File Options	dataservice-hub-service [347], hub-service [347]
Description	The data service to use for the hub of a star topology
Value Type	string

10.7.8. **I tpm Options**

--info

Option	--info [347]
Aliases	-i [347]
Config File Options	
Description	Display info, notice, warning and error messages
Value Type	string

--install

Option	--install [347]
Config File Options	install [347]
Description	Install service start scripts
Value Type	string

--install-directory

Option	--install-directory [347]
Aliases	--home-directory [347]
Config File Options	home-directory [347], install-directory [347]
Description	Installation directory
Value Type	string

--install-vmware-redo-reader

Option	--install-vmware-redo-reader [348]
Aliases	--repl-install-vmware-redo-reader [348]
Config File Options	install-vmware-redo-reader [348], repl-install-vmware-redo-reader [348]
Description	Install or upgrade the VMware Redo Reader
Value Type	string

10.7.9. J tpm Options

--java-connector-keystore-password

Option	--java-connector-keystore-password [348]
Config File Options	java-connector-keystore-password [348]
Description	The password for unlocking the tungsten_connector_keystore.jks file in the security directory
Value Type	string

--java-connector-keystore-path

Option	--java-connector-keystore-path [348]
Config File Options	java-connector-keystore-path [348]
Description	Local path to the Java Connector Keystore file.
Value Type	filename

--java-connector-truststore-password

Option	--java-connector-truststore-password [348]
Config File Options	java-connector-truststore-password [348]
Description	The password for unlocking the tungsten_connector_truststore.jks file in the security directory
Value Type	string

--java-connector-truststore-path

Option	--java-connector-truststore-path [348]
Config File Options	java-connector-truststore-path [348]
Description	Local path to the Java Connector Truststore file.
Value Type	filename

--java-enable-concurrent-gc

Option	--java-enable-concurrent-gc [348]
Aliases	--repl-java-enable-concurrent-gc [348]
Config File Options	java-enable-concurrent-gc [348], repl-java-enable-concurrent-gc [348]
Description	Replicator Java uses concurrent garbage collection

--java-external-lib-dir

Option	--java-external-lib-dir [348]
Aliases	--repl-java-external-lib-dir [348]
Config File Options	java-external-lib-dir [348], repl-java-external-lib-dir [348]
Description	Directory for 3rd party Jar files required by replicator

--java-file-encoding

Option	--java-file-encoding [348]
Aliases	--repl-java-file-encoding [348]
Config File Options	java-file-encoding [348] , repl-java-file-encoding [348]
Description	Java platform charset (esp. for heterogeneous replication)
Value Type	string

--java-jgroups-key

Option	--java-jgroups-key [349]
Config File Options	java-jgroups-key [349]
Description	The alias to use for the JGroups TLS key in the keystore.
Value Type	string

--java-jgroups-keystore-path

Option	--java-jgroups-keystore-path [349]
Config File Options	java-jgroups-keystore-path [349]
Description	Local path to the JGroups Java Keystore file.
Value Type	filename

--java-jmxremote-access-path

Option	--java-jmxremote-access-path [349]
Config File Options	java-jmxremote-access-path [349]
Description	Local path to the Java JMX Remote Access file.
Value Type	filename

--java-keystore-password

Option	--java-keystore-password [349]
Config File Options	java-keystore-password [349]
Description	The password for unlocking the tungsten_keystore.jks file in the security directory
Value Type	string

--java-keystore-path

Option	--java-keystore-path [349]
Config File Options	java-keystore-path [349]
Description	Local path to the Java Keystore file.
Value Type	filename

--java-mem-size

Option	--java-mem-size [349]
Aliases	--repl-java-mem-size [349]
Config File Options	java-mem-size [349] , repl-java-mem-size [349]
Description	Replicator Java heap memory size in Mb (min 128)
Value Type	numeric

--java-passwordstore-path

Option	--java-passwordstore-path [349]
Config File Options	java-passwordstore-path [349]
Description	Local path to the Java Password Store file.

Value Type	filename
------------	----------

--java-tls-alias

Option	--java-tls-alias [350]
Config File Options	java-tls-alias [350]
Description	The alias to use for the TLS key/certificate in the keystore and truststore.
Value Type	string

--java-tls-key-lifetime

Option	--java-tls-key-lifetime [350]
Config File Options	java-tls-key-lifetime [350]
Description	Lifetime for the Java TLS key
Value Type	numeric

--java-tls-keystore-path

Option	--java-tls-keystore-path [350]
Config File Options	java-tls-keystore-path [350]
Description	The keystore holding a certificate to use for all Continuent TLS encryption.
Value Type	string

--java-truststore-password

Option	--java-truststore-password [350]
Config File Options	java-truststore-password [350]
Description	The password for unlocking the tungsten_truststore.jks file in the security directory
Value Type	string

--java-truststore-path

Option	--java-truststore-path [350]
Config File Options	java-truststore-path [350]
Description	Local path to the Java Truststore file.
Value Type	filename

--java-user-timezone

Option	--java-user-timezone [350]
Aliases	--repl-java-user-timezone [350]
Config File Options	java-user-timezone [350], repl-java-user-timezone [350]
Description	Java VM Timezone (esp. for cross-site replication)
Value Type	numeric

10.7.10. L tpm Options

--log

Option	--log [350]
Config File Options	log [350]
Description	Write all messages, visible and hidden, to this file. You may specify a filename, 'pid' or 'timestamp'.
Value Type	numeric

--log-slave-updates

Option	<code>--log-slave-updates [350]</code>
Config File Options	<code>log-slave-updates [350]</code>
Description	Should slaves log updates to binlog
Value Type	string

10.7.11. M `tpm` Options

--master

Option	<code>--master [351]</code>
Aliases	<code>--dataservice-master-host [351], --masters [351], --relay [351]</code>
Config File Options	<code>dataservice-master-host [351], master [351], masters [351], relay [351]</code>
Description	What is the master host for this dataservice?
Value Type	string

--master-preferred-role

Option	<code>--master-preferred-role [351]</code>
Aliases	<code>--repl-master-preferred-role [351]</code>
Config File Options	<code>master-preferred-role [351], repl-master-preferred-role [351]</code>
Description	Preferred role for master THL when connecting as a slave (master, slave, etc.)
Value Type	string

--master-services

Option	<code>--master-services [351]</code>
Aliases	<code>--dataservice-master-services [351]</code>
Config File Options	<code>dataservice-master-services [351], master-services [351]</code>
Description	Data service names that should be used on each master
Value Type	string

--members

Option	<code>--members [351]</code>
Aliases	<code>--dataservice-hosts [351]</code>
Config File Options	<code>dataservice-hosts [351], members [351]</code>
Description	Hostnames for the dataservice members
Value Type	string

--metadata-directory

Option	<code>--metadata-directory [351]</code>
Aliases	<code>--repl-metadata-directory [351]</code>
Config File Options	<code>metadata-directory [351], repl-metadata-directory [351]</code>
Description	Replicator metadata directory
Value Type	string

--mgr-api

Option	<code>--mgr-api [351]</code>
Config File Options	<code>mgr-api [351]</code>
Description	Enable the Manager API
Value Type	string

--mgr-api-address

Option	--mgr-api-address [352]
Config File Options	mgr-api-address [352]
Description	Address for the Manager API
Value Type	string

--mgr-api-full-access

Option	--mgr-api-full-access [352]
Config File Options	mgr-api-full-access [352]
Description	Enable all Manager API commands. Only the status command will be enabled without it.
Value Type	string

--mgr-api-port

Option	--mgr-api-port [352]
Config File Options	mgr-api-port [352]
Description	Port for the Manager API
Value Type	string

--mgr-group-communication-port

Option	--mgr-group-communication-port [352]
Config File Options	mgr-group-communication-port [352]
Description	Port to use for manager group communication
Value Type	string

--mgr-heap-threshold

Option	--mgr-heap-threshold [352]
Config File Options	mgr-heap-threshold [352]
Description	Java memory usage (MB) that will force a Manager restart
Value Type	string

--mgr-java-enable-concurrent-gc

Option	--mgr-java-enable-concurrent-gc [352]
Config File Options	mgr-java-enable-concurrent-gc [352]
Description	Manager Java uses concurrent garbage collection
Value Type	string

--mgr-java-mem-size

Option	--mgr-java-mem-size [352]
Config File Options	mgr-java-mem-size [352]
Description	Manager Java heap memory size in Mb (min 128)
Value Type	numeric

--mgr-listen-interface

Option	--mgr-listen-interface [352]
Config File Options	mgr-listen-interface [352]
Description	Listen interface to use for the manager
Value Type	string

--mgr-ping-method

Option	--mgr-ping-method [353]
Config File Options	mgr-ping-method [353]
Description	Mechanism to use when identifying the liveness of other datasources (ping, echo)
Value Type	string

--mgr-policy-mode

Option	--mgr-policy-mode [353]
Config File Options	mgr-policy-mode [353]
Description	Manager policy mode
Value Type	string
Valid Values	automatic
	maintenance
	manual

--mgr-rmi-port

Option	--mgr-rmi-port [353]
Config File Options	mgr-rmi-port [353]
Description	Port to use for the manager RMI server
Value Type	string

--mgr-rmi-remote-port

Option	--mgr-rmi-remote-port [353]
Config File Options	mgr-rmi-remote-port [353]
Description	Port to use for calling the remote manager RMI server
Value Type	string

--mgr-ro-slave

Option	--mgr-ro-slave [353]
Config File Options	mgr-ro-slave [353]
Description	Make slaves read-only
Value Type	string

--mgr-vip-arp-path

Option	--mgr-vip-arp-path [353]
Config File Options	mgr-vip-arp-path [353]
Description	Path to the arp binary
Value Type	filename

--mgr-vip-device

Option	--mgr-vip-device [353]
Config File Options	mgr-vip-device [353]
Description	VIP network device
Value Type	string

--mgr-vip-ifconfig-path

Option	--mgr-vip-ifconfig-path [353]
---------------	-------------------------------

Config File Options	mgr-vip-ifconfig-path [353]
Description	Path to the ifconfig binary
Value Type	filename

`--mgr-wait-for-members`

Option	--mgr-wait-for-members [354]
Config File Options	mgr-wait-for-members [354]
Description	Wait for all datasources to be available before completing installation
Value Type	string

`--mysql-allow-intensive-checks`

Option	--mysql-allow-intensive-checks [354]
Config File Options	mysql-allow-intensive-checks [354]
Description	For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility.
Value Type	string

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

`--mysql-connectorj-path`

Option	--mysql-connectorj-path [354]
Config File Options	mysql-connectorj-path [354]
Description	Path to MySQL Connector/J
Value Type	filename

`--mysql-driver`

Option	--mysql-driver [354]
Config File Options	mysql-driver [354]
Description	MySQL Driver Vendor
Value Type	string

`--mysql-enable-ansiQuotes`

Option	--mysql-enable-ansiQuotes [354]
Aliases	--repl-mysql-enable-ansiQuotes [354]
Config File Options	mysql-enable-ansiQuotes [354], repl-mysql-enable-ansiQuotes [354]
Description	Enables ANSI_QUOTES mode for incoming events?
Value Type	string

`--mysql-enable-noonlykeywords`

Option	--mysql-enable-noonlykeywords [354]
Aliases	--repl-mysql-enable-noonlykeywords [354]
Config File Options	mysql-enable-noonlykeywords [354], repl-mysql-enable-noonlykeywords [354]
Description	Translates DELETE FROM ONLY -> DELETE FROM and UPDATE ONLY -> UPDATE.
Value Type	string

`--mysql-enable-settostring`

Option	--mysql-enable-settostring [354]
---------------	--

Aliases	<code>--repl-mysql-enable-settostring [354]</code>
Config File Options	<code>mysql-enable-settostring [354], repl-mysql-enable-settostring [354]</code>
Description	Decode SET values into their text values?
Value Type	string

`--mysql-ro-slave`

Option	<code>--mysql-ro-slave [355]</code>
Aliases	<code>--repl-mysql-ro-slave [355]</code>
Config File Options	<code>mysql-ro-slave [355], repl-mysql-ro-slave [355]</code>
Description	Slaves are read-only?
Value Type	string

`--mysql-server-id`

Option	<code>--mysql-server-id [355]</code>
Aliases	<code>--repl-mysql-server-id [355]</code>
Config File Options	<code>mysql-server-id [355], repl-mysql-server-id [355]</code>
Description	Explicitly set the MySQL server ID
Value Type	string

Setting this option explicitly sets the server-id information normally located in the MySQL configuration (`my.cnf`). This is useful in situations where there may be multiple MySQL installations and the server ID needs to be identified to prevent collisions when reading from the same master.

`--mysql-use-bytes-for-string`

Option	<code>--mysql-use-bytes-for-string [355]</code>
Aliases	<code>--repl-mysql-use-bytes-for-string [355]</code>
Config File Options	<code>mysql-use-bytes-for-string [355], repl-mysql-use-bytes-for-string [355]</code>
Description	Transfer strings as their byte representation?
Value Type	string

`--mysql-xtrabackup-dir`

Option	<code>--mysql-xtrabackup-dir [355]</code>
Aliases	<code>--repl-mysql-xtrabackup-dir [355]</code>
Config File Options	<code>mysql-xtrabackup-dir [355], repl-mysql-xtrabackup-dir [355]</code>
Description	Directory to use for storing xtrabackup full & incremental backups
Value Type	string

10.7.12. N `tpm` Options

`--native-slave-takeover`

Option	<code>--native-slave-takeover [355]</code>
Aliases	<code>--repl-native-slave-takeover [355]</code>
Config File Options	<code>native-slave-takeover [355], repl-native-slave-takeover [355]</code>
Description	Takeover native replication
Value Type	string

`--net-ssh-option=key=value`

Option	<code>--net-ssh-option=key=value</code>
---------------	---

Config File Options	
Description	Set the Net::SSH option for remote system calls
Value Type	string

--no-deployment

Option	--no-deployment [356]
Config File Options	no-deployment [356]
Description	Skip deployment steps that create the install directory
Value Type	string

--no-validation

Option	--no-validation [356]
Config File Options	no-validation [356]
Description	Skip validation checks that run on each host
Value Type	string

--notice

Option	--notice [356]
Aliases	-n [356]
Config File Options	
Description	Display notice, warning and error messages
Value Type	string

10.7.13. O `tpm` Options

--optimize-row-events

Option	--optimize-row-events [356]	
Config File Options	optimize-row-events [356]	
Description	Enables or disables optimized row updates	
Value Type	boolean	
Valid Values	false	Disable optimized row updates
	true	

Optimized row updates bundle multiple row-based updates into a single [INSERT](#) or [UPDATE](#) statement. This increases the throughput of large batches of row-based updates.

--oracle-extractor-method

Option	--oracle-extractor-method [356]	
Aliases	--repl-oracle-extractor-method [356]	
Config File Options	oracle-extractor-method [356], repl-oracle-extractor-method [356]	
Description	Oracle extractor method	
Value Type	string	

--oracle-home

Option	--oracle-home [356]	
Aliases	--repl-oracle-home [356]	
Config File Options	oracle-home [356], repl-oracle-home [356]	
Description	Oracle Home	

Value Type	string
------------	--------

--oracle-redo-fetcher-host

Option	--oracle-redo-fetcher-host [357]
Aliases	--repl-oracle-redo-fetcher-host [357]
Config File Options	oracle-redo-fetcher-host [357], repl-oracle-redo-fetcher-host [357]
Description	Oracle host that will run the VMRR fetcher process
Value Type	string

--oracle-redo-miner-directory

Option	--oracle-redo-miner-directory [357]
Aliases	--repl-oracle-redo-miner-directory [357]
Config File Options	oracle-redo-miner-directory [357], repl-oracle-redo-miner-directory [357]
Description	Oracle Redo Miner Directory
Value Type	string

--oracle-redo-miner-transaction-fragment-size

Option	--oracle-redo-miner-transaction-fragment-size [357]
Aliases	--repl-oracle-redo-miner-transaction-fragment-size [357]
Config File Options	oracle-redo-miner-transaction-fragment-size [357], repl-oracle-redo-miner-transaction-fragment-size [357]
Description	Oracle Redo Miner Transaction Fragment Size
Value Type	numeric

--oracle-redo-password

Option	--oracle-redo-password [357]
Aliases	--repl-oracle-redo-password [357]
Config File Options	oracle-redo-password [357], repl-oracle-redo-password [357]
Description	Password for --oracle-redo-user
Value Type	string

--oracle-redo-replicate-tables

Option	--oracle-redo-replicate-tables [357]
Aliases	--repl-oracle-redo-replicate-tables [357]
Config File Options	oracle-redo-replicate-tables [357], repl-oracle-redo-replicate-tables [357]
Description	Oracle schema or tables to include in replication
Value Type	string

--oracle-redo-tablespace

Option	--oracle-redo-tablespace [357]
Aliases	--repl-oracle-redo-tablespace [357]
Config File Options	oracle-redo-tablespace [357], repl-oracle-redo-tablespace [357]
Description	Oracle tablespace to use with the redo reader
Value Type	string

--oracle-redo-user

Option	--oracle-redo-user [357]
--------	--------------------------

Aliases	--repl-oracle-redo-user [357]
Config File Options	oracle-redo-user [357], repl-oracle-redo-user [357]
Description	Oracle user to create for the redo reader
Value Type	string

--oracle-sys-user-password

Option	--oracle-sys-user-password [358]
Aliases	--repl-oracle-sys-user-password [358]
Config File Options	oracle-sys-user-password [358], repl-oracle-sys-user-password [358]
Description	Password for the Oracle SYS user
Value Type	string

--oracle-system-user-password

Option	--oracle-system-user-password [358]
Aliases	--repl-oracle-system-user-password [358]
Config File Options	oracle-system-user-password [358], repl-oracle-system-user-password [358]
Description	Password for the Oracle SYSTEM user
Value Type	string

10.7.14. P tpm Options

--pg-archive-timeout

Option	--pg-archive-timeout [358]
Aliases	--repl-pg-archive-timeout [358]
Config File Options	pg-archive-timeout [358], repl-pg-archive-timeout [358]
Description	Timeout for sending unfilled WAL buffers (data loss window)
Value Type	numeric

--pg-ctl

Option	--pg-ctl [358]
Aliases	--repl-pg-ctl [358]
Config File Options	pg-ctl [358], repl-pg-ctl [358]
Description	Path to the pg_ctl script
Value Type	filename

--pg-method

Option	--pg-method [358]
Aliases	--repl-pg-method [358]
Config File Options	pg-method [358], repl-pg-method [358]
Description	Postgres Replication method
Value Type	string

--pg-standby

Option	--pg-standby [358]
Aliases	--repl-pg-standby [358]
Config File Options	pg-standby [358], repl-pg-standby [358]
Description	Path to the pg_standby script

Value Type	filename
------------	----------

--postgresql-database

Option	--postgresql-database [359]
Aliases	--repl-postgresql-database [359]
Config File Options	postgresql-database [359], repl-postgresql-database [359]
Description	Name of the database to replicate
Value Type	string

--postgresql-enable-mysql2pgddl

Option	--postgresql-enable-mysql2pgddl [359]
Aliases	--repl-postgresql-enable-mysql2pgddl [359]
Config File Options	postgresql-enable-mysql2pgddl [359], repl-postgresql-enable-mysql2pgddl [359]
Description	Enable MySQL to PostgreSQL DDL dialect converting filter placeholder
Value Type	string
Valid Values	false

--postgresql-slonek

Option	--postgresql-slonek [359]
Aliases	--repl-postgresql-slonek [359]
Config File Options	postgresql-slonek [359], repl-postgresql-slonek [359]
Description	Path to the slonek executable
Value Type	filename

--postgresql-tables

Option	--postgresql-tables [359]
Aliases	--repl-postgresql-tables [359]
Config File Options	postgresql-tables [359], repl-postgresql-tables [359]
Description	Tables to replicate in form: schema1.table1,schema2.table2,...
Value Type	string

--preferred-path

Option	--preferred-path [359]
Config File Options	preferred-path [359]
Description	Additional command path
Value Type	filename

Specifies one or more additional directories that will be added before the current `PATH` environment variable when external commands are run from within the backup environment. This affects all external tools used by Tungsten Replication, 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` [359] information propagated to all remote servers within the `tpm` configuration. However, if the staging server is one of the servers to which you are deploying, the `PATH` must be manually updated.

--prefetch-enabled

Option	--prefetch-enabled [359]
Config File Options	prefetch-enabled [359]

Description	Should the replicator service be setup as a prefetch applier
Value Type	string

--prefetch-max-time-ahead

Option	--prefetch-max-time-ahead [360]
Config File Options	prefetch-max-time-ahead [360]
Description	Maximum number of seconds that the prefetch applier can get in front of the standard applier
Value Type	numeric

--prefetch-min-time-ahead

Option	--prefetch-min-time-ahead [360]
Config File Options	prefetch-min-time-ahead [360]
Description	Minimum number of seconds that the prefetch applier must be in front of the standard applier
Value Type	numeric

--prefetch-schema

Option	--prefetch-schema [360]
Config File Options	prefetch-schema [360]
Description	Schema to watch for timing prefetch progress
Value Type	string
Default	tungsten_
Valid Values	tungsten_
	tungsten_

--prefetch-sleep-time

Option	--prefetch-sleep-time [360]
Config File Options	prefetch-sleep-time [360]
Description	How long to wait when the prefetch applier gets too far ahead
Value Type	string

--preview

Option	--preview [360]
Aliases	-p [360]
Config File Options	
Description	Displays the help message and preview the effect of the command line options
Value Type	string

--privileged-master

Option	--privileged-master [360]
Config File Options	privileged-master [360]
Description	Does the login for the master database service have superuser privileges
Value Type	string

--privileged-slave

Option	--privileged-slave [360]
Config File Options	privileged-slave [360]
Description	Does the login for the slave database service have superuser privileges

Value Type	string
-------------------	--------

--profile file

Option	--profile file [361]
Config File Options	
Description	Sets name of config file
Value Type	string
Valid Values	tungsten.cfg

--profile-script

Option	--profile-script [361]
Config File Options	profile-script [361]
Description	Append commands to include env.sh in this profile script
Value Type	string

--property=key~=/match/replace/

Option	--property=key~=/match/replace/
Aliases	--property=key+=value, --property=key=value
Config File Options	
Description	Modify the value for key in any file that the configure script touches; key=value - Set key to value without evaluating template values or other rules; key+=value - Evaluate template values and then append value to the end of the line; key~/=match/replace/- Evaluate template values then execute the specified Ruby regex with sub. For example --property=replicator.key~=(.*)/somevalue,\1/ will prepend 'somevalue' before the template value for 'replicator.key'
Value Type	string

--protect-configuration-files

Option	--protect-configuration-files [361]
Config File Options	protect-configuration-files [361]
Description	When enabled, configuration files are protected to be only readable and updatable by the configured user
Value Type	string
Valid Values	false true

When enabled (default), the configuration that contain user, password and other information are configured so that they are only readable by the configured user. For example:

```
shell> ls -al /opt/continuent/tungsten/tungsten-replicator/conf/
total 148
drwxr-xr-x 2 tungsten mysql 4096 May 14 14:32 .
drwxr-xr-x 11 tungsten mysql 4096 May 14 14:32 ../
-rw-r--r-- 1 tungsten mysql 33 May 14 14:32 dynamic-alpha.role
-rw-r--r-- 1 tungsten mysql 5059 May 14 14:32 log4j.properties
-rw-r--r-- 1 tungsten mysql 3488 May 14 14:32 log4j-thl.properties
-rw-r--r-- 1 tungsten mysql 972 May 14 14:32 mysql-java-charsets.properties
-rw-r--r-- 1 tungsten mysql 420 May 14 14:32 replicator.service.properties
-rw-r----- 1 tungsten mysql 1590 May 14 14:35 services.properties
-rw-r----- 1 tungsten mysql 1590 May 14 14:35 .services.properties.orig
-rw-r--r-- 1 tungsten mysql 896 May 14 14:32 shard.list
-rw-r----- 1 tungsten mysql 43842 May 14 14:35 static-alpha.properties
-rw-r----- 1 tungsten mysql 43842 May 14 14:35 .static-alpha.properties.orig
-rw-r----- 1 tungsten mysql 5667 May 14 14:35 wrapper.conf
-rw-r----- 1 tungsten mysql 5667 May 14 14:35 .wrapper.conf.orig
```

When disabled, the files are readable by all users:

```
shell> ll /opt/continuent/tungsten/tungsten-replicator/conf/
total 148
drwxr-xr-x 2 tungsten mysql 4096 May 14 14:32 .
drwxr-xr-x 11 tungsten mysql 4096 May 14 14:32 ..
```

```
-rw-r--r-- 1 tungsten mysql 33 May 14 14:32 dynamic-alpha.role
-rw-r--r-- 1 tungsten mysql 5059 May 14 14:32 log4j.properties
-rw-r--r-- 1 tungsten mysql 3488 May 14 14:32 log4j-thl.properties
-rw-r--r-- 1 tungsten mysql 972 May 14 14:32 mysql-java-charsets.properties
-rw-r--r-- 1 tungsten mysql 420 May 14 14:32 replicator.service.properties
-rw-r--r-- 1 tungsten mysql 1590 May 14 14:32 services.properties
-rw-r--r-- 1 tungsten mysql 1590 May 14 14:32 .services.properties.orig
-rw-r--r-- 1 tungsten mysql 896 May 14 14:32 shard.list
-rw-r--r-- 1 tungsten mysql 43842 May 14 14:32 static-alpha.properties
-rw-r--r-- 1 tungsten mysql 43842 May 14 14:32 .static-alpha.properties.orig
-rw-r--r-- 1 tungsten mysql 5667 May 14 14:32 wrapper.conf
-rw-r--r-- 1 tungsten mysql 5667 May 14 14:32 .wrapper.conf.orig
```

10.7.15. Q tpm Options

--quiet

Option	--quiet [362]
Aliases	-q [362]
Config File Options	
Description	Only display warning and error messages
Value Type	string

10.7.16. R tpm Options

--redshift-dbname

Option	--redshift-dbname [362]
Aliases	--repl-redshift-dbname [362]
Config File Options	redshift-dbname [362], repl-redshift-dbname [362]
Description	NAme of the Redshift database to replicate into

--relay-directory

Option	--relay-directory [362]
Aliases	--repl-relay-directory [362]
Config File Options	relay-directory [362], repl-relay-directory [362]
Description	Directory for logs transferred from the master
Value Type	string
Default	{home directory}/relay
Valid Values	{home directory}/relay
	{home directory}/relay

--relay-enabled

Option	--relay-enabled [362]
Config File Options	relay-enabled [362]
Description	Should the replicator service be setup as a relay master
Value Type	string

--relay-source

Option	--relay-source [362]
Aliases	--dataservice-relay-source [362], --master-dataservice [362]
Config File Options	dataservice-relay-source [362], master-dataservice [362], relay-source [362]
Description	Dataservice name to use as a relay source

--remove-property=key

Option	--remove-property=key
Config File Options	
Description	Remove a corresponding --property argument.
Value Type	string

--replication-host

Option	--replication-host [363]
Aliases	--datasource-host [363], --repl-datasource-host [363]
Config File Options	datasource-host [363], repl-datasource-host [363], replication-host [363]
Description	Database server hostname
Value Type	string

--replication-password

Option	--replication-password [363]
Aliases	--datasource-password [363], --repl-datasource-password [363]
Config File Options	datasource-password [363], repl-datasource-password [363], replication-password [363]
Description	Database password
Value Type	string

--replication-port

Option	--replication-port [363]
Aliases	--datasource-port [363], --repl-datasource-port [363]
Config File Options	datasource-port [363], repl-datasource-port [363], replication-port [363]
Description	Database server port
Value Type	string

--replication-user

Option	--replication-user [363]
Aliases	--datasource-user [363], --repl-datasource-user [363]
Config File Options	datasource-user [363], repl-datasource-user [363], replication-user [363]
Description	Database login for Tungsten
Value Type	string

--reset

Option	--reset [363]
Config File Options	reset [363]
Description	Clear the current configuration before processing any arguments
Value Type	string

--rmi-port

Option	--rmi-port [363]
Aliases	--repl-rmi-port [363]
Config File Options	repl-rmi-port [363], rmi-port [363]
Description	Replication RMI listen port

--rmi-user

Option	<code>--rmi-user [363]</code>
Config File Options	<code>rmi-user [363]</code>
Description	The username for RMI authentication
Value Type	string

`--role`

Option	<code>--role [364]</code>
Aliases	<code>--repl-role [364]</code>
Config File Options	<code>repl-role [364], role [364]</code>
Description	What is the replication role for this service?
Value Type	string
Valid Values	master
	relay
	slave

`--router-gateway-port`

Option	<code>--router-gateway-port [364]</code>
Config File Options	<code>router-gateway-port [364]</code>
Description	The router gateway port
Value Type	string

`--router-jmx-port`

Option	<code>--router-jmx-port [364]</code>
Config File Options	<code>router-jmx-port [364]</code>
Description	The router jmx port
Value Type	string

10.7.17. `S tpm Options`

`--security-directory`

Option	<code>--security-directory [364]</code>
Config File Options	<code>security-directory [364]</code>
Description	Storage directory for the Java security/encryption files
Value Type	string

`--service-alias`

Option	<code>--service-alias [364]</code>
Aliases	<code>--dataservice-service-alias [364]</code>
Config File Options	<code>dataservice-service-alias [364], service-alias [364]</code>
Description	Replication alias of this dataservice
Value Type	string

`--service-type`

Option	<code>--service-type [364]</code>
Aliases	<code>--repl-service-type [364]</code>
Config File Options	<code>repl-service-type [364], service-type [364]</code>
Description	What is the replication service type?

Value Type	string
Valid Values	local
	remote

--skip-statemap

Option	--skip-statemap [365]
Config File Options	skip-statemap [365]
Description	Do not copy the cluster-home/conf/statemap.properties from the previous install
Value Type	string

--skip-validation-check String

Option	--skip-validation-check String [365]
Config File Options	
Description	Do not run the specified validation check. Validation checks are identified by the string included in the error they output.
Value Type	string

--skip-validation-warnings String

Option	--skip-validation-warnings String [365]
Config File Options	
Description	Do not display warnings for the specified validation check. Validation checks are identified by the string included in the warning they output.
Value Type	string

--slaves

Option	--slaves [365]
Aliases	--dataservice-slaves [365]
Config File Options	dataservice-slaves [365], slaves [365]
Description	What are the slaves for this dataservice?
Value Type	string

--start

Option	--start [365]
Config File Options	start [365]
Description	Start the services after configuration
Value Type	string

--start-and-report

Option	--start-and-report [365]
Config File Options	start-and-report [365]
Description	Start the services and report out the status after configuration
Value Type	string

--svc-allow-any-remote-service

Option	--svc-allow-any-remote-service [365]
Aliases	--repl-svc-allow-any-remote-service [365]
Config File Options	repl-svc-allow-any-remote-service [365], svc-allow-any-remote-service [365]
Description	Replicate from any service

Value Type	boolean
Valid Values	false
	true

--svc-applier-block-commit-interval

Option	--svc-applier-block-commit-interval [366]	
Aliases	--repl-svc-applier-block-commit-interval [366]	
Config File Options	repl-svc-applier-block-commit-interval [366], svc-applier-block-commit-interval [366]	
Description	Minimum interval between commits	
Value Type	string	
Valid Values	0 #d #h #m #s	When batch service is not enabled Number of days Number of hours Number of minutes Number of seconds

--svc-applier-block-commit-size

Option	--svc-applier-block-commit-size [366]
Aliases	--repl-svc-applier-block-commit-size [366]
Config File Options	repl-svc-applier-block-commit-size [366], svc-applier-block-commit-size [366]
Description	Applier block commit size (min 1)
Value Type	numeric

--svc-applier-filters

Option	--svc-applier-filters [366]
Aliases	--repl-svc-applier-filters [366]
Config File Options	repl-svc-applier-filters [366], svc-applier-filters [366]
Description	Replication service applier filters
Value Type	string

--svc-extractor-filters

Option	--svc-extractor-filters [366]
Aliases	--repl-svc-extractor-filters [366]
Config File Options	repl-svc-extractor-filters [366], svc-extractor-filters [366]
Description	Replication service extractor filters
Value Type	string

--svc-fail-on-zero-row-update

Option	--svc-fail-on-zero-row-update [366]
Aliases	--repl-svc-fail-on-zero-row-update [366]
Config File Options	repl-svc-fail-on-zero-row-update [366], svc-fail-on-zero-row-update [366]
Description	How should the replicator behave when a Row-Based Replication UPDATE does not affect any rows.
Value Type	string

--svc-parallelization-type

Option	--svc-parallelization-type [366]
Aliases	--repl-svc-parallelization-type [366]

Config File Options	<code>repl-svc-parallelization-type</code> [366], <code>svc-parallelization-type</code> [366]
Description	Method for implementing parallel apply
Value Type	string
Valid Values	disk
	memory
	none

--svc-remote-filters

Option	<code>--svc-remote-filters</code> [367]
Aliases	<code>--repl-svc-remote-filters</code> [367]
Config File Options	<code>repl-svc-remote-filters</code> [367], <code>svc-remote-filters</code> [367]
Description	Replication service remote download filters
Value Type	string

--svc-reposition-on-source-id-change

Option	<code>--svc-reposition-on-source-id-change</code> [367]
Aliases	<code>--repl-svc-reposition-on-source-id-change</code> [367]
Config File Options	<code>repl-svc-reposition-on-source-id-change</code> [367], <code>svc-reposition-on-source-id-change</code> [367]
Description	The master will come ONLINE from the current position if the stored source_id does not match the value in the static properties
Value Type	string

--svc-shard-default-db

Option	<code>--svc-shard-default-db</code> [367]
Aliases	<code>--repl-svc-shard-default-db</code> [367]
Config File Options	<code>repl-svc-shard-default-db</code> [367], <code>svc-shard-default-db</code> [367]
Description	Mode for setting the shard ID from the default db
Value Type	string
Valid Values	relaxed
	stringent

--svc-table-engine

Option	<code>--svc-table-engine</code> [367]
Aliases	<code>--repl-svc-table-engine</code> [367]
Config File Options	<code>repl-svc-table-engine</code> [367], <code>svc-table-engine</code> [367]
Description	Replication service table engine
Value Type	string
Default	innodb
Valid Values	innodb
	innodb

--svc-thl-filters

Option	<code>--svc-thl-filters</code> [367]
Aliases	<code>--repl-svc-thl-filters</code> [367]
Config File Options	<code>repl-svc-thl-filters</code> [367], <code>svc-thl-filters</code> [367]
Description	Replication service THL filters
Value Type	string

10.7.18. T `tpm` Options

`--target-dataservice`

Option	<code>--target-dataservice [368]</code>
Aliases	<code>--slave-dataservice [368]</code>
Config File Options	<code>slave-dataservice [368], target-dataservice [368]</code>
Description	Dataservice to use to determine the value of host configuration
Value Type	string

`--temp-directory`

Option	<code>--temp-directory [368]</code>
Config File Options	<code>temp-directory [368]</code>
Description	Temporary Directory
Value Type	string

`--template-file-help`

Option	<code>--template-file-help [368]</code>
Config File Options	<code>template-file-help [368]</code>
Description	Display the keys that may be used in configuration template files
Value Type	string

`--template-search-path`

Option	<code>--template-search-path [368]</code>
Config File Options	<code>template-search-path [368]</code>
Description	Adds a new template search path for configuration file generation
Value Type	filename

`--thl-directory`

Option	<code>--thl-directory [368]</code>
Aliases	<code>--repl-thl-directory [368]</code>
Config File Options	<code>repl-thl-directory [368], thl-directory [368]</code>
Description	Replicator log directory
Value Type	string
Default	{home directory}/thl
Valid Values	{home directory}/thl
	{home directory}/thl

`--thl-do-checksum`

Option	<code>--thl-do-checksum [368]</code>
Aliases	<code>--repl-thl-do-checksum [368]</code>
Config File Options	<code>repl-thl-do-checksum [368], thl-do-checksum [368]</code>
Description	Execute checksum operations on THL log files

`--thl-interface`

Option	<code>--thl-interface [368]</code>
Aliases	<code>--repl-thl-interface [368]</code>

Config File Options	repl-thl-interface [368] , thl-interface [368]
Description	Listen interface to use for THL operations
Value Type	string

`--thl-log-connection-timeout`

Option	--thl-log-connection-timeout [369]
Aliases	--repl-thl-log-connection-timeout [369]
Config File Options	repl-thl-log-connection-timeout [369] , thl-log-connection-timeout [369]
Description	Number of seconds to wait for a connection to the THL log
Value Type	numeric

`--thl-log-file-size`

Option	--thl-log-file-size [369]
Aliases	--repl-thl-log-file-size [369]
Config File Options	repl-thl-log-file-size [369] , thl-log-file-size [369]
Description	File size in bytes for THL disk logs
Value Type	numeric

`--thl-log-fsync`

Option	--thl-log-fsync [369]
Aliases	--repl-thl-log-fsync [369]
Config File Options	repl-thl-log-fsync [369] , thl-log-fsync [369]
Description	Fsync THL records on commit. More reliable operation but adds latency to replication when using low-performance storage
Value Type	string

`--thl-log-retention`

Option	--thl-log-retention [369]	
Aliases	--repl-thl-log-retention [369]	
Config File Options	repl-thl-log-retention [369] , thl-log-retention [369]	
Description	How long do you want to keep THL files.	
Value Type	string	
Valid Values	#d	Number of days
	#h	Number of hours
	#m	Number of minutes
	#s	Number of seconds
	7d	7 days

`--thl-protocol`

Option	--thl-protocol [369]
Aliases	--repl-thl-protocol [369]
Config File Options	repl-thl-protocol [369] , thl-protocol [369]
Description	Protocol to use for THL communication with this service
Value Type	string

`--topology`

Option	--topology [369]
---------------	----------------------------------

Aliases	--dataservice-topology [369]
Config File Options	dataservice-topology [369], topology [369]
Description	Replication topology for the dataservice Valid values are star,cluster-slave,master-slave,fan-in,clustered,cluster-alias,all-masters,direct
Value Type	string

--track-schema-changes

Option	--track-schema-changes [370]
Config File Options	track-schema-changes [370]
Description	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.
Value Type	string

10.7.19. U tpm Options

--user

Option	--user [370]
Config File Options	user [370]
Description	System User
Value Type	string

10.7.20. V tpm Options

--verbose

Option	--verbose [370]
Aliases	-v [370]
Config File Options	
Description	Display debug, info, notice, warning and error messages
Value Type	string

--vertica-database

Option	--vertica-database [370]
Aliases	--repl-vertica-database [370]
Config File Options	repl-vertica-database [370], vertica-database [370]
Description	Name of the database to replicate into
Value Type	string

10.7.21. W tpm Options

--witnesses

Option	--witnesses [370]
Aliases	--dataservice-witnesses [370]
Config File Options	dataservice-witnesses [370], witnesses [370]
Description	Witness hosts for the dataservice
Value Type	string

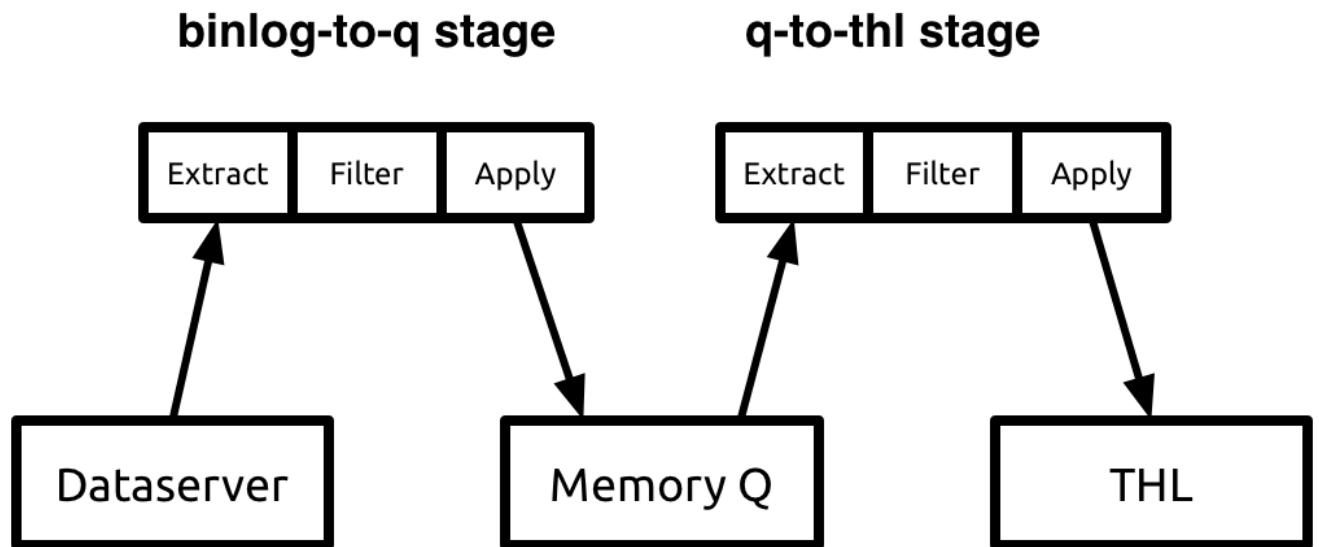
Chapter 11. Replication Filters

Filtering operates by applying the filter within one, or more, of the stages configured within the replicator. Stages are the individual steps that occur within a pipeline, that take information from a source (such as MySQL binary log) and write that information to an internal queue, the transaction history log, or apply it to a database. Where the filters are applied ultimately affect how the information is stored, used, or represented to the next stage or pipeline in the system.

For example, a filter that removed out all the tables from a specific database would have different effects depending on the stage it was applied. If the filter was applied on the 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 11.1, “Filters: Pipeline Stages on Masters”](#).

Figure 11.1. Filters: Pipeline Stages on Masters



Where:

- **binlog-to-q Stage**

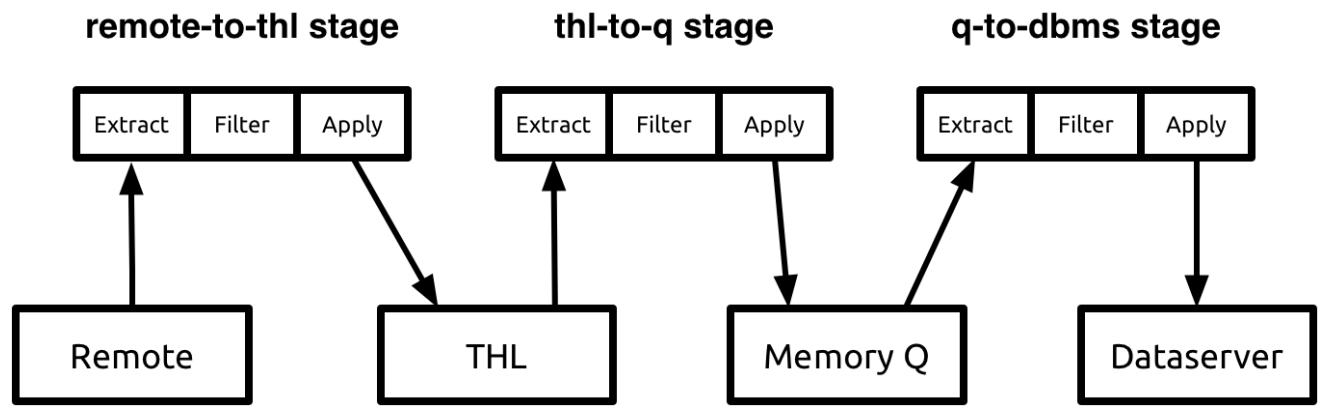
The **binlog-to-q** stage reads information from the MySQL binary log and stores the information within an in-memory queue.

- **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 11.2, “Filters: Pipeline Stages on Slaves”](#).

Figure 11.2. Filters: Pipeline Stages on Slaves



- [remote-to-thl Stage](#)

Remote THL information is read from a master datasource and written to a local file on disk.

- [thl-to-q Stage](#)

The THL information is read from the file on disk and stored in an in-memory queue.

- [q-to-dbms Stage](#)

The data from the in-memory queue is written to the target database.

Filters can be applied during any configured stage, and where the filter is applied alters the content and availability of the information. The staging and filtering mechanism can also be used to apply multiple filters to the data, altering content when it is read and when it is applied.

Where more than one filter is configured for a pipeline, each filter is executed in the order it appears in the configuration. For example, within the following fragment:

```
...
replicator.stage.binlog-to-q.filters=settostring,enumtostring,pkey,colnames
...
```

`settostring` is executed first, followed by `enumtostring`, `pkey` and `colnames`.

For certain filter combinations this order can be significant. Some filters rely on the information provided by earlier filters.

11.1. Enabling/Disabling Filters

A number of standard filter configurations are created and defined by default within the static properties file for the Tungsten Replicator configuration.

Filters can be enabled through `tpm` to update the filter configuration

- [--repl-svc-extractor-filters \[366\]](#)

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 \[367\]](#)

Apply the filter between the internal queue and when the transactions are written to the THL. (`q-to-thl`).

- [--repl-svc-applier-filters \[366\]](#)

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`:

```
shell> ./tools/tpm update alpha --hosts=host1,host2,host3 \
    --repl-svc-applier-filters=replicate \
    --property=replicator.filter.replicate.ignore=contacts
```

A bad filter configuration will not stop the replicator from starting, but the replicator will be placed into the [OFFLINE](#) [207] state.

To disable a previously enabled filter, empty the filter specification and (optionally) unset the corresponding property or properties. For example:

```
shell> ./tools/tpm update alpha --hosts=host1,host2,host3 \
    --repl-svc-applier-filters= \
    --remove-property=replicator.filter.replicate.ignore
```

Multiple filters can be applied on any stage, and the filters will be processes and called within the order defined within the configuration. For example, the following configuration:

```
shell> ./tools/tpm update alpha --hosts=host1,host2,host3 \
    --repl-svc-applier-filters=enumtostring,settostring,pkey \
    --remove-property=replicator.filter.replicate.ignore
```

The filters are called in order:

1. `enumtostring`
2. `settostring`
3. `pkey`

The order and sequence can be important if operations are being performed on the data and they are relied on later in the stage. For example, if data is being filtered by a value that exists in a `SET` column within the source data, the `settostring` filter must be defined before the data is filtered, otherwise the actual string value will not be identified.

Warning

In some cases, the filter order and sequence can also introduce errors. For example, when using the `pkey` filter and the `optimizeupdates` filters together, `pkey` may remove KEY information from the THL before `optimizeupdates` attempts to optimize the ROW event, causing the filter to raise a failure condition.

The currently active filters can be determined by using the `trepcctl status -name stages` command:

```
shell> trepcctl status -name stages
Processing status command (stages)...
...
NAME          VALUE
----          -----
applier.class : com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier
applier.name   : dbms
blockCommitRowCount: 10
committedMinSeqno: 3600
extractor.class : com.continuent.tungsten.replicator.thl.THLParallelQueueExtractor
extractor.name  : parallel-q-extractor
filter.0.class : com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter
filter.0.name   : mysqlsessions
filter.1.class : com.continuent.tungsten.replicator.filter.PrimaryKeyFilter
filter.1.name   : pkey
filter.2.class : com.continuent.tungsten.replicator.filter.BidiRemoteSlaveFilter
filter.2.name   : bidiSlave
name          : q-to-dbms
processedMinSeqno: -1
taskCount      : 5
Finished status command (stages)...
```

The above output is from a standard slave replication installation showing the default filters enabled. The filter order can be determined by the number against each filter definition.

11.2. Enabling Additional Filters

The Tungsten Replication 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 be created with the new filter definition in the configuration file.

Additional properties from this base should then be used. For example, to create a second rename filter definition called `custom`:

```
replicator.filter.rename.custom=com.continuent.tungsten.replicator.filter.RenameFilter
replicator.filter.rename.custom.definitionsFile=${replicator.home.dir}/samples/extensions/java/renamecustom.csv
```

The filter can be enabled against the desired stage using the filter name `custom`:

```
shell> ./tools/tpm configure \
    --repl-svc-applier-filters=custom
```

11.3. Filter Status

To determine which filters are currently being applied within a replicator, use the `treptctl status -name stages` command. This outputs a list of the current stages and their configuration. For example:

```
shell> treptctl status -name stages
Processing status command (stages)...
NAME          VALUE
----          -----
applier.class : com.continuent.tungsten.replicator.thl.THLStoreApplier
applier.name   : thl-applier
blockCommitRowCount: 1
committedMinSeqno : 15
extractor.class : com.continuent.tungsten.replicator.thl.RemoteTHLExtractor
extractor.name   : thl-remote
name           : remote-to-thl
processedMinSeqno : -1
taskCount      : 1
NAME          VALUE
----          -----
applier.class : com.continuent.tungsten.replicator.thl.THLParallelQueueApplier
applier.name   : parallel-q-applier
blockCommitRowCount: 10
committedMinSeqno : 15
extractor.class : com.continuent.tungsten.replicator.thl.THLStoreExtractor
extractor.name   : thl-extractor
name           : thl-to-q
processedMinSeqno : -1
taskCount      : 1
NAME          VALUE
----          -----
applier.class : com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier
applier.name   : dbms
blockCommitRowCount: 10
committedMinSeqno : 15
extractor.class : com.continuent.tungsten.replicator.thl.THLParallelQueueExtractor
extractor.name   : parallel-q-extractor
filter.0.class  : com.continuent.tungsten.replicator.filter.TimeDelayFilter
filter.0.name    : delay
filter.1.class  : com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter
filter.1.name    : mysqlsessions
filter.2.class  : com.continuent.tungsten.replicator.filter.PrimaryKeyFilter
filter.2.name    : pkey
name           : q-to-dbms
processedMinSeqno : -1
taskCount      : 5
Finished status command (stages)...
```

In the output, the filters applied to the applier stage are shown in the last block of output. Filters are listed in the order in which they appear within the configuration.

For information about the filter operation and any modifications or changes made, check the `trepsvc.log` log file.

11.4. Filter Reference

The different filter types configured and available within the replicate are designed to provide a number of different functionality and operations. Since the information exchanged through the THL system contains a copy of the statement or the row data that is being updated, the filters allow schemas, table and column names, as well as actual data to be converted at the stage in which they are applied.

Filters are identified according to the underlying Java class that defines their operation. For different filters, further configuration and naming is applied according to the templates used when Tungsten Replication 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.

Filter	Type	Description
BidiRemoteSlave-Filter	Content	Suppresses events that originated on the local service (required for correct slave operation)
BuildAuditTable	Auditing	Builds an audit table of changes for specified schemas and tables
BuildIndexTable	Transformation	Merges multiple schemas into a single schema
CaseMappingFilter	Transformation	Transforms schema, table and column names to upper or lower case
CDCMetadataFilter	Auditing	Records change data capture for transactions to a separate change table (auditing)
ColumnNameFilter	Validation	Adds column name information to row-based replication events
ConsistencyCheck-Filter	Validation	Adds consistency checking to events
DatabaseTransformFilter	Transformation	Transforms database or table names using regular expressions
DummyFilter	Miscellaneous	Allows for confirmation of filter configuration
EnumToStringFilter	Content	Updates enumerated values to their string-based equivalent
EventMetadataFilter	Content	Filters events based on metadata; used by default within sharding and multi-master topologies
HeartbeatFilter	Validation	Detects heartbeat events on masters or slaves
JavaScriptFilter	Miscellaneous	Enables filtering through custom JavaScripts
LoggingFilter	Logging	Logs filtered events through the standard replicator logging mechanism
MySQLSessionSupportFilter	Content	Filters transactions for session specific temporary tables and variables

Filter	Type	Description
OptimizeUpdates-Filter	Optimization	Optimizes update statements where the current and updated value are the same
PrimaryKeyFilter	Optimization	Used during row-based replication to optimize updates using primary keys
PrintEventFilter	Logging	Outputs transaction event information to the replication logging system
RenameFilter	Transformation	Advanced schema, table and column-based renaming
ReplicateColumns-Filter	Content	Removes selected columns from row-based transaction data
ReplicateFilter	Content	Selects or ignores specification schemas and/or databases
SetToStringFilter	Content	Converts integer values in <code>SET</code> datatypes to string values
ShardFilter	Content	Used to enforce database schema sharding between specific masters
TimeDelayFilter	Miscellaneous	Delays transactions until a specific point in time has passed

In the following reference sections:

- **Pre-configured filter name** is the filter name that can be used against a stage without additional configuration.
- **Property prefix** is the prefix string for the filter to be used when assigning property values.
- **Classname** is the Java class name of the filter.
- **Parameter** is the name of the filter parameter can be set as a property within the configuration.
- **Data compatibility** indicates whether the filter is compatible with row-based events, statement-based events, or both.

11.4.1. `ansiquotes.js` Filter

The `ansiquotes` filter operates by inserting an SQL mode change to `ANSI_QUOTES` into the replication stream before a statement is executed, and returning to an empty SQL mode.

Pre-configured filter name	<code>ansiquotes</code>		
JavaScript Filter File	<code>tungsten-replicator/support/filters-javascript/ansiquotes.js</code>		
Property prefix	<code>replicator.filter.ansiquotes</code>		
Stage compatibility	<code>binlog-to-q</code>		
<code>tpm</code> Option compatibility	<code>--svc-extractor-filters [366]</code>		
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description

This changes a statement such as:

```
INSERT INTO notepad VALUES ('message',0);
```

To:

```
SET sql_mode='ANSI_QUOTES';
INSERT INTO notepad VALUES ('message',0);
SET sql_mode='';
```

This is achieved within the JavaScript by processing the incoming events and adding a new statement before the first `DBMSData` object in each event:

```
query = "SET sql_mode='ANSI_QUOTES'";
newStatement = new com.continuent.tungsten.replicator.dbms.StatementData(
    query,
    null,
    null
);
data.add(0, newStatement);
```

A corresponding statement is appended to the end of the event:

```
query = "SET sql_mode=''";
newStatement = new com.continuent.tungsten.replicator.dbms.StatementData(
```

```

        query,
        null,
        null
    );
data.add(data.size(), newStatement);

```

11.4.2. BidiRemoteSlave (BidiSlave) Filter

The BidiRemoteSlaveFilter is used by Tungsten Replicator to prevent statements that originated from this service (i.e. where data was extracted), being re-applied to the database. This is a requirement for replication to prevent data that may be transferred between hosts being re-applied, particularly in multi-master and other bi-directional replication deployments.

Pre-configured filter name	<code>bidiSlave</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.BidiRemoteSlaveFilter</code>		
Property prefix	<code>replicator.filter.bidiSlave</code>		
Stage compatibility			
tpm Option compatibility			
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description
<code>localServiceName</code>	string	<code> \${local.service.name}</code>	Local service name of the service that reads the binary log
<code>allowBidiUnsafe</code>	boolean	<code>false</code>	If true, allows statements that may be unsafe for bi-directional replication
<code>allowAnyRemoteService</code>	boolean	<code>false</code>	If true, allows statements from any remote service, not just the current service

The filter works by comparing the server ID of the THL event that was created when the data was extracted against the server ID of the current server.

When deploying through the `tpm` service the filter is automatically enabled for remote 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.

11.4.3. `breadcrumbs.js` Filter

The `breadcrumbs` filter records regular 'breadcrumb' points into a MySQL table for systems that do not have global transaction IDs. This can be useful if recovery needs to be made to a specific point. The example also shows how metadata information for a given event can be updated based on the information from a table.

Pre-configured filter name	<code>ansiQuotes</code>		
JavaScript Filter File	<code>tungsten-replicator/support/filters-javascript/breadcrumbs.js</code>		
Property prefix	<code>replicator.filter.breadcrumbs</code>		
Stage compatibility	<code>binlog-to-q</code>		
tpm Option compatibility	<code>--svc-extractor-filters [366]</code>		
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description
<code>server_id</code>	numeric	(none)	MySQL server ID of the current host

To use the filter:

1. A table is created and populated with one more rows on the master server. For example:

```

CREATE TABLE `tungsten_svcl`.`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_svcl.breadcrumbs(id, counter) values(@@server_id, 1);

```

2. Now set an event to update the table regularly. For example, within MySQL an event can be created for this purpose:

```
CREATE EVENT breadcrumbs_refresh
  ON SCHEDULE EVERY 5 SECOND
  DO
    UPDATE tungsten_svc1.breadcrumbs SET counter=counter+1;
SET GLOBAL event_scheduler = ON;
```

The filter will extract the value of the counter each time it sees to the table, and then mark each transaction with a particular server ID with the counter value plus an offset. For convenience we assume row replication is enabled.

If you need to failover to another server that has different logs, you can figure out the restart point by looking in the THL for the breadcrumb metadata on the last transaction. Use this to search the binary logs on the new server for the correct restart point.

The filter itself work in two stages, and operates because the JavaScript instance is persistent as long as the Replicator is running. This means that data extracted during replication stays in memory and can be applied to later transactions. Hence the breadcrumb ID and offset information can be identified and used on each call to the filter function.

The first part of the filter event identifies the breadcrumb table and extracts the identified breadcrumb counter:

```
if (table.compareToIgnoreCase("breadcrumbs") == 0)
{
  columnValues = oneRowChange.getColumnValues();
  for (row = 0; row < columnValues.size(); row++)
  {
    values = columnValues.get(row);
    server_id_value = values.get(0);
    if (server_id == null || server_id == server_id_value.getValue())
    {
      counter_value = values.get(1);
      breadcrumb_counter = counter_value.getValue();
      breadcrumb_offset = 0;
    }
  }
}
```

The second part updates the event metadata using the extracted breadcrumb information:

```
topLevelEvent = event.getDBMSEvent();
if (topLevelEvent != null)
{
  xact_server_id = topLevelEvent.getMetadataOptionValue("mysql_server_id");
  if (server_id == xact_server_id)
  {
    topLevelEvent.setMetaDataTable("breadcrumb_counter", breadcrumb_counter);
    topLevelEvent.setMetaDataTable("breadcrumb_offset", breadcrumb_offset);
  }
}
```

To calculate the offset (i.e. the number of events since the last breadcrumb value was extracted), the filter determines if the event was the last fragment processed, and updates the offset counter:

```
if (event.getLastFrag())
{
  breadcrumb_offset = breadcrumb_offset + 1;
}
```

11.4.4. 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.

Pre-configured filter name	<i>Not defined</i>		
Classname	<code>com.continuent.tungsten.replicator.filter.BuildAuditTable</code>		
Property prefix	<code>replicator.filter.bidiSlave</code>		
Stage compatibility			
<code>tpm</code> Option compatibility			
Data compatibility	Row events		
Parameters			
Parameter	Type	Default	Description
<code>targetTableName</code>	string		Name of the table where audit information will be stored

11.4.5. BuildIndexTable Filter

Pre-configured filter name	<code>buildindextable</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.BuildIndexTable</code>		
Property prefix	<code>replicator.filter.buildindextable</code>		
Stage compatibility			
<code>tpm</code> Option compatibility			
Data compatibility	Row events		
Parameters			
Parameter	Type	Default	Description
<code>target_schema_name</code>	string	<code>test</code>	Name of the schema where the new index information will be created

11.4.6. CaseMapping (CaseTransform) Filter

Pre-configured filter name	<code>casetransform</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.CaseMappingFilter</code>		
Property prefix	<code>replicator.filter.casetransform</code>		
Stage compatibility			
<code>tpm</code> Option compatibility			
Data compatibility	Any Event		
Parameters			
Parameter	Type	Default	Description
<code>to_upper_case</code>	boolean	<code>true</code>	If true, converts object names to upper case; if false converts them to lower case

11.4.7. CDCMetadata (CustomCDC) Filter

Pre-configured filter name	<code>customcdc</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.CDCMetadataFilter</code>		
Property prefix	<code>replicator.filter.customcdc</code>		
Stage compatibility			
<code>tpm</code> Option compatibility			
Data compatibility	Row events		
Parameters			
Parameter	Type	Default	Description
<code>cdcColumnsAtFront</code>	boolean	<code>false</code>	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
<code>schemaNameSuffix</code>		string	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
<code>tableNameSuffix</code>	string		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
<code>toSingleSchema</code>	string		Creates and writes CDC data within a single schema
<code>sequenceBeginning</code>	numeric	1	Sets the sequence number of the CDC data. The sequence is used to identify individual changesets in the CDC

11.4.8. 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.

Pre-configured filter name	<code>colnames</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.ColumnNameFilter</code>		
Property prefix	<code>replicator.filter.colnames</code>		
Stage compatibility	<code>binlog-to-q</code>		
<code>tpm</code> Option compatibility	<code>--svc-extractor-filters [366]</code>		
Data compatibility	Row events		
Parameters			
Parameter	Type	Default	Description
<code>user</code>	string	<code> \${replicator.global.extract.db.user}</code>	The username for the connection to the database for looking up column definitions
<code>password</code>	string	<code> \${replicator.global.extract.db.password}</code>	The password for the connection to the database for looking up column definitions
<code>url</code>	string	<code>jdbc:mysql:thin://\${replicator.global.extract.db.host}:\${replicator.global.extract.db.port}/\${replicator.schema}?createDB=true</code>	JDBC URL of the database connection to use for looking up column definitions
<code>addSignedFlag</code>	boolean	<code>true</code>	Determines whether the signed flag information for columns should be added to the metadata for each column.
<code>ignoreMissingTables</code>	boolean	<code>true</code>	When true, tables that do not exist will not trigger metadata and column names to be added to the THL data.

Note

This filter is designed to be used for testing and with heterogeneous replication where the field name information can be used to construct and build target data structures.

The filter is required for the correct operation of heterogeneous replication, for example when replicating to MongoDB. The filter works by using the replicator username and password to access the underlying database and obtain the table definitions. The table definition information is cached within the replication during operation to improve performance.

When extracting data from the binary log using row-based replication, the column names for each row of changed data are added to the THL.

Enabling this filter changes the THL data from the following example, shown without the column names:

```
SEQ# = 27 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 18:29:38.0
- EPOCH# = 11
- EVENTID = mysql-bin.000012:0000000000004369:0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
- ACTION = INSERT
- SCHEMA = test
- TABLE = sales
- ROW# = 0
- COL(1: ) = 1
- COL(2: ) = 23
- COL(3: ) = 45
- COL(4: ) = 45000.00
```

To a version where the column names are included as part of the THL record:

```
SEQ# = 43 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 18:34:18.0
- EPOCH# = 28
- EVENTID = mysql-bin.000012:0000000000006814:0
- SOURCEID = host31
```

```

- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
  - ACTION = INSERT
  - SCHEMA = test
  - TABLE = sales
  - ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 23
  - COL(3: city) = 45
  - COL(4: value) = 45000.00

```

When the row-based data is applied to a non-MySQL database the column name information is used by the applier to specify the column, or they key when the column and value is used as a key/value pair in a document-based store.

11.4.9. ConsistencyCheck Filter

Pre-configured filter name	<i>Not defined</i>
Classname	<code>com.continuent.tungsten.replicator.consistency.ConsistencyCheckFilter</code>
Property prefix	<i>Not defined</i>
Stage compatibility	
<code>tpm</code> Option compatibility	
Data compatibility	Any event
Parameters	
(none)	

11.4.10. DatabaseTransform (dbtransform) Filter

Pre-configured filter name	<code>dbtransform</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.DatabaseTransformFilter</code>		
Property prefix	<code>replicator.filter.dbtransform</code>		
Stage compatibility			
<code>tpm</code> Option compatibility			
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description
<code>transformTables</code>	<code>boolean</code>	<code>false</code>	If set to true, forces the rename transformations to operate on tables, not databases
<code>from_regex1</code>	<code>string</code>	foo	The search regular expression to use when renaming databases or tables (group 1); corresponds to <code>to_regex1</code>
<code>to_regex1</code>	<code>string</code>	bar	The replace regular expression to use when renaming databases or tables (group 1); corresponds to <code>from_regex1</code>
<code>from_regex2</code>	<code>string</code>		The search regular expression to use when renaming databases or tables (group 2); corresponds to <code>to_regex1</code>
<code>to_regex2</code>	<code>string</code>		The replace regular expression to use when renaming databases or tables (group 2); corresponds to <code>from_regex1</code>
<code>from_regex3</code>	<code>string</code>		The search regular expression to use when renaming databases or tables (group 3); corresponds to <code>to_regex1</code>
<code>to_regex3</code>	<code>string</code>		The replace regular expression to use when renaming databases or tables (group 3); corresponds to <code>from_regex1</code>

11.4.11. `dbrename.js` Filter

The `dbrename` JavaScript filter renames database (schemas) using two parameters from the properties file, the `dbsource` and `dbtarget`. Each event is then processed, and the statement or row based schema information is updated to `dbtarget` when the `dbsource` schema is identified.

Pre-configured filter name	<code>dbrename</code>		
JavaScript Filter File	<code>tungsten-replicator/support/filters-javascript/dbrename.js</code>		
Property prefix	<code>replicator.filter.dbrename</code>		
Stage compatibility	<code>binlog-to-q</code>		
<code>tpm</code> Option compatibility	<code>--svc-extractor-filters [366]</code>		
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description
<code>dbsource</code>	<code>string</code>	(none)	Source table name (database/table to be renamed)
<code>dbtarget</code>	<code>string</code>	(none)	New database/table name

To configure the filter you would add the following to your properties:

```
replicator.filter.dbrename=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.dbrename.script=${replicator.home.dir}/samples/extensions/javascript/dbrename.js
replicator.filter.dbrename.dbsource=SOURCE
replicator.filter.dbrename.dbtarget=TEST
```

The operation of the filter is straightforward, because the schema name is exposed and settable within the statement and row change objects:

```
function filter(event)
{
    sourceName = filterProperties.getString("dbsource");
    targetName = filterProperties.getString("dbtarget");

    data = event.getData();

    for(i=0;i<data.size();i++)
    {
        d = data.get(i);

        if(d instanceof
            com.continuent.tungsten.replicator.dbms.StatementData)
        {
            if(d.getDefaultSchema() != null &&
                d.getDefaultSchema().compareTo(sourceName)==0)
            {
                d.setDefaultSchema(targetName);
            }
        }
        else if(d instanceof
            com.continuent.tungsten.replicator.dbms.RowChangeData)
        {
            rowChanges = data.get(i).getRowChanges();

            for(j=0;j<rowChanges.size();j++)
            {
                oneRowChange = rowChanges.get(j);

                if(oneRowChange.getSchemaName().compareTo(sourceName)==0)
                {
                    oneRowChange.setSchemaName(targetName);
                }
            }
        }
    }
}
```

11.4.12. `dbselector.js` Filter

Filtering only a single database schema can be useful when you want to extract a single schema for external processing, or for sharding information across multiple replication targets. The `dbselector` filter deletes all statement and row changes, except those for the selected table. To configure, the `db` parameter to the filter configuration specifies the schema to be replicated.

Pre-configured filter name	<code>dbselector</code>		
JavaScript Filter File	<code>tungsten-replicator/support/filters-javascript/dbselector.js</code>		
Property prefix	<code>replicator.filter.dbselector</code>		
Stage compatibility	<code>binlog-to-q, q-to-thl, q-to-dbms</code>		
tpm Option compatibility	<code>--svc-extractor-filters [366], --svc-applier-filters [366]</code>		
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description
<code>db</code>	<code>string</code>	(none)	Database to be selected

Within the filter, statement changes look for the schema in the `StatementData` object and remove it from the array:

```
if (d instanceof com.continuent.tungsten.replicator.dbms.StatementData)
{
    if(d.getDefaultSchema().compareTo(db)!=0)
    {
        data.remove(i);
        i--;
    }
}
```

Because entries are being removed from the list of statements, the iterator used to process each item must be explicitly decremented by 1 to reset the counter back to the new position.

Similarly, when looking at row changes in the `RowChangeData`:

```
else if(d instanceof com.continuent.tungsten.replicator.dbms.RowChangeData)
{
    rowChanges = data.get(i).getRowChanges();
    for(j=0;j<rowChanges.size();j++)
    {
        oneRowChange = rowChanges.get(j);
        if(oneRowChange.getSchemaName().compareTo(db)!=0)
        {
            rowChanges.remove(j);
            j--;
        }
    }
}
```

11.4.13. `dbupper.js` Filter

The `dbupper` filter changes the case of the schema name for all schemas to uppercase. The schema information is easily identified in the statement and row based information, and therefore easy to update.

Pre-configured filter name	<code>dbupper</code>		
JavaScript Filter File	<code>tungsten-replicator/support/filters-javascript/dbupper.js</code>		
Property prefix	<code>replicator.filter.dbupper</code>		
Stage compatibility	<code>binlog-to-q</code>		
tpm Option compatibility	<code>--svc-extractor-filters [366], --svc-applier-filters [366]</code>		
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description
<code>from</code>	<code>string</code>	(none)	Database name to be converted to uppercase

For example, within statement data:

```
from = d.getDefaultSchema();
if (from != null)
{
    to   = from.toUpperCase();
    d.setDefaultSchema(to);
}
```

11.4.14. `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.

Pre-configured filter name	<code>dropcolumn</code>		
JavaScript Filter File	tungsten-replicator/support/filters-javascript/dropcolumn.js		
Property prefix	<code>replicator.filter.dropcolumn</code>		
Stage compatibility	<code>binlog-to-q, q-to-dbms</code>		
<code>tpm</code> Option compatibility	<code>--svc-extractor-filters</code> [366], <code>--svc-applier-filters</code> [366]		
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description
<code>definitionsFile</code>	Filename	<code>~/dropcolumn.json</code>	Location of the definitions file for dropping columns

The filter is available by default as `dropcolumn`, and the filter is configured through a JSON file that defines the list of columns to be dropped. The filter relies on the `colnames` filter being enabled.

To enable the filter:

```
shell> tpm update --svc-extractor-filters=colnames,dropcolumn \
    --property=replicator.filter.dropcolumn.definitionsFile=/opt/continuent/share/dropcolumn.json
```

A sample configuration file is provided in `/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"
        ]
    }
}]
```

```

        "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`

11.4.15. `dropcomments.js` Filter

The `dropcomments` filter removes comments from statements within the event data.

Pre-configured filter name	<code>dropcomments</code>		
JavaScript Filter File	<code>tungsten-replicator/support/filters-javascript/dropcomments.js</code>		
Property prefix	<code>replicator.filter.dropcomments</code>		
Stage compatibility	<code>binlog-to-q, q-to-dbms</code>		
tpm Option compatibility	<code>--svc-extractor-filters [366], --svc-applier-filters [366]</code>		
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description

Row changes do not have comments, so the filter only has to change the statement information, which is achieved by using a regular expression:

```

sqlOriginal = d.getQuery();
sqlNew = sqlOriginal.replaceAll("//*(?:.|[\n\r])*?//", "");
d.setQuery(sqlNew);
```

To handle the case where the statement could only be a comment, the statement is removed:

```

if(sqlNew.trim().length()==0)
{
    data.remove(i);
    i--;
}
```

11.4.16. `dropmetadata.js` Filter

All events within the replication stream contain metadata about each event. This information can be individual processed and manipulated. The `dropmetadata` filter removes specific metadata from each event, configured through the `option` parameter to the filter.

Pre-configured filter name	<code>dropmetadata</code>		
JavaScript Filter File	<code>tungsten-replicator/support/filters-javascript/dropmetadata.js</code>		
Property prefix	<code>replicator.filter.ansiQuotes</code>		
Stage compatibility	<code>binlog-to-q, q-to-dbms</code>		
tpm Option compatibility	<code>--svc-extractor-filters [366], --svc-applier-filters [366]</code>		
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description
<code>option</code>	<code>string</code>	(none)	Name of the metadata field to be dropped

Metadata information can be processed at the event top-level:

```

metaData = event.getDBMSEvent().getMetadata();
for(m = 0; m < metaData.size(); m++)
{
    option = metaData.get(m);
```

```

        if(option.getOptionName().compareTo(optionName)==0)
        {
            metaData.remove(m);
            break;
        }
    }
}

```

11.4.17. `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	<code>dropstatementdata</code>		
JavaScript Filter File	tungsten-replicator/support/filters-javascript/dropstatementdata.js		
Property prefix	<code>replicator.filter.dropstatementdata</code>		
Stage compatibility	<code>binlog-to-q, q-to-dbms</code>		
<code>tpm</code> Option compatibility	<code>--svc-extractor-filters</code> [366], <code>--svc-applier-filters</code> [366]		
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description

This is achieved by checking for statements, and then removing them from the event:

```

data = event.getData();

for(i = 0; i < data.size(); i++)
{
    d = data.get(i);

    if(d instanceof com.continuent.tungsten.replicator.dbms.StatementData)
    {
        data.remove(i);
        i--;
    }
}

```

11.4.18. Dummy Filter

Pre-configured filter name	<code>dummy</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.DummyFilter</code>		
Property prefix	<code>replicator.filter.dummy</code>		
Stage compatibility			
<code>tpm</code> Option compatibility			
Data compatibility	Any event		
Parameters			
(none)			

11.4.19. `EnumToString` Filter

The `EnumToString` filter translates `ENUM` datatypes within MySQL tables into their string equivalent within the THL.

Pre-configured filter name	<code>enumtostring</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.EnumToStringFilter</code>		
Property prefix	<code>replicator.filter.enumtostring</code>		
Stage compatibility	<code>binlog-to-q</code>		
<code>tpm</code> Option compatibility	<code>--repl-svc-extractor-filters</code> [366]		
Data compatibility	Row events		
Parameters			
Parameter	Type	Default	Description

<code>user</code>	string	<code>`\${replicator.global.extract.db.user}`</code>	The username for the connection to the database for looking up column definitions
<code>password</code>	string	<code>`\${replicator.global.extract.db.password}`</code>	The password for the connection to the database for looking up column definitions
<code>url</code>	string	<code>jdbc:mysql:thin://\${replicator.global.extract.db.host}:\${replicator.global.extract.db.port}/\${replicator.schema}?createDB=true</code>	JDBC URL of the database connection to use for looking up column definitions

The [EnumToString](#) filter should be used with heterogeneous replication to ensure that the data is represented as the string value, not the internal numerical representation.

In the THL output below, the table has a `ENUM` column, `country`:

```
mysql> describe salesadv;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id   | int(11) | NO  | PRI | NULL    | auto_increment |
| country | enum('US','UK','France','Australia') | YES |     | NULL    |          |
| city  | int(11) | YES |     | NULL    |          |
| salesman | set('Alan','Zachary') | YES |     | NULL    |          |
| value | decimal(10,2) | YES |     | NULL    |          |
+-----+-----+-----+-----+-----+
```

When extracted in the THL, the representation uses the internal value (for example, 1 for the first enumerated value). This can be seen in the THL output below.

```
SEQ# = 138 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:09:35.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:0000000000021434:0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
- ACTION = INSERT
- SCHEMA = test
- TABLE = salesadv
- ROW# = 0
- COL(1: id) = 2
- COL(2: country) = 1
- COL(3: city) = 8374
- COL(4: salesman) = 1
- COL(5: value) = 35000.00
```

For the `country` column, the corresponding value in the THL is 1. With the [EnumToString](#) filter enabled, the value is expanded to the corresponding string value:

```
SEQ# = 121 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:05:14.0
- EPOCH# = 102
- EVENTID = mysql-bin.000012:0000000000018866:0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
- ACTION = INSERT
- SCHEMA = test
- TABLE = salesadv
- ROW# = 0
- COL(1: id) = 1
- COL(2: country) = US
- COL(3: city) = 8374
- COL(4: salesman) = Alan
- COL(5: value) = 35000.00
```

The information is critical when applying the data to a dataserver that is not aware of the table definition, such as when replicating to Oracle or MongoDB.

The examples here also show the [Section 11.4.34, “SetToString Filter”](#) and [Section 11.4.8, “ColumnName Filter”](#) filters.

11.4.20. EventMetadata Filter

Pre-configured filter name	<code>eventmetadata</code>
Classname	<code>com.continuent.tungsten.replicator.filter.EventMetadataFilter</code>
Property prefix	<code>replicator.filter.eventmetadata</code>
Stage compatibility	
tpm Option compatibility	
Data compatibility	Row events
Parameters	
(none)	

11.4.21. `foreignkeychecks.js` Filter

The `foreignkeychecks` filter switches off foreign key checks for statements using the following statements:

```
CREATE TABLE
DROP TABLE
ALTER TABLE
RENAME TABLE
```

Pre-configured filter name	<code>foreignkeychecks</code>		
JavaScript Filter File	<code>tungsten-replicator/support/filters-javascript/foreignkeychecks.js</code>		
Property prefix	<code>replicator.filter.foreignkeychecks</code>		
Stage compatibility	<code>binlog-to-q, q-to-dbms</code>		
tpm Option compatibility	<code>--svc-extractor-filters [366], --svc-applier-filters [366]</code>		
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description

The process checks the statement data and parses the content of the SQL statement by first trimming any extraneous space, and then converting the statement to upper case:

```
upCaseQuery = d.getQuery().trim().toUpperCase();
```

Then comparing the string for the corresponding statement types:

```
if(upCaseQuery.startsWith("CREATE TABLE") ||
   upCaseQuery.startsWith("DROP TABLE") ||
   upCaseQuery.startsWith("ALTER TABLE") ||
   upCaseQuery.startsWith("RENAME TABLE"))
{
}
```

If they match, a new statement is inserted into the event that disables foreign key checks:

```
query = "SET foreign_key_checks=0";
newStatement = new com.continuent.tungsten.replicator.dbms.StatementData(
    d.getDefaultSchema(),
    null,
    query
);
data.add(0, newStatement);
i++;
```

The use of `0` in the `add()` method inserts the new statement before the others within the current event.

11.4.22. Heartbeat Filter

Pre-configured filter name	(none)
Classname	<code>com.continuent.tungsten.replicator.filter.HeartbeatFilter</code>
Property prefix	(none)
Stage compatibility	
tpm Option compatibility	

Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description
<code>heartbeatInterval</code>	numeric	3000	Interval in milliseconds when a heartbeat event is inserted into the THL

11.4.23. `insertonly.js` Filter

The `insertonly` filter filters events to only include ROW-based events using `INSERT`.

Pre-configured filter name	<code>insertonly</code>		
JavaScript Filter File	<code>tungsten-replicator/support/filters-javascript/insertonly.js</code>		
Property prefix	<code>replicator.filter.insertonly</code>		
Stage compatibility	<code>q-to-dbms</code>		
<code>tpm</code> Option compatibility	<code>--svc-applier-filters</code> [366]		
Data compatibility	Row events		
Parameters			
Parameter	Type	Default	Description

This is achieved by examining each row and removing row changes that do not match the `INSERT` action type:

```
if(oneRowChange.getAction()!="INSERT")
{
    rowChanges.remove(j);
    j--;
}
```

11.4.24. Logging Filter

Pre-configured filter name	<code>logger</code>					
Classname	<code>com.continuent.tungsten.replicator.filter.LoggingFilter</code>					
Property prefix	<code>replicator.filter.logger</code>					
Stage compatibility						
<code>tpm</code> Option compatibility						
Data compatibility	Any event					
Parameters						
(none)						

11.4.25. MySQLSessionSupport (mysqlsessions) Filter

Pre-configured filter name	<code>mysqlsessions</code>					
Classname	<code>com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter</code>					
Property prefix	<code>replicator.filter.mysqlsession</code>					
Stage compatibility						
<code>tpm</code> Option compatibility						
Data compatibility	Any event					
Parameters						
(none)						

11.4.26. NetworkClient Filter

The `NetworkClientFilter` processes data in selected columns

Pre-configured filter name	<code>networkclient</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.NetworkClientFilter</code>		
Property prefix	<code>replicator.filter.networkclient</code>		
Stage compatibility	Any		
tpm Option compatibility	<code>--svc-extractor-filters</code> [366], <code>--svc-thl-filters</code> [367], <code>--svc-applier-filters</code> [366]		
Data compatibility	Row events		
Parameters			
Parameter	Type	Default	Description
<code>definitionsFile</code>	pathname	<code> \${replicator.home.dir}/samples/extensions/java/network-client.json</code>	The name of a file containing the definitions for how columns should be processed by filters
<code>serverPort</code>	number	3112	The network port to use when communicating with the network client
<code>timeout</code>	number	10	Timeout in seconds before treating the network client as failed when waiting to send or receive content.

The network filter operates by sending field data, as defined in the corresponding filter configuration file, out to a network server that processes the information and sends it back to be re-introduced in place of the original field data. This can be used to translate and reformat information during the replication scheme.

The filter operation works as follows:

- All filtered data will be sent to a single network server, at the configured port.
- A single network server can be used to provide multiple transformations.
- The JSON configuration file for the filter supports multiple types and multiple column definitions.
- The protocol used by the network filter must be followed to effectively process the information. A failure in the network server or communication will cause the replicator to raise an error and replication to go [OFFLINE](#) [207].
- The network server must be running before the replicator is started. If the network server cannot be found, replication will go [OFFLINE](#) [207].

Correct operation requires building a suitable network filter, and creating the JSON configuration file.

11.4.26.1. Network Client Configuration

The format of the configuration file defines the translation operation to be requested from the network client, in addition to the schema, table and column name. The format for the file is JSON, with the top-level hash defining the operation, and an array of field selections for each field that should be processed accordingly. For example:

```
{
  "String_to_HEX_v1" : [
    {
      "table" : "hextable",
      "schema" : "hexdb",
      "columns" : [
        "hexcol"
      ]
    }
  ]
}
```

The operation in this case is `String_to_HEX_v1`; this will be sent to the network server as part of the request. The column definition follows.

To send multiple columns from different tables to the same translation:

```
{
  "String_to_HEX_v1" : [
    {
      "table" : "hextable",
      "schema" : "hexdb",
      "columns" : [
        "hexcol"
      ]
    },
    {
      "table" : "another_table",
      "schema" : "another_hexdb",
      "columns" : [
        "another_hexcol"
      ]
    }
  ]
}
```

```
{
    "table" : "hexagon",
    "schema" : "sourcetext",
    "columns" : [
        "itemtext"
    ]
}
]
```

Alternatively, to configure different operations for the same two tables:

```
{
    "String_to_HEX_v1" : [
        {
            "table" : "hextable",
            "schema" : "hexdb",
            "columns" : [
                "hexcol"
            ]
        }
    ],
    "HEX_to_String_v1" : [
        {
            "table" : "hexagon",
            "schema" : "sourcetext",
            "columns" : [
                "itemtext"
            ]
        }
    ]
}
```

11.4.26.2. Network Filter Protocol

The network filter protocol has been designed to be both lightweight and binary data compatible, as it is designed to work with data that may be heavily encoded, binary, or compressed in nature.

The protocol operates through a combined JSON and optional binary payload structure that communicates the information. The JSON defines the communication type and metadata, while the binary payload contains the raw or translated information.

The filter communicates with the network server using the following packet types:

- **prepare**

The `prepare` message is called when the filter goes online, and is designed to initialize the connection to the network server and confirm the supported filter types and operation. The format of the connection message is:

```
{
    "payload" : -1,
    "type" : "prepare",
    "service" : "firstrep",
    "protocol" : "v0_9"
}
```

Where:

- `protocol`

The protocol version.

- `service`

The name of the replicator service that called the filter.

- `type`

The message type.

- `payload`

The size of the payload; a value of -1 indicates that there is no payload.

The format of the response should be a JSON object and payload with the list of supported filter types in the payload section. The payload immediately follows the JSON, with the size of the list defined within the `payload` field of the returned JSON object:

```
{
```

```
    "payload" : 22,
    "type" : "acknowledged",
    "protocol" : "v0_9",
    "service" : "firstrep",
    "return" : 0
}Perl_BLOB_to_String_v1
```

Where:

- **protocol**

The protocol version.

- **service**

The name of the replicator service that called the filter.

- **type**

The message type; when acknowledging the original prepare request it should be [acknowledge](#).

- **return**

The return value. A value of 0 (zero) indicates no faults. Any true value indicates there was an issue.

- **payload**

The length of the appended payload information in bytes. This is used by the filter to identify how much additional data to read after the JSON object has been read.

The payload should be a comma-separated list of the supported transformation types within the network server.

- **filter**

The [filter](#) message type is sent by Tungsten Replicator for each value from the replication stream that needs to be filtered and translated in some way. The format of the request is a JSON object with a trailing block of data, the payload, that contains the information to be filtered. For example:

```
{
    "schema" : "hexdb",
    "transformation" : "String_to_HEX_v1",
    "service" : "firstrep",
    "type" : "filter",
    "payload" : 22,
    "row" : 0,
    "column" : "hexcol",
    "table" : "hextable",
    "seqno" : 145196,
    "fragments" : 1,
    "protocol" : "v0_9",
    "fragment" : 1
}48656c6c6f20576f726c64
```

Where:

- **protocol**

The protocol version.

- **service**

The service name the requested the filter.

- **type**

The message type, in this case, [filter](#).

- **row**

The row of the source information from the THL that is being filtered.

- **schema**

The schema of the source information from the THL that is being filtered.

- **table**

The table of the source information from the THL that is being filtered.

- [column](#)

The column of the source information from the THL that is being filtered.

- [seqno](#)

The sequence number of the event from the THL that is being filtered.

- [fragments](#)

The number of fragments in the THL that is being filtered.

- [fragment](#)

The fragment number within the THL that is being filtered. The fragments may be sent individually and sequentially to the network server, so they may need to be retrieved, merged, and reconstituted depending on the nature of the source data and the filter being applied.

- [transformation](#)

The transformation to be performed on the supplied payload data. A single network server can support multiple transformations, so this information is provided to perform the corrupt operation. The actual transformation to be performed is taken from the JSON configuration file for the filter.

- [payload](#)

The length, in bytes, of the payload data that will immediately follow the JSON filter request..

The payload that immediately follows the JSON block is the data from the column that should be processed by the network filter.

The response package should contain a copy of the supplied information from the requested filter, with the [payload](#) size updated to the size of the returned information, the message type changed to [filtered](#), and the payload containing the translated data. For example:

```
{  
    "transformation" : "String_to_HEX_v1",  
    "fragments" : 1,  
    "type" : "filtered",  
    "fragment" : 1,  
    "return" : 0,  
    "seqno" : 145198,  
    "table" : "hextable",  
    "service" : "firstrep",  
    "protocol" : "v0_9",  
    "schema" : "hexdb",  
    "payload" : 8,  
    "column" : "hexcol",  
    "row" : 0  
}FILTERED
```

11.4.26.3. Sample Network Client

The following sample network server script is written in Perl, and is designed to translate packed hex strings (two-hex characters per byte) from their hex representation into their character representation.

```
#!/usr/bin/perl  
  
use Switch;  
use IO::Socket::INET;  
use JSON qw( decode_json encode_json );  
use Data::Dumper;  
  
# auto-flush on socket  
$| = 1;  
  
my $serverName = "Perl_BLOB_to_String_v1";  
  
while(1)  
{  
    # creating a listening socket  
    my $socket = new IO::Socket::INET (  
        LocalHost => '0.0.0.0',  
        LocalPort => '3112',  
        Proto => 'tcp',  
        Listen => 5,
```

```

    Reuse => 1
};

die "Cannot create socket $!\n" unless $socket;
print "*****\nServer waiting for client connection on port 3112\n*****\n\n\n";

# Waiting for a new client connection
my $client_socket = $socket->accept();

# Get information about a newly connected client
my $client_address = $client_socket->peerhost();
my $client_port = $client_socket->peerport();
print "Connection from $client_address:$client_port\n";

my $data = "";

while(   $data = $client_socket->getline())
{
    # Read up to 1024 characters from the connected client
    chomp($data);

    print "\n\nReceived: <$data>\n";

    # Decode the JSON part
    my $msg = decode_json($data);

    # Extract payload
    my $payload = undef;

    if ($msg->{payload} > 0)
    {
        print STDERR "Reading $msg->{payload} bytes\n";
        $client_socket->read($payload,$msg->{payload});
        print "Payload: <$payload>\n";
    }

    switch( $msg->{'type'} )
    {
        case "prepare"
        {
            print STDERR "Received prepare request\n";

            # Send acknowledged message
            my $out = '{ "protocol": "v0_9", "type": "acknowledged", "return": 0, "service": "' . $msg->{'service'} . '", "payload": "' . length($serverName) . '" }' . "\n" . $serverName;

            print $client_socket "$out";
            print "Sent: <$out>\n";
            print STDERR "Sent acknowledge request\n";
        }
        case "release"
        {
            # Send acknowledged message
            my $out = '{ "protocol": "v0_9", "type": "acknowledged", "return": 0, "service": "' . $msg->{'service'} . '", "payload": "' . length($serverName) . '" }' . "\n" . $serverName;

            print $client_socket "$out\n";
            print "Sent: <$out>\n";
        }
        case "filter"
        {
            # Send filtered message

            print STDERR "Sending filtered payload\n";

            my $filtered = "FILTERED";
            my $out = <<END;
{
"protocol": "v0_9",
"type": "filtered",
"transformation": "$msg->{'transformation'}",
"return": 0,
"service": "$msg->{'service'}",
"seqno": $msg->{'seqno'},
"row": $msg->{'row'},
"schema": "$msg->{'schema'}",
"table": "$msg->{'table'}",
"column": "$msg->{'column'}",
"fragment": 1,
"fragments": 1,
"payload": @[$length($filtered)]}
}
END

$out =~ s/\n//g;
print "About to send: <$out>\n";

```

```

        $client_socket->send("$out\n" . $filtered);
print( "Response sent\n");
    }
}

print("End of loop, hoping for next packet\n");
}

# Notify client that we're done writing
shutdown($client_socket, 1);

$socket->close();
}

```

11.4.27. `nocreatedbifnotexists.js` Filter

The `nocreatedbifnotexists` filter removes statements that start with:

`CREATE DATABASE IF NOT EXISTS`

Pre-configured filter name	<code>nocreatedbifnotexists</code>		
JavaScript Filter File	<code>tungsten-replicator/support/filters-javascript/nocreatedbifnotexists.js</code>		
Property prefix	<code>replicator.filter.nocreatedbifnotexists</code>		
Stage compatibility	<code>q-to-dbms</code>		
<code>tpm</code> Option compatibility	<code>--svc-applier-filters</code> [366]		
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description

This can be useful in heterogeneous replication where Tungsten Replicator specific databases need to be removed from the replication stream.

The filter works in two phases. The first phase creates a global variable within the `prepare()` [407] function that defines the string to be examined:

```

function prepare()
{
    beginning = "CREATE DATABASE IF NOT EXISTS";
}

```

Row based changes can be ignored, but for statement based events, the SQL is examined and the statement removed if the SQL starts with the text in the `beginning` variable:

```

sql = d.getQuery();
if(sql.startsWith(beginning))
{
    data.remove(i);
    i--;
}

```

11.4.28. OptimizeUpdates Filter

The `optimizeupdates` filter works with row-based events to simplify the update statement and remove columns/values that have not changed. This reduces the workload and row data exchanged between replicators.

Pre-configured filter name	<code>optimizeupdates</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.OptimizeUpdatesFilter</code>		
Property prefix	<code>replicator.filter.optimizeupdates</code>		
Stage compatibility			
<code>tpm</code> Option compatibility			
Data compatibility	Row events		
Parameters			
(none)			

The filter operates by removing column values for keys in the update statement that do not change. For example, when replicating the row event from the statement:

```
mysql> update testopt set msg = 'String1', string = 'String3' where id = 1;
```

Generates the following THL event data:

```
- SQL(0) =
- ACTION = UPDATE
- SCHEMA = test
- TABLE = testopt
- ROW# = 0
- COL(1: id) = 1
- COL(2: msg) = String1
- COL(3: string) = String3
- KEY(1: id) = 1
```

Column 1 (`id`) in this case is automatically implied by the KEY entry required for the update.

With the `optimizeupdates` filter enabled, the data in the THL is simplified to:

```
- SQL(0) =
- ACTION = UPDATE
- SCHEMA = test
- TABLE = testopt
- ROW# = 0
- COL(2: msg) = String1
- COL(3: string) = String4
- KEY(1: id) = 1
```

In tables where there are multiple keys the stored THL information can be reduced further.

Warning

The filter works by comparing the value of each KEY and COL entry in the THL and determining whether the value has changed or not. If the number of keys and columns do not match then the filter will fail with the following error message:

```
Caused by: java.lang.Exception: Column and key count is different in this event! Cannot filter
```

This may be due to a filter earlier within the filter configuration that has optimized or simplified the data. For example, the `pkey` filter removes KEY entries from the THL that are not primary keys, or `dropcolumn` which drops column data.

The following error message may appear in the logs and in the output from `trepctl status` to indicate that this ordering issue may be the problem:

```
OptimizeUpdatesFilter cannot filter, because column and key count is different.
Make sure that it is defined before filters which remove keys (eg. PrimaryKeyFilter).
```

11.4.29. PrimaryKey Filter

The `PrimaryKey` adds primary key information to row-based replication data. This is required by heterogeneous environments to ensure that the primary key is identified when updating or deleting tables. Without this information, the primary to use, for example as the document ID in a document store such as MongoDB, is generated dynamically. In addition, without this filter in place, when performing update or delete operations a full table scan is performed on the target dataserver to determine the record that must be updated.

Pre-configured filter name	<code>pkey</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.PrimaryKeyFilter</code>		
Property prefix	<code>replicator.filter.pkey</code>		
Stage compatibility	<code>binlog-to-q</code>		
tpm Option compatibility	<code>--repl-svc-extractor-filters [366]</code>		
Data compatibility	Row events		
Parameters			
Parameter	Type	Default	Description
<code>user</code>	string	<code> \${replicator.global.extract.db.user}</code>	The username for the connection to the database for looking up column definitions
<code>password</code>	string	<code> \${replicator.global.extract.db.password}</code>	The password for the connection to the database for looking up column definitions
<code>url</code>	string	<code> jdbc:mysql:thin://\${replicator.global.extract.db.host}:\${replica-</code>	JDBC URL of the database connection to use for looking up column definitions

		<code>tor.global.extract.db.port}/\${replicator.schema}?createDB=true</code>	
<code>addPkeyToInsert</code>	<code>boolean</code>	<code>false</code>	If set to true, primary keys are added to <code>INSERT</code> operations. This setting is required for batch loading
<code>addColumnstoDeletes</code>	<code>boolean</code>	<code>false</code>	If set to true, full column metadata is added to <code>DELETE</code> operations. This setting is required for batch loading

Note

This filter is designed to be used for testing and with heterogeneous replication where the field name information can be used to construct and build target data structures.

For example, in the following THL fragment, the key information includes data for all columns, which is the default behavior for `UPDATE` and `DELETE` operations.

```
SEQ# = 142 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:31:04.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:000000000022187:0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
- ACTION = UPDATE
- SCHEMA = test
- TABLE = salesadv
- ROW# = 0
- COL(1: id) = 2
- COL(2: country) = 1
- COL(3: city) = 8374
- COL(4: salesman) = 1
- COL(5: value) = 89000.00
- KEY(1: id) = 2
- KEY(2: country) = 1
- KEY(3: city) = 8374
- KEY(4: salesman) = 1
- KEY(5: value) = 89000.00
```

When the `PrimaryKey` is enabled, the key information has been optimized to only contain the actual primary keys are added to the row-based THL record:

```
SEQ# = 142 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:31:04.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:000000000022187:0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
- ACTION = UPDATE
- SCHEMA = test
- TABLE = salesadv
- ROW# = 0
- COL(1: id) = 2
- COL(2: country) = 1
- COL(3: city) = 8374
- COL(4: salesman) = 1
- COL(5: value) = 89000.00
- KEY(1: id) = 2
```

The final line shows the addition of the primary key `id` added to THL event.

The two options, `addPkeyToInsert` and `addColumnstoDeletes` add the primary key information to `INSERT` and `DELETE` operations respectively. In a heterogeneous environment, these options should be enabled to prevent full-table scans during update and deletes.

11.4.30. PrintEvent Filter

Pre-configured filter name	<code>printevent</code>
Classname	<code>com.continuent.tungsten.replicator.filter.PrintEventFilter</code>
Property prefix	<code>replicator.filter.printevent</code>
Stage compatibility	

tpm Option compatibility	
Data compatibility	Any event
Parameters	
(none)	

11.4.31. Rename Filter

The `rename` filter enables schemas to be renamed at the database, table and column levels, and for complex combinations of these renaming operations. Configuration is through a CSV file that defines the rename parameters. A single CSV file can contain multiple rename definitions. The rename operations occur only on ROW based events.

Schema Renaming in 2.2.1 and later. The `rename` filter also performs schema renames on statement data only in Tungsten Replicator 2.2.1.

Pre-configured filter name	<code>rename</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.RenameFilter</code>		
Property prefix	<code>replicator.filter.rename</code>		
Stage compatibility			
tpm Option compatibility			
Data compatibility	Row events; Schema names of Statement events in 2.2.1 and later.		
Parameters			
Parameter	Type	Default	Description
<code>definitionsFile</code>	string	<code>{replicator.home.dir}/samples/extensions/java/rename.csv</code>	Location of the CSV file that contains the rename definitions.

The CSV file is only read when an explicit reconfigure operation is triggered. If the file is changed, a configure operation (using `tpm update`) must be initiated to force reconfiguration.

To enable using the default CSV file:

```
shell> ./tools/tpm update alpha --svc-applier-filters=rename
```

The CSV consists of multiple lines, one line for each rename specification. Comments are supposed using the `#` character.

The format of each line of the CSV is:

```
originalSchema,originalTable,originalColumn,newSchema,newTable,newColumn
```

Where:

- `originalSchema, originalTable, originalColumn` define the original schema, table and column.

Definition can either be:

- Explicit schema, table or column name
- `*` character, which indicates that all entries should match.
- `newSchema, newTable, newColumn` define the new schema, table and column for the corresponding original specification.

Definition can either be:

- Explicit schema, table or column name
- `-` character, which indicates that the corresponding object should not be updated.

For example, the specification:

```
*.chicago,*,-,newyork,-
```

Would rename the table `chicago` in every database schema to `newyork`. The schema and column names are not modified.

The specification:

```
*.chicago,destination,-,-,source
```

Would match all schemas, but update the column `destination` in the table `chicago` to the column name `source`, without changing the schema or table name.

Processing of the individual rules is executed in a specific order to allow for complex matching and application of the rename changes.

- Rules are case sensitive.
- Schema names are looked up in the following order:
 1. `schema.table` (explicit schema/table)
 2. `schema.*` (explicit schema, wildcard table)
- Table names are looked up in the following order:
 1. `schema.table` (explicit schema/table)
 2. `*.table` (wildcard schema, explicit table)
- Column names are looked up in the following order:
 1. `schema.table` (explicit schema/table)
 2. `schema.*` (explicit schema, wildcard table)
 3. `*.table` (wildcard schema, explicit table)
 4. `*.*` (wildcard schema, wildcard table)
- Rename operations match the first specification according to the above rules, and only one matching rule is executed.

11.4.31.1. Rename Filter Examples

When processing multiple entries that would match the same definition, the above ordering rules are applied. For example, the definition:

```
asia,*,*,america,-,-  
asia,shanghai,*,europe,-,-
```

Would rename `asia.shanghai` to `europe.shanghai`, while renaming all other tables in the schema `asia` to the schema `america`. This is because the explicit `schema.table` rule is matched first and then executed.

Complex renames involving multiple schemas, tables and columns can be achieved by writing multiple rules into the same CSV file. For example given a schema where all the tables currently reside in a single schema, but must be renamed to specific continents, or to a 'miscellaneous' schema, while also updating the column names to be more neutral would require a detailed rename definition.

Existing tables are in the schema `sales`:

```
chicago  
newyork  
london  
paris  
munich  
moscow  
tokyo  
shanghai  
sydney
```

Need to be renamed to:

```
northamerica.chicago  
northamerica.newyork  
europe.london  
europe.paris  
europe.munich  
misc.moscow  
asiapac.tokyo  
asiapac.shanghai  
misc.sydney
```

Meanwhile, the table definition needs to be updated to support more complex structure:

```
id  
area  
country  
city  
value
```

type

The area is being updated to contain the region within the country, while the value should be renamed to the three-letter currency code, for example, the `london` table would rename the `value` column to `gbp`.

The definition can be divided up into simple definitions at each object level, relying on the processing order to handle the individual exceptions. Starting with the table renames for the continents:

```
sales,chicago,*,northamerica,--,-
sales,newyork,*,northamerica,--,-
sales,london,*,europe,--,-
sales,paris,*,europe,--,-
sales,munich,*,europe,--,-
sales,tokyo,*,asiapac,--,-
sales,shanghai,*,asiapac,--,-
```

A single rule to handle the renaming of any table not explicitly mentioned in the list above into the `misc` schema:

```
*,*,*,misc,--,
```

Now a rule to change the `area` column for all tables to `region`. This requires a wildcard match against the schema and table names:

```
*,*,area,--,region
```

And finally the explicit changes for the value column to the corresponding currency:

```
*,chicago,value,--,usd
*,newyork,value,--,usd
*,london,value,--,gbp
*,paris,value,--,eur
*,munich,value,--,eur
*,moscow,value,--,rub
*,tokyo,value,--,jpy
*,shanghai,value,--,cny
*,sydney,value,--,aud
```

11.4.32. ReplicateColumns Filter

Pre-configured filter name	<code>replicatemcolumns</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.ReplicateColumnsFilter</code>		
Property prefix	<code>replicator.filter.replicatemcolumns</code>		
Stage compatibility			
tpm Option compatibility			
Data compatibility	Row events		
Parameters			
Parameter	Type	Default	Description
<code>ignore</code>	string	empty	Comma separated list of tables and optional column names to ignore during replication
<code>do</code>	string	empty	Comma separated list of tables and optional column names to replicate

11.4.33. Replicate Filter

The `replicate` filter enables explicit inclusion or exclusion of tables and schemas. Each specification supports wildcards and multiple entries.

Pre-configured filter name	<code>replicate</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.ReplicateFilter</code>		
Property prefix	<code>replicator.filter.replicate</code>		
Stage compatibility	Any		
tpm Option compatibility			
Data compatibility	Any event		
Parameters			

Parameter	Type	Default	Description
<code>ignore</code>	string	empty	Comma separated list of database/tables to ignore during replication
<code>do</code>	string	empty	Comma separated list of database/tables to replicate

Rules using the supplied parameters are evaluated as follows:

- When both `do` and `ignore` are empty, updates are allowed to any table.
- When only `do` is specified, only the schemas (or schemas and tables) mentioned in the list are replicated.
- When only `ignore` is specified, all schemas/tables are replicated except those defined.

For each parameter, a comma-separated list of schema or schema and table definitions are supported, and wildcards using `*` (any number of characters) and `?` (single character) are also honored. For example:

- `do=sales`

Replicates only the schema `sales`.

- `ignore=sales`

Replicates everything, ignoring the schema `sales`.

- `ignore=sales.*`

Replicates everything, ignoring the schema `sales`.

- `ignore=sales.quarter?`

Replicates everything, ignoring all tables within the `sales` schema starting with `sales.quarter` and a single character. This would ignore `sales.quarter1` but replicate `sales.quarterlytotals`.

- `ignore=sales.quarter*`

Replicates everything, ignoring all tables in the schema `sales` starting with `quarter`.

- `do=*.quarter`

Replicates only the table named `quarter` within any schema.

- `do=sales.*totals,invoices`

Replicates only tables in the `sales` schema that end with `totals`, and the entire `invoices` schema.

11.4.34. SetToString Filter

The `SetToString` converts the `SET` column type from the internal representation to a string-based representation in the THL. This achieved by accessing the extractor database, obtaining the table definitions, and modifying the THL data before it is written into the THL file.

Pre-configured filter name	<code>settoString</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.SetToStringFilter</code>		
Property prefix	<code>replicator.filter.settoString</code>		
Stage compatibility	<code>binlog-to-q</code>		
<code>tpm</code> Option compatibility	<code>--repl-svc-extractor-filters [366]</code>		
Data compatibility	Row events		
Parameters			
Parameter	Type	Default	Description
<code>user</code>	string	<code> \${replicator.global.extract.db.user} </code>	The username for the connection to the database for looking up column definitions
<code>password</code>	string	<code> \${replicator.global.extract.db.password} </code>	The password for the connection to the database for looking up column definitions
<code>url</code>	string	<code> jdbc:mysql:thin://\${replicator.global.extract.db.host}:\${replica-} </code>	JDBC URL of the database connection to use for looking up column definitions

	<code>tor.global.extract.db.port} / \${replicator.schema} ?createDB=true</code>
--	---

The `SetToString` filter should be used with heterogeneous replication to ensure that the data is represented as the string value, not the internal numerical representation.

In the THL output below, the table has a `SET` column, `salesman`:

```
mysql> describe salesadv;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id   | int(11) | NO  | PRI | NULL    | auto_increment |
| country | enum('US','UK','France','Australia') | YES |     | NULL    |
| city  | int(11) | YES |     | NULL    |
| salesman | set('Alan','Zachary') | YES |     | NULL    |
| value  | decimal(10,2) | YES |     | NULL    |
+-----+-----+-----+-----+-----+
```

When extracted in the THL, the representation uses the internal value (for example, 1 for the first element of the set description). This can be seen in the THL output below.

```
SEQ# = 138 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:09:35.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:0000000000021434:0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
  - ACTION = INSERT
  - SCHEMA = test
  - TABLE = salesadv
  - ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 1
  - COL(3: city) = 8374
  - COL(4: salesman) = 1
  - COL(5: value) = 35000.00
```

For the `salesman` column, the corresponding value in the THL is 1. With the `SetToString` filter enabled, the value is expanded to the corresponding string value:

```
SEQ# = 121 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:05:14.0
- EPOCH# = 102
- EVENTID = mysql-bin.000012:0000000000018866:0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
  - ACTION = INSERT
  - SCHEMA = test
  - TABLE = salesadv
  - ROW# = 0
  - COL(1: id) = 1
  - COL(2: country) = US
  - COL(3: city) = 8374
  - COL(4: salesman) = Alan
  - COL(5: value) = 35000.00
```

The examples here also show the [Section 11.4.19, “EnumToString Filter”](#) and [Section 11.4.8, “ColumnName Filter”](#) filters.

11.4.35. Shard Filter

Pre-configured filter name	<code>shardfilter</code>
Classname	<code>com.continuent.tungsten.replicator.filter.ShardFilter</code>
Property prefix	<code>replicator.filter.shardfilter</code>
Stage compatibility	
<code>tpm</code> Option compatibility	
Data compatibility	Any event

Parameters			
Parameter	Type	Default	Description
<code>enabled</code>	<code>boolean</code>	<code>false</code>	If set to true, enables the shard filter
<code>unknownShardPolicy</code>	<code>string</code>	<code>error</code>	Select the filter policy when the shard unknown; valid values are <code>accept</code> , <code>drop</code> , <code>warn</code> , and <code>error</code>
<code>unwantedShardPolicy</code>	<code>string</code>	<code>error</code>	Select the filter policy when the shard is unwanted; valid values are <code>accept</code> , <code>drop</code> , <code>warn</code> , and <code>error</code>
<code>enforcedHome</code>	<code>boolean</code>	<code>false</code>	If true, enforce the home for the shard
<code>allowWhitelisted</code>	<code>boolean</code>	<code>false</code>	If true, allow explicitly whitelisted shards
<code>autoCreate</code>	<code>boolean</code>	<code>false</code>	If true, allow shard rules to be created automatically

11.4.36. `shardbyseqno.js` Filter

Shards within the replicator enable data to be parallelized when they are applied on the slave.

Pre-configured filter name	<code>shardbyseqno</code>		
JavaScript Filter File	<code>tungsten-replicator/support/filters-javascript/shardbyseqno.js</code>		
Property prefix	<code>replicator.filter.shardbyseqno</code>		
Stage compatibility	<code>q-to-dbms</code>		
<code>tpm</code> Option compatibility	<code>--svc-applier-filters [366]</code>		
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description
<code>shards</code>	<code>numeric</code>	(none)	Number of shards to be used by the applier

The `shardbyseqno` filter updates the shard ID, which is embedded into the event metadata, by a configurable number of shards, set by the `shards` parameter in the configuration:

```
replicator.filter.shardbyseqno=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.shardbyseqno.script=${replicator.home}/samples/extensions/javascript/shardbyseqno.js
replicator.filter.shardbyseqno.shards=10
```

The filter works by setting the shard ID in the event using the `setShardId()` method on the event object:

```
event.setShardId(event.getSegno() % 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.

11.4.37. `shardbytable.js` Filter

An alternative to `sharding by sequence number` is to create a shard ID based on the individual database and table. The `shardbytable` filter achieves this at a row level by combining the schema and table information to form the shard ID. For all other events, including statement based events, the shard ID `#UNKNOWN` is used.

Pre-configured filter name	<code>shardbytable</code>		
JavaScript Filter File	<code>tungsten-replicator/support/filters-javascript/shardbytable.js</code>		
Property prefix	<code>replicator.filter.shardbytable</code>		
Stage compatibility	<code>q-to-dbms</code>		
<code>tpm</code> Option compatibility	<code>--svc-applier-filters [366]</code>		
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description

The key part of the filter is the extraction and construction of the ID, which occurs during row processing:

```
oneRowChange = rowChanges.get(j);
schemaName = oneRowChange.getSchemaName();
tableName = oneRowChange.getTableName();

id = schemaName + "_" + tableName;
if (proposedShardId == null)
{
    proposedShardId = id;
}
```

11.4.38. 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.

Pre-configured filter name	<code>delay</code>		
Classname	<code>com.continuent.tungsten.replicator.filter.TimeDelayFilter</code>		
Property prefix	<code>replicator.filter.delay</code>		
Stage compatibility	<code>remote-to-thl</code>		
<code>tpm</code> Option compatibility	<code>--repl-svc-thl-filters [367]</code>		
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description
<code>delay</code>	numeric	300	Number of seconds to delay transaction processing row

The `TimeDelay` delays the application of transactions recorded in the THL. The delay can be used to allow point-in-time recovery of DML operations before the transaction has been applied to the slave, or where data may need to be audited or checked before transactions are committed.

Note

For effective operation, master and slaves should be synchronized using NTP or a similar protocol.

To enable the `TimeDelayFilter`, use `tpm` command to enable the filter operation and the required delay. For example, to enable the delay for 900 seconds:

```
shell> ./tools/tpm update alpha --hosts=host1,host2,host3 \
--repl-svc-applier-filters=delay \
--property=replicator.filter.delay.delay=900
```

Time delay of transaction events should be performed with care, since the delay will prevent a slave from being up to date compared to the master. In the event of a node failure, an up to date slave is required to ensure that data is safe.

11.4.39. `tosingledb.js` Filter

This filter updates the replicated information so that it goes to an explicit schema, as defined by the user. The filter can be used to combine multiple tables to a single schema.

Pre-configured filter name	<code>tosingledb</code>		
JavaScript Filter File	<code>tungsten-replicator/support/filters-javascript/tosingledb.js</code>		
Property prefix	<code>replicator.filter.ansiquotes</code>		
Stage compatibility	<code>q-to-dbms</code>		
<code>tpm</code> Option compatibility	<code>--svc-applier-filters [366]</code>		
Data compatibility	Any event		
Parameters			
Parameter	Type	Default	Description
<code>db</code>	<code>string</code>	(none)	Database name into which to replicate all tables
<code>skip</code>	<code>string</code>	(none)	Comma-separated list of databases to be ignored

A database can be optionally ignored through the `skip` parameter within the configuration:

```
replicator.filter.tosingledb=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.tosingledb.script=${replicator.home.dir}/samples/extensions/javascript/tosingledb.js
replicator.filter.tosingledb.db=dbtoreplicateto
replicator.filter.tosingledb.skip=tungsten
```

Similar to other filters, the filter operates by explicitly changing the schema name to the configured schema, unless the skipped schema is in the event data. For example, at a statement level:

```
if(oldDb!=null && oldDb.compareTo(skip)!=0)
{
    d.setDefaultSchema(db);
}
```

11.4.40. `truncatetext.js` Filter

The `truncatetext` filter truncates a MySQL `BLOB` field.

Pre-configured filter name	<code>truncatetext</code>		
JavaScript Filter File	<code>tungsten-replicator/support/filters-javascript/truncatetext.js</code>		
Property prefix	<code>replicator.filter.truncatetext</code>		
Stage compatibility	<code>binlog-to-q, q-to-dbms</code>		
<code>tpm</code> Option compatibility	<code>--svc-extractor-filters [366], --svc-extractor-filters [366]</code>		
Data compatibility	Row events		
Parameters			
Parameter	Type	Default	Description
<code>length</code>	<code>numeric</code>	(none)	Maximum size of truncated field (bytes)

The length is determined by the `length` parameter in the properties:

```
replicator.filter.truncatetext=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.truncatetext.script=${replicator.home.dir}/samples/extensions/javascript/truncatetext.js
replicator.filter.truncatetext.length=4000
```

Statement-based events are ignored, but row-based events are processed for each volume value, checking the column type, `isBlob()` method and then truncating the contents when they are identified as larger than the configured length. To confirm the type, it is compared against the Java class `com.continuent.tungsten.replicator.extractor.mysql.SerialBlob`, the class for a serialized `BLOB` value. These need to be processed differently as they are not exposed as a single variable.

```
if (value.getValue() instanceof com.continuent.tungsten.replicator.extractor.mysql.SerialBlob)
{
    blob = value.getValue();
    if (blob != null)
    {
        valueBytes = blob.getBytes(1, blob.length());
        if (blob.length() > truncateTo)
        {
            blob.truncate(truncateTo);
        }
    }
}
```

11.4.41. `zerodate2null.js` Filter

The `zerodate2null` filter looks complicated, but is very simple. It processes row data looking for date columns. If the corresponding value is zero within the column, the value is updated to NULL. This is required for MySQL to Oracle replication scenarios.

Pre-configured filter name	<code>zerodate2null</code>		
JavaScript Filter File	<code>tungsten-replicator/support/filters-javascript/zerodate2null.js</code>		
Property prefix	<code>replicator.filter.zerodate2null</code>		
Stage compatibility	<code>q-to-dbms</code>		
<code>tpm</code> Option compatibility	<code>--svc-applier-filters [366]</code>		
Data compatibility	Row events		
Parameters			

Parameter	Type	Default	Description
-----------	------	---------	-------------

The filter works by examining the column specification using the `getColumnSpec()` method. Each column is then checked to see if the column type is a `DATE`, `DATETIME` or `TIMESTAMP` by looking the type ID using some stored values for the date type.

Because the column index and corresponding value index match, when the value is zero, the column value is explicitly set to NULL using the `setValueNull()` method.

```
for(j = 0; j < rowChanges.size(); j++)
{
    oneRowChange = rowChanges.get(j);
    columns = oneRowChange.getColumnSpec();
    columnValues = oneRowChange.getColumnValues();
    for (c = 0; c < columns.size(); c++)
    {
        columnSpec = columns.get(c);
        type = columnSpec.getType();
        if (type == TypesDATE || type == TypesTIMESTAMP)
        {
            for (row = 0; row < columnValues.size(); row++)
            {
                values = columnValues.get(row);
                value = values.get(c);

                if (value.getValue() == 0)
                {
                    value.setValueNull();
                }
            }
        }
    }
}
```

11.5. JavaScript Filters

In addition to the supplied Java filters, Tungsten Replicator also includes support for custom script-based filters written in JavaScript and supported through the JavaScript filter. This filter provides a JavaScript environment that exposes the transaction information as it is processed internally through an object-based JavaScript API.

The JavaScript implementation is provided through the Rhino open-source implementation. Rhino provides a direct interface between the underlying Java classes used to implement the replicator code and a full JavaScript environment. This enables scripts to be developed that have access to the replicator constructs and data structures, and allows information to be updated, reformatted, combined, extracted and reconstructed.

At the simplest level, this allows for operations such as database renames and filtering. More complex solutions allow for modification of the individual data, such as removing nulls, bad dates, and duplication of information.

Warning

Updating the static properties file for the replicator will break automated upgrades through `tpm`. When upgrading, `tpm` relies on existing template files to create the new configuration based on the `tpm` parameters used.

Making a backup copy of the configuration file automatically generated by `tpm`, and then using this before performing an `upgrade` will enable you to update your configuration automatically. Settings for the JavaScript filter will then need to be updated in the configuration file manually.

To enable a JavaScript filter that has not already been configured, the static properties file (`static-SERVICE.properties`) must be edited to include the definition of the filter using the `JavaScriptFilter` class, using the `script` property to define the location of the actual JavaScript file containing the filter definition. For example, the supplied `ansiQuotes` filter is defined as follows:

```
replicator.filter.ansiQuotes=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.ansiQuotes.script=${replicator.home.dir}/support/filters-javascript/ansiQuotes.js
```

To use the filter, add the filter name, `ansiQuotes` in the above example, to the required stage:

```
replicator.stage.q-to-dbms.filters=mysqlSessions,pkey,bidiSlave,ansiQuotes
```

Then restart the replicator to enable the configuration:

```
shell> replicator restart
```

Note

This procedure will need to be enabled on each replicator that you want to use the JavaScript filter.

If there is a problem with the JavaScript filter during restart, the replicator will be placed into the [OFFLINE](#) [207] state and the reason for the error will be provided within the replicator [trepsvc.log](#) log.

11.5.1. Writing JavaScript Filters

The JavaScript interface to the replicator enables filters to be written using standard JavaScript with a complete object-based interface to the internal Java objects and classes that make up the THL data.

For more information on the Rhino JavaScript implementation, see [Rhino](#).

The basic structure of a JavaScript filter is as follows:

```
// Prepare the filter and setup structures
prepare()
{
}

// Perform the filter process; function is called for each event in the THL
filter(event)
{
    // Get the array of DBMSData objects
    data = event.getData();

    // Iterate over the individual DBMSData objects
    for(i=0;i<data.size();i++)
    {
        // Get a single DBMSData object
        d = data.get(i);

        // Process a Statement Event; event type is identified by comparing the object class type
        if (d = instanceof com.continuent.tungsten.replicator.dbms.StatementData)
        {
            // Do statement processing
        }
        else if (d = instanceof com.continuent.tungsten.replicator.dbms.RowChangeData)
        {
            // Get an array of all the row changes
            rows = data.get(i).getRowChanges();

            // Iterate over row changes
            for(j=0;j<rows.size();j++)
            {
                // Get the single row change
                rowchange = rows.get(j);

                // Identify the row change type
                if (rowchange.getAction() == "INSERT")
                {
                }
                ....
            }
        }
    }
}
```

The following sections will examine the different data structures, functions, and information available when processing these individual events.

11.5.1.1. Implementable Functions

Each JavaScript filter must defined one or more functions that are used to operate the filter process. The [filter\(\)](#) [408] 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\(\)](#) [407] 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()` [408] 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() [408]`

The `release()` [408] function is called when the filter is deallocated and removed, typically during shutdown of the replicator, although it may also occur when a processing thread is restarted.

11.5.1.2. Getting Configuration Parameters

The JavaScript interface enables you to get two different sets of configuration properties, the filter specific properties, and the general replicator properties. The filter specific properties should be used to configure and specify configuration information unique to that instance of the filter configuration. Since multiple filter configurations using the same filter definition can be created, using the filter-specific content is the simplest method for obtaining this information.

- **Getting Filter Properties**

To obtain the properties configured for the filter within the static configuration file according to the context of the filter configuration, use the `filterProperties` class with the `getString()` method. For example, the `jsdbrename` filter uses two properties, `dbsource` and `dbtarget` to identify the database to be renamed and the new name. The definition for the filter within the configuration file might be:

```
replicator.filter.jsdbrename=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.jsdbrename.script=${replicator.home.dir}/support/filters-javascript/dbrename.js
replicator.filter.jsdbrename.dbsource=contacts
replicator.filter.jsdbrename.dbtarget=nyc_contacts
```

Within the JavaScript filter, they are retrieved using:

```
sourceName = filterProperties.getString("dbsource");
targetName = filterProperties.getString("dbtarget");
```

- **Generic Replicator Properties**

General properties can be retrieved using the `properties` class and the `getString()` method:

```
master = properties.getString("replicator.thl.remote_uri");
```

11.5.1.3. Logging Information and Exceptions

Information about the filtering process can be reported into the standard `trepsvc.log` file by using the `logger` object. This supports different methods according to the configured logging level:

- `logger.info()` — information level entry, used to indicate configuration, loading or progress.
- `logger.debug()` — information will be logged when debugging is enabled, used when showing progress during development.
- `logger.error()` — used to log an error that would cause a problem or replication to stop.

For example, to log an informational entry that includes data from the filter process:

```
logger.info("regexp: Translating string " + valueString.valueOf());
```

To raise an exception that causes replication to stop, a new `ReplicatorException` object must be created that contains the error message:

```
if(col == null)
{
    throw new com.continuent.tungsten.replicator.ReplicatorException(
        "dropcolumn.js: column name in " + schema + "." + table +
        " is undefined - is colnames filter enabled and is it before the dropcolumn filter?"
    );
}
```

The error string provided will be used as the error provided through `trepctl`, in addition to raising and exception and backtrace within the log.

11.5.1.4. Exposed Data Structures

Within the `filter()` [408] function that must be defined within the JavaScript filter, a single `event` object is supplied as the only argument. That event object contains all of the information about a single event as recorded within the THL as part of the replication process.

Each event contains metadata information that can be used to identify or control the content, and individual statement and row data that contain the database changes.

The content of the information is a compound set of data that contains one or more further blocks of data changes, which in turn contains one or more blocks of SQL statements or row data. These blocks are defined using the Java objects that describe their internal format, and are exposed within the JavaScript wrapper as JavaScript objects, that can be parsed and manipulated.

At the top level, the Java object provided to the `filter()` [408] function as the `event` argument is `ReplDBMSEvent`. The `ReplDBMSEvent` class provides the core event information with additional management metadata such as the global transaction ID (`seqno`), latency of the event and sharding information.

That object contains one or more `DBMSData` objects. Each `DBMSData` object contains either a `StatementData` object (in the case of a statement based event), or a `RowChangeData` object (in the case of row-based events). For row-based events, there will be one or more `OneRowChange` [411] 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:

<code>ReplDBMSEvent</code>	<code>DBMSData</code>	<code>StatementData</code>	
	<code>DBMSData</code>	<code>StatementData</code>	
	<code>DBMSData</code>	<code>RowChangeData</code>	<code>OneRowChange</code> [411]
			<code>OneRowChange</code> [411]
			...
		<code>StatementData</code>	
<code>ReplDBMSEvent</code>	<code>DBMSData</code>	<code>RowChangeData</code>	<code>OneRowChange</code> [411]
			<code>OneRowChange</code> [411]
			...

A single event can contain both statement and row change information within the list of individual `DBMSData` events. An event or

11.5.1.4.1. `ReplDBMSEvent` Objects

The base object from which all of the data about replication can be obtained is the `ReplDBMSEvent` class. The class contains all of the information about each event, including the global transaction ID and statement or row data.

The interface to the underlying information is through a series of methods that provide the embedded information or data structures, described in the table below.

Method	Description
<code>getAppliedLatency()</code>	Returns the latency of the embedded event. See Section E.2.6, "Terminology: Fields <code>appliedLatency</code>"
<code>getData()</code>	Returns an array of the <code>DBMSData</code> objects within the event
<code>getDBMSEvent()</code>	Returns the original <code>DBMSEvent</code> object
<code>getEpochNumber()</code>	Get the Epoch number of the stored event. See THL EPOCH# [460]
<code>getEventId()</code>	Returns the native event ID. See THL EVENTID [460]
<code>getExtractedTstamp()</code>	Returns the timestamp of the event.
<code>getFragno()</code>	Returns the fragment ID. See THL SEQNO [459]
<code>getLastFrag()</code>	Returns true if the fragment is the last fragment in the event.
<code>getSeqno()</code>	Returns the native sequence number. See THL SEQNO [459]
<code>getShardId()</code>	Returns the shard ID for the event.
<code>getSourceId()</code>	Returns the source ID of the event. See THL SOURCEID [460]
<code>setShardId()</code>	Sets the shard ID for the event, which can be used by the filter to set the shard.

The primary method used is `getData()`, which returns an array of the individual `DBMSData` objects contained in the event:

```
function filter(event)
{
```

```
    data = event.getData();
    if(data != null)
    {
        for (i = 0; i < data.size(); i++)
        {
            change = data.get(i);
        ...
    }
```

Access to the underlying array structure uses the `get()` method to request individual objects from the array. The `size()` method returns the length of the array.

Removing or Adding Data Changes

Individual `DBMSData` objects can be removed from the replication stream by using the `remove()` method, supplying the index of the object to remove:

```
data.remove(1);
```

The `add()` method can be used to add new data changes into the stream. For example, data can be duplicated across tables by creating and adding a new version of the event, for example:

```
if(d.getDefaultSchema() != null &&
   d.getDefaultSchema().compareTo(sourceName)==0)
{
    newStatement = new
        com.continuent.tungsten.replicator.dbms.StatementData(d.getQuery(),
                                                               null,
                                                               targetName);
    data.add(data.size(),newStatement);
}
```

The above code looks for statements within the `sourceName` schema and creates a copy of each statement into the `targetName` schema.

The first argument to `add()` is the index position to add the statement. Zero (0) indicates before any existing changes, while using `size()` on the array effectively adds the new statement change at the end of the array.

Updating the Shard ID

The `setShardId()` method can also be used to set the shard ID within an event. This can be used in filters where the shard ID is updated by examining the schema or table being updated within the embedded SQL or row data. An example of this is provided in [Section 11.4.37, "shardbytable.js Filter"](#).

11.5.1.4.2. `DBMSData` Objects

The `DBMSData` object provides encapsulation of either the SQL or row change data within the THL. The class provides no methods for interacting with the content, instead, the real object should be identified and processed accordingly. Using the JavaScript `instanceof` operator the underlying type can be determined:

```
if (d != null &&
    d instanceof com.continuent.tungsten.replicator.dbms.StatementData)
{
    // Process Statement data
}
else if (d != null &&
         d instanceof com.continuent.tungsten.replicator.dbms.RowChangeData)
{
    // Process Row data
}
```

Note the use of the full object class for the different `DBMSData` types.

For information on processing `StatementData`, see [Section 11.5.1.4.3, "StatementData Objects"](#). For row data, see [Section 11.5.1.4.4, "RowChangeData Objects"](#).

11.5.1.4.3. `StatementData` Objects

The `StatementData` class contains information about data that has been replicated as an SQL statement, as opposed to information that is replicated as row-based data.

Processing and filtering statement information relies on editing the original SQL query statement, or the metadata recorded with it in the THL, such as the schema name or character set. Care should be taken when modifying SQL statement data to ensure that you are modifying the right part of the original statement. For example, a search and replace on an SQL statement should be made with care to ensure that embedded data is not altered by the process.

The key methods used for interacting with a `StatementData` object are listed below:

Method	Description
<code>getQuery()</code>	Returns the SQL statement
<code>setQuery()</code>	Updates the SQL statement
<code>appendToQuery()</code>	Appends a string to an existing query
<code>getDefaultSchema()</code>	Returns the default schema in which the statement was executed. The schema may be null for explicit or multi-schema queries.
<code>setDefaultSchema()</code>	Set the default schema for the SQL statement
<code>getTimestamp()</code>	Gets the timestamp of the query. This is required if data must be applied with a relative value by combining the timestamp with the relative value

Updating the SQL

The primary method of processing statement based data is to load and identify the original SQL statement (using `getQuery()`, update or modify the SQL statement string, and then update the statement within the THL again using `setQuery()`). For example:

```
sqlOriginal = d.getQuery();
sqlNew = sqlOriginal.replaceAll('NOTEPAD', 'notepad');
d.setQuery(sqlNew);
```

The above replaces the uppercase 'NOTEPAD' with a lowercase version in the query before updating the stored query in the object.

Changing the Schema Name

Some schema and other information is also provided in this structure. For example, the schema name is provided within the statement data and can be explicitly updated. In the example below, the schema "products" is updated to "nyc_products":

```
if (change.getDefaultSchema().compareTo("products") == 0)
{
    change.setDefaultSchema("nyc_products");
}
```

A similar operation should be performed for any row-based changes. A more complete example can be found in [Section 11.4.11, "abre-name.js Filter"](#).

11.5.1.4.4. `RowChangeData` Objects

`RowChangeData` is information that has been written into the THL in row format, and therefore consists of rows of individual data divided into the individual columns that make up each row-based change. Processing of these individual changes must be performed one row at a time using the list of `OneRowChange` [411] objects provided.

The following methods are supported for the `RowChangeData` object:

Method	Description
<code>appendOneRowChange(rowChange)</code>	Appends a single row change to the event, using the supplied <code>OneRowChange</code> [411] object.
<code>getRowChanges()</code>	Returns an array list of all the changes as <code>OneRowChange</code> [411] objects.
<code>setRowChanges(rowChanges)</code>	Sets the row changes within the event using the supplied list of <code>OneRowChange</code> objects.

For example, a typical row-based process will operate as follows:

```
if (d != null && d instanceof com.continuent.tungsten.replicator.dbms.RowChangeData)
{
    rowChanges = d.getRowChanges();
    for(j = 0; j < rowChanges.size(); j++)
    {
        oneRowChange = rowChanges.get(j);
        // Do row filter
    }
}
```

The `OneRowChange` [411] object contains the changes for just one row within the event. The class contains the information about the tables, field names and field values. The following methods are supported:

Method	Description
<code>getAction()</code>	Returns the row action type, i.e. whether the row change is an <code>INSERT</code> , <code>UPDATE</code> or <code>DELETE</code>
<code>getColumnSpec()</code>	Returns the specification of each column within the row change

Method	Description
<code>getColumnValues()</code>	Returns the value of each column within the row change
<code>getSchemaName()</code>	Gets the schema name of the row change
<code>getTableName()</code>	Gets the table name of the row change
<code>setColumnSpec()</code>	Sets the column specification using an array of column specifications
<code>setColumnValues()</code>	Sets the column values
<code>setSchemaName()</code>	Sets the schema name
<code>setTableName()</code>	Sets the table name

Changing Schema or Table Names

The schema, table and column names are exposed at different levels within the `OneRowChange` [411] object. Updating the schema name can be achieved by getting and setting the name through the `getSchemaName()` and `setSchemaName()` methods. For example, to add a prefix to a schema name:

```
rowchange.setSchemaName('prefix_' + rowchange.getSchemaName());
```

To update a table name, the `getTableName()` and `setTableName()` can be used in the same manner:

```
oneRowChange.setTableName('prefix_' + oneRowChange.getTableName());
```

Getting Action Types

Row operations are categorised according to the action of the row change, i.e. whether the change was an insert, update or delete operation. This information can be extracted from each row change by using the `getAction()` method:

```
action = oneRowChange.getAction();
```

The action information is returned as a string, i.e. `INSERT`, `UPDATE`, or `DELETE`. This enables information to be filtered according to the changes; for example by selectively modifying or altering events.

For example, `DELETE` events could be removed from the list of row changes:

```
for(j=0;j<rowChanges.size();j++)
{
    oneRowChange = rowChanges.get(j);
    if (oneRowChange.actionType == 'DELETE')
    {
        rowChanges.remove(j);
        j--;
    }
}
```

The `j--` is required because as each row change is removed, the size of the array changes and our current index within the array needs to be explicitly modified.

Extracting Column Definitions

To extract the row data, the `getColumnValues()` method returns the an array containing the value of each column in the row change. Obtaining the column specification information using `getColumnSpec()` returns a corresponding specification of each corresponding column. The column data can be used to obtain the column type information

To change column names or values, first the column information should be identified. The column information in each row change should be retrieved and/or updated. The `getColumnSpec()` returns the column specification of the row change. The information is returned as an array of the individual columns and their specification:

```
columns = oneRowChange.getColumnSpec();
```

For each column specification a `ColumnSpec` object is returned, which supports the following methods:

Method	Description
<code>getIndex()</code>	Gets the index of the column within the row change
<code>getLength()</code>	Gets the length of the column
<code>getName()</code>	Returns the column name if available
<code>getType()</code>	Gets the type number of the column
<code>getTypeDescription()</code>	

Method	Description
<code>isBlob()</code>	Returns true if the column is a blob
<code>isNotNull()</code>	Returns true if the column is configured as <code>NOT NULL</code>
<code>isUnsigned()</code>	Returns true if the column is unsigned.
<code>setBlob()</code>	Set the column blob specification
<code>setIndex()</code>	Set the column index order
<code>setLength()</code>	Returns the column length
<code>setName()</code>	Set the column name
<code>setNotNull()</code>	Set whether the column is configured as <code>NOT NULL</code>
<code>setSigned()</code>	Set whether the column data is signed
<code>setType()</code>	Set the column type
<code>setTypeDescription()</code>	Set the column type description

To identify the column type, use the `getType()` method which returns an integer matching the underlying data type. There are no predefined types, but common values include:

Type	Value	Notes
<code>INT</code>	4	
<code>CHAR OR VARCHAR</code>	12	
<code>TEXT OR BLOB</code>	2004	Use <code>isBlob()</code> to identify if the column is a blob or not
<code>TIME</code>	92	
<code>DATE</code>	91	
<code>DATETIME OR TIMESTAMP</code>	92	
<code>DOUBLE</code>	8	

Other information about the column, such as the length, and value types (unsigned, null, etc.) can be determined using the other functions against the column specification.

Extracting Row Data

The `getColumnValues()` method returns an array that corresponds to the information returned by the `getColumnSpec()` method. That is, the method returns a complementary array of the row change values, one element for each row, where each row is itself a further array of each column:

```
values = oneRowChange.getColumnValues();
```

This means that index 0 of the array from `getColumnSpec()` refers to the same column as index 0 of the array for a single row from `get-ColumnValues()`.

<code>getColumnSpec()</code>	<code>msgid</code>	<code>message</code>	<code>msgdate</code>
<code>getColumnValues()</code>			
[0]	1	Hello New York!	Thursday, June 13, 2013
[1]	2	Hello San Francisco!	Thursday, June 13, 2013
[2]	3	Hello Chicago!	Thursday, June 13, 2013

This enables the script to identify the column type by the index, and then the corresponding value update using the same index. In the above example, the `message` field will always be index 1 within the corresponding values.

Each value object supports the following methods:

Method	Description
<code>getValue()</code>	Get the current column value
<code>setValue()</code>	Set the column value to the supplied value
<code>setValueNull()</code>	Set the column value to NULL

For example, within the `zerodate2null` sample, dates with a zero value are set to NULL using the following code:

```
columns = oneRowChange.getColumnSpec();
columnValues = oneRowChange.getColumnValues();
for (c = 0; c < columns.size(); c++)
{
    columnSpec = columns.get(c);
    type = columnSpec.getType();

    if (type == TypesDATE || type == TypesTIMESTAMP)
    {
        for (row = 0; row < columnValues.size(); row++)
        {
            values = columnValues.get(row);
            value = values.get(c);

            if (value.getValue() == 0)
            {
                value.setValueNull()
            }
        }
    }
}
```

In the above example, the column specification is retrieved to determine which columns are date types. Then the list of embedded row values is extracted, and iterates over each row, setting the value for a date that is zero (0) to be NULL using the `setValueNull()` method.

An alternative would be to update to an explicit value using the `setValue()` method.

Chapter 12. Performance and Tuning

To help improve the performance of Tungsten Replication, a number of guides and tuning techniques are provided in this chapter. This may involve parameters entirely within Tungsten Replication, or changes and modifications to the parameters within the OS.

Tuning related to the Tungsten Replicator functionality

- [Section 12.1, “Block Commit”](#) — Increasing performance of replication solutions making use of block commit.

Tuning related to the network performance

- [Section 12.2, “Improving Network Performance”](#) — Increasing performance of networking between components by tuning OS parameters.

12.1. Block Commit

Introduced in 2.2.0. The commit size and interval settings were introduced in 2.2.0.

The replicator commits changes read from the THL and commits these changes in slaves during the applier stage according to the block commit size or interval. These replace the single `replicator.global.buffer.size` parameter that controls the size of the buffers used within each stage of the replicator.

When applying transactions to the database, the decision to commit a block of transactions is controlled by two parameters:

- When the event count reaches the specified event limit (set by `--svc-applier-block-commit-size` [366])
- When the commit timer reaches the specified commit interval (set by `--svc-applier-block-commit-interval` [366])

The default operation is for block commits to take place based on the transaction count. Commits by the timer are disabled. The default block commit size is 10 transactions from the incoming stream of THL data; the block commit interval is zero (0), which indicates that the interval is disabled.

When both parameters are configured, block commit occurs when either value limit is reached. For example, if the event count is set to 10 and the commit interval to 50s, events will be committed by the applier either when the event count hits 10 or every 50 seconds, whichever is reached first. This means, for example, that even if only one transaction exists, when the 50 seconds is up, that single transaction will be applied.

In addition, the execution of implied commits during specific events within the replicator can also be controlled to prevent fragmented block commits by using the `replicator.stage.q-to-dbms.blockCommitPolicy` property. This property can have either of the following values:

- `strict` — Commit block on service name changes, multiple fragments in a transaction, or `unsafe_for_block_commit`. This is the default setting.
- `lax` — Don't commit in any of these cases.

The block commit size can be controlled using the `--repl-svc-applier-block-commit-size` [366] option to `tpm`, or through the `blockCommitRowCount`.

The block commit interval can be controlled using the `--repl-svc-applier-block-commit-interval` [366] 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 [Vertica](#), larger block commit sizes and intervals may improve performance during the batch loading process:

```
shell> ./tools/tpm update alpha \
--rep1-svc-applier-block-commit-size=100000 \
--rep1-svc-applier-block-commit-interval=60s
```

This sets a large block commit size and interval enabling large batch loading.

12.1.1. Monitoring Block Commit Status

The block commit status can be monitored using the `treptl status -name tasks` command. This outputs the `lastCommittedBlockSize` and `lastCommittedBlockTime` values which indicate the size and interval (in seconds) of the last block commit.

```
shell> treptl status -name tasks
Processing status command (tasks)...
...
NAME          VALUE
-----
appliedLastEventId : mysql-bin.000015:0000000000001117:0
appliedLastSeqno : 5271
appliedLatency   : 4656.231
applyTime        : 0.066
averageBlockSize : 0.500
cancelled       : false
commits         : 10
currentBlockSize : 0
currentLastEventId : mysql-bin.000015:0000000000001117:0
currentLastFragno : 0
currentLastSeqno : 5271
eventCount       : 5
extractTime      : 0.394
filterTime       : 0.017
lastCommittedBlockSize: 1
lastCommittedBlockTime: 0.033
otherTime        : 0.001
stage            : q-to-dbms
state            : extract
taskId           : 0
Finished status command (tasks)...
```

12.2. Improving Network Performance

The performance of the network can be critical when replicating data. The information transferred over the network contains the full content of the THL in addition to a small protocol overhead. Improving your network performance can have a significant impact on the overall performance of the replication process.

The following network parameters should be configured within your `/etc/sysctl.conf` and can safely applied to all the hosts within your cluster deployments:

```
# Increase size of file handles and inode cache
fs.file-max = 2097152

# tells the kernel how many TCP sockets that are not attached to any
# user file handle to maintain. In case this number is exceeded,
# orphaned connections are immediately reset and a warning is printed.
net.ipv4.tcp_max_orphans = 60000

# Do not cache metrics on closing connections
net.ipv4.tcp_no_metrics_save = 1

# Turn on window scaling which can enlarge the transfer window:
net.ipv4.tcp_window_scaling = 1

# Enable timestamps as defined in RFC1323:
net.ipv4.tcp_timestamps = 1

# Enable select acknowledgments:
net.ipv4.tcp_sack = 1

# Maximum number of remembered connection requests, which did not yet
# receive an acknowledgment from connecting client.
net.ipv4.tcp_max_syn_backlog = 10240

# recommended default congestion control is htcp
net.ipv4.tcp_congestion_control=htcp
```

```

# recommended for hosts with jumbo frames enabled
net.ipv4.tcp_mtu_probing=1

# Number of times SYNACKs for passive TCP connection.
net.ipv4.tcp_synack_retries = 2

# Allowed local port range
net.ipv4.ip_local_port_range = 1024 65535

# Protect Against TCP Time-Wait
net.ipv4.tcp_rfc1337 = 1

# Decrease the time default value for tcp_fin_timeout connection
net.ipv4.tcp_fin_timeout = 15

# Increase number of incoming connections
# somaxconn defines the number of request_sock structures
# allocated per each listen call. The
# queue is persistent through the life of the listen socket.
net.core.somaxconn = 1024

# Increase number of incoming connections backlog queue
# Sets the maximum number of packets, queued on the INPUT
# side, when the interface receives packets faster than
# kernel can process them.
net.core.netdev_max_backlog = 65536

# Increase the maximum amount of option memory buffers
net.core.optmem_max = 25165824

# Increase the maximum total buffer-space allocatable
# This is measured in units of pages (4096 bytes)
net.ipv4.tcp_mem = 65536 131072 262144
net.ipv4.udp_mem = 65536 131072 262144

### Set the max OS send buffer size (wmem) and receive buffer
# size (rmem) to 12 MB for queues on all protocols. In other
# words set the amount of memory that is allocated for each
# TCP socket when it is opened or created while transferring files

# Default Socket Receive Buffer
net.core.rmem_default = 25165824

# Maximum Socket Receive Buffer
net.core.rmem_max = 25165824

# Increase the read-buffer space allocatable (minimum size,
# initial size, and maximum size in bytes)
net.ipv4.tcp_rmem = 20480 12582912 25165824
net.ipv4.udp_rmem_min = 16384

# Default Socket Send Buffer
net.core.wmem_default = 25165824

# Maximum Socket Send Buffer
net.core.wmem_max = 25165824

# Increase the write-buffer-space allocatable
net.ipv4.tcp_wmem = 20480 12582912 25165824
net.ipv4.udp_wmem_min = 16384

# Increase the tcp-time-wait buckets pool size to prevent simple DOS attacks
net.ipv4.tcp_max_tw_buckets = 1440000
# net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1

```

12.3. Tungsten Replicator Block Commit and Memory Usage

Replicators are implemented as Java processes, which use two types of memory: stack space, which is allocated per running thread and holds objects that are allocated within individual execution stack frames, and heap memory, which is where objects that persist across individual method calls live. Stack space is rarely a problem for Tungsten as replicators rarely run more than 200 threads and use limited recursion. The Java defaults are almost always sufficient. Heap memory on the other hand runs out if the replicator has too many transactions in memory at once. This results in the dreaded Java OutOfMemory exception, which causes the replicator to stop operating. When this happens you need to look at tuning the replicator memory size.

To understand replicator memory usage, we need to look into how replicators work internally. Replicators use a "pipeline" model of execution that streams transactions through 1 or more concurrently executing stages. As you can 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 indepen-

dently. This streaming model is quite efficient and normally permits Tungsten to transfer even exceedingly large transactions, as the replicator breaks them up into smaller pieces called transaction fragments.

The pipeline model has consequences for memory management. First of all, replicators are doing many things at one, hence need enough memory to hold all current objects. Second, the replicator works fastest if the in-memory queues between stages are large enough that they do not ever become empty. This keeps delays in upstream processing from delaying things at the end of the pipeline. Also, it allows replicators to make use of block commit. Block commit is an important performance optimization in which stages try to commit many transactions at once on slaves to amortize the cost of commit. In block commit the end stage continues to commit transactions until it either runs out of work (i.e., the upstream queue becomes empty) or it hits the block commit limit. Larger upstream queues help keep the end stage from running out of work, hence increase efficiency.

Bearing this in mind, we can alter replicator behavior in a number of ways to make it use less memory or to handle larger amounts of traffic without getting a Java OutOfMemory error. You should look at each of these when tuning memory:

- Property `wrapper.java.memory` in file `wrapper.conf`. This controls the amount of heap memory available to replicators. 1024 MB is the minimum setting for most replicators. Busy replicators, those that have multiple services, or replicators that use parallel apply should consider using 2048 MB instead. If you get a Java OutOfMemory exception, you should first try raising the current setting to a higher value. This is usually enough to get past most memory-related problems. You can set this at installation time as the `--repl-java-mem-size` [349] parameter.
- Property `replicator.global.buffer.size` in the replicator properties file. This controls two things, the size of in-memory queues in the replicator as well as the block commit size. If you still have problems after increasing the heap size, try reducing this value. It reduces the number of objects simultaneously stored on the Java heap. A value of 2 is a good setting to try to get around temporary problems. This can be set at installation time as the `--repl-buffer-size` [331] 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` [366] parameter.
- Property `replicator.extractor.dbms.transaction_frag_size` in the `replicator.properties` file. This parameter controls the size of fragments for long transactions. Tungsten automatically breaks up long transactions into fragments. This parameter controls the number of bytes of binlog per transaction fragment. You can try making this value smaller to reduce overall memory usage if many transactions are simultaneously present. Normally however this value has minimal impact.

Finally, it is worth mentioning that the main cause of out-of-memory conditions in replicators is large transactions. In particular, Tungsten cannot fragment individual statements or row changes, so changes to very large column values can also result in OutOfMemory conditions. For now the best approach is to raise memory, as described above, and change your application to avoid such transactions.

The replicator commits changes read from the THL and commits these changes in 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.

In addition, the execution of implied commits during specific events within the replicator can also be controlled to prevent fragmented block commits by using the `replicator.stage.q-to-dbms.blockCommitPolicy` property. This property can have either of the following values:

- `strict` — Commit block on service name changes, multiple fragments in a transaction, or `unsafe_for_block_commit`. This is the default setting.
- `lax` — Don't commit in any of these cases.

The block commit size can be controlled using the `--repl-svc-applier-block-commit-size` [366] option to `tpm`, or through the `blockCommitRowCount`.

The block commit interval can be controlled using the `--repl-svc-applier-block-commit-interval` [366] 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 [Vertica](#), larger block commit sizes and intervals may improve performance during the batch loading process:

```
shell> ./tools/tpm update alpha \
    --repl-svc-applier-block-commit-size=100000 \
    --repl-svc-applier-block-commit-interval=60s
```

This sets a large block commit size and interval enabling large batch loading.

Appendix A. Release Notes

A.1. Tungsten Replicator 5.0.1 GA (23 Feb 2017)

Tungsten Replication 5.0.1 is a bugfix release that contains critical fixes and improvements from the Tungsten Replication 5.0.0 release. Specifically, it changes the default security and other settings to make upgrades from previous releases easier, and other fixes and improvements to the Oracle support and command-line tools.

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:

- The default security configuration for new installations is for security, including SSL and TLS and authentication, to be disabled. In 5.0.0 the default was to enable full security on all components which could lead to problems and difficulty when upgrading.

Issues: CT-18

- The Ruby Net::SSH module, which has been bundled with Tungsten Replication in past releases, is no longer included. This is due to the wide range of Ruby versions and deployment environments that we support, and differences in the Net::SSH module supported and used with different Ruby versions. In order to simplify the process and ensure that the platforms we support operate correctly, the Net::SSH module has been removed and will now need to be installed before deployment.

To ensure you have the correct environment before deployment, ensure both the Net::SSH and Net::SCP Ruby modules are installed using `gem`:

```
shell> gem install net-ssh  
shell> gem install net-scp
```

Depending on your environment, you may also need to install the `io-console` module:

```
shell> gem install io-console
```

If during installation you get an error similar to this:

```
mkmf.rb can't find header files for ruby at /usr/lib/ruby/include/ruby.h
```

It indicates that you do not have the Ruby development headers installed. Use your native package management interface (for example `yum` or `apt`) and install the `ruby-dev` package. For example:

```
shell> sudo apt install ruby-dev
```

Issues: CT-88

- The `replicator` is no longer restarted when updating the configuration with `tpm` when using the `--replace-tls-certificate` option.

Issues: CT-120

- For compatibility with MySQL 5.7, the `tpm` command will now check for the `super_read_only` setting and warn if this setting is enabled.

Issues: CONT-1039

- For compatibility with MySQL 5.7, the `tpm` command will use the `authentication_string` field for validating passwords.

Issues: CONT-1058

- For compatibility with MySQL 5.7, the `tpm` command will now ignore the `sys` schema.

Issues: CONT-1059

Improvements, new features and functionality

- Installation and Deployment

- Tungsten Replication is now certified for deployment on systems running Java 8.

Issues: CT-27

- **Core Replicator**

- The replicator will now generate a detailed heap dump in the event of a failure. This will help during debugging and identifying any issues.

Issues: CT-11

- **Filters**

- The Rhino JS, which is incorporated for use by the filtering and batch loading mechanisms, has been updated to Rhino 1.7R4. This addresses a number of different issues with the embedded library, including a performance issue that could lead to increased latency during filter operations.

Issues: CT-21

Bug Fixes

- **Installation and Deployment**

- The Ruby Net::SSH libraries used by `tpm` have been updated to the latest version. This addresses issues with SSH and staging based deployments, including KEX algorithm errors.

Issues: CT-16

- On some platforms the `keytool` command could fail to be found, causing an error within the installation when generating certificates.

Issues: CT-73

- **Command-line Tools**

- The `tpasswd` could create a log file with the wrong permissions.

Issues: CT-117

- **Core Replicator**

- Checksums in MySQL could cause problems when parsing the MySQL binary log due to a change in the way the checksum information is recorded within the binary log. This would cause the replicator to become unable to come online.

Issues: CT-72

A.2. Tungsten Replicator 5.0.0 GA (7 December 2015)

VMware Continuent for Replication 5.0.0 is a major release that incorporates the following changes:

- The software release has been renamed. For most users of VMware Continuent for Replication, the filename will start with `vmware-continuent-replication`. If you are using an Oracle DBMS as the source and have purchased support for the latest version, the filename will start with `vmware-continuent-replication-oracle-source`.

The documentation has not been updated to reflect this change. While reading these examples you will see references to `tungsten-replicator` which will apply to your software release.

- New Oracle Extraction module that reads the Oracle Redo logs provided faster, more compatible, and more efficient method for extracting data from Oracle databases. For more information, see [Section 5.1, “Oracle Replication using Redo Reader”](#).
- Security, including file permissions and TLS/SSL is now enabled by default. For more information, see [Section 7.5, “Deployment Security”](#).
- License keys are now required during installation. For more information, see [Section 2.4, “Deploy License Keys”](#).
- Support for RHEL 7 and CentOS 7.
- Basic support for MySQL 5.7.
- Cleaner and simpler directory layout.

Upgrading from previous versions should be fully tested before attempted in a production environment. The changes listed below affect `tpm` output and the requirements for operation.

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:

- Tungsten Replication now enables security by default. Security includes:
 - Authentication between command-line tools (`trepctl`) and background services.
 - SSL/TLS between command-line tools and background services.
 - SSL/TLS between Tungsten Replicator and datasources.
 - File permissions and access by all components.

The security changes require a certificate file to be generated prior to operation. The `tpm` command can do that during upgrade if you are using a staging directory. Alternatively, you can [create the certificate](#) and update your configuration with the corresponding argument. This is required if you are installing from an INI file. See [Section 7.5.3, “Installing from a Staging Host with Manually Generated Certificates”](#) or [Section 7.5.4, “Installing via INI File with Manually Generated Certificates”](#) for more information. This functionality may be disabled by adding `--disable-security-controls` [342] to your configuration.

If you would like `tpm` to generate the necessary certificate from the staging directory. Run `tpm update` with the `--replace-tls-certificate` option.

```
staging-shell> ./tools/tpm update --replace-tls-certificate
```

For more information, see [Section 7.5, “Deployment Security”](#).

- Tungsten Replication now requires license keys in order to operate.

License keys are provided to all customers with an active support contract. Login to [my.vmware.com](#) to identify your support contract and the associated license keys. After collecting the license keys, they should be placed into `/etc/tungsten/continuent.licenses` or `/opt/continuent/share/continuent.licenses`. The `/opt/continuent` path should be replaced with your value for `--install-directory` [347]. Place each license on a new line in the file and make sure it is readable by the tungsten system user.

If you are testing VMware Continuent or don't have your license key, talk with your sales contact for assistance. You may enable a trial-mode by using the license key `TRIAL`. This will not affect the runtime operation of VMware Continuent but may impact your ability to get rapid support.

The `tpm` script will display a warning if license keys are not provided or if the provided license keys are not valid.

- For compatibility with MySQL 5.7, the `tpm` command will now check for the `super_read_only` setting and warn if this setting is enabled.

Issues: CONT-1039

- For compatibility with MySQL 5.7, the `tpm` command will use the `authentication_string` field for validating passwords.

Issues: CONT-1058

- For compatibility with MySQL 5.7, the `tpm` command will now ignore the `sys` schema.

Issues: CONT-1059

- Tungsten Replication now includes `RELEASE_NOTES` in the package and displays a warning if they have not been reviewed.

During some `tpm` commands, the script will check to see if the release notes have been reviewed and accepted. This may be done by running `tools/accept_release_notes` from the staging directory. The script will display the information and prompt the user for acceptance. A hidden file will be created on the staging server to mark the release notes have been accepted and the warning will not be displayed.

This process may be automated by calling `tools/accept_release_notes -y` prior to installation. The script will mark the release notes as accepted and the warning will not be displayed.

Issues: CONT-1122

Improvements, new features and functionality

- Installation and Deployment

- During installation, `tpm` writes the configuration log to `/tmp/tungsten-configure.log`. If the file exists, but is owned by a separate user the operation will fail with a Permission Denied error. The operation has now been updated to create a directory within `/tmp` with the name of the current user where the configuration log will be stored. For example, if the user is `tungsten`, the log will be written to `/tmp/tungsten/tungsten-configure.log`.

Issues: CONT-1402

- **Oracle Replication**

- The replicator will automatically determine if the Oracle JDBC driver is available within `$ORACLE_HOME/jdbc/lib` or the current path, and will copy it into the distribution directory during installation if available.

Issues: CONT-1344

Bug Fixes

- **Installation and Deployment**

- During installation, a failed installation by `tpm`, running `tpm uninstall` could also fail. The command now correctly uninstalls even a partial installation.

Issues: CONT-1359

Known Issues

- **Oracle Replication**

- The user configuration for Oracle users required when enabling Oracle extraction has a number of rules that must be followed to ensure valid replication:

The replication user (configured with `--replication-user` [363]) has the following rules

- The user should not contain data that will be replicated to other hosts.
- If the user contains replicated data and filters are used, the results of replication cannot be guaranteed.
- A different replication user must be used for each service extracting from the same Oracle Database.

Issues: CONT-1403

Tungsten Replicator 5.0.0 Includes the following changes from Tungsten Replicator 5.0.0

Behavior Changes

The following changes have been made to Tungsten Replicator and may affect existing scripts and integration tools. Any scripts or environment which make use of these tools should check and update for the new configuration:

- The Bristlecone load generator toolkit is no longer included with Tungsten Replicator by default.

Issues: CONT-903

- The scripts previously located within the `scripts` directory have now been relocated to the standard `bin` directory. This does not affect their availability if the `env.sh` script has been used to update your path. This includes, but is not limited to, the following commands:

- `ebs_snapshot.sh`
- `file_copy_snapshot.sh`
- `multi_trepctl`
- `tungsten_get_position`
- `tungsten_provision_slave`
- `tungsten_provision_thl`
- `tungsten_read_master_events`
- `tungsten_set_position`

- `xtrabackup.sh`
- `xtrabackup_to_slave`

Issues: CONT-904

- The `backup` and `restore` functionality in `trepctl` has been deprecated and will be removed in a future release.

Issues: CONT-906

- The location of the JavaScript filters has been moved to new location in keeping with the rest of the configuration:

- `samples/extensions/javascript` has moved to `support/filters-javascript`
- `samples/scripts/javascript-advanced` has moved to `support/filters-javascript`

The use of these filters has not changed but the default location for some filter configuration files has moved to `support/filters-config`. Check your current configuration before upgrading.

Issues: CONT-908

- The `ddlscan` templates have been moved to the `support/ddlscan` directory.

Issues: CONT-909

- The Vertica applier should write exceptions to a temporary file during replication.

The applier statements will include the `EXCEPTIONS` attribute in each statement to assist in debugging. Review the replicator log or `trepctl status` output for more details.

Issues: CONT-1169

Known Issues

The following issues may affect the operation of Tungsten Replication and should be taken into account when deploying or updating to this release.

- **Core Replicator**

- The replicator can hit a MySQL lock wait timeout when processing large transactions.

Issues: CONT-1106

- The replicator can run into OutOfMemory when handling very large Row-Based replication events. This can be avoided by setting `--optimize-row-events=false` [356].

Issues: CONT-1115

- The replicator can fail during `LOAD DATA` commands or Vertica loading if the system permissions are not set correctly. If this is encountered, make sure the MySQL or Vertica system users are a member of the Tungsten system group. The issue may also be avoided by removing system file protections with `--file-protection-level=none` [346].

Issues: CONT-1460

Improvements, new features and functionality

- **Command-line Tools**

- The `dsctl` has been updated to provide help output when specifically requested with the `-h` or `-help` options.

Issues: CONT-1003

For more information, see [Section 9.6.4, “dsctl help Command”](#).

Bug Fixes

- **Core Replicator**

- A master replicator could fail to finish extracting a fragmented transaction if disconnected during processing.

Issues: CONT-1163

- A slave replicator could fail to come [ONLINE \[207\]](#) if the last THL file is empty.

Issues: CONT-1164

- The replicator applier and filters may fail with ORA-955 because the replicator did not check for metadata tables using uppercase table names.

Issues: CONT-1375

Appendix B. Prerequisites

Before you install Tungsten Replication, there are a number of setup and prerequisite installation and configuration steps that must have taken place before any installation can continue. [Section B.2, "Staging Host Configuration"](#) and [Section B.3, "Host Configuration"](#) must be performed on every host within your chosen cluster or replication configuration. Additional steps are required to configure explicit databases, such as [Section B.4, "MySQL Database Setup"](#), and will need to be performed on each appropriate host.

B.1. Requirements

B.1.1. Operating Systems Support

Operating System	Variant	Status	Notes
Linux	RedHat/CentOS	Primary platform	RHEL 4, 5, and 6 as well as CentOS 5.x and 6.x versions are fully supported. CentOS 7 is supported in 5.0.1 and higher.
Linux	Ubuntu	Primary platform	Ubuntu 9.x-13.x versions are fully supported.
Linux	Debian/Suse/Other	Secondary Platform	Other Linux platforms are supported but are not regularly tested. We will fix any bugs reported by customers.
Solaris		Secondary Platform	Solaris 10 is fully supported. OpenSolaris is not supported at this time.
Mac OS X		Secondary platform	Mac OS/X Leopard and Snow Leopard are used for development at Continuent but not certified. We will fix any bugs reported by customers.
Windows		Limited Support	Tungsten 1.3 and above will support Windows platforms for connectivity (Tungsten Connector and SQL Router) but may require manual configuration. Tungsten clusters do not run on Windows.
BSD		Limited Support	Tungsten 1.3 and above will support BSD for connectivity (Tungsten Connector and SQL Router) but may require manual configuration. Tungsten clusters do not run on BSD.

B.1.2. Database Support

Database	Version	Support Status	Notes
MySQL	5.0, 5.1, 5.5, 5.6	Primary platform	Statement and row based replication is supported. MyISAM and InnoDB table types are fully supported; InnoDB tables are recommended.
MySQL	5.7	Primary platform	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.
Percona	5.5, 5.6	Primary platform	
MariaDB	5.5, 10.0	Primary platform	
Oracle (CDC)	10g Release 2 (10.2.0.5), 11g	Primary Platform	Synchronous CDC is supported on Standard Edition only; Synchronous and Asynchronous are supported on Enterprise Editions
Oracle (Redo Reader)	9.2, 10g, 11g, 12c	Primary Platform	Redo Reader reads data direct from the Oracle redo logs.
Drizzle		Secondary Platform	Experimental support for Drizzle is available. Drizzle replication is not tested.

B.1.3. RAM Requirements

RAM requirements are dependent on the workload being used and applied, but the following provide some guidance on the basic RAM requirements:

- Tungsten Replicator requires 2GB of VM space for the Java execution, including the shared libraries, with approximate 1GB of Java VM heapspace. This can be adjusted as required, for example, to handle larger transactions or bigger commit blocks and large packets.

Performance can be improved within the Tungsten Replicator if there is a 2-3GB available in the OS Page Cache. Replicators work best when pages written to replicator log files remain memory-resident for a period of time, so that there is no file system I/O required to read that data back within the replicator. This is the biggest potential point of contention between replicators and DBMS servers.

B.1.4. Disk Requirements

Disk space usage is based on the space used by the core application, the staging directory used for installation, and the space used for the THL files:

- The staging directory containing the core installation is approximately 150MB. When performing a staging-directory based installation, this space requirement will be used once. When using a INI-file based deployment, this space will be required on each server. For more information on the different methods, see [Section 10.1, “Comparing Staging and INI tpm Methods”](#).
- Deployment of a live installation also requires approximately 150MB.
- The THL files required for installation are based on the size of the binary logs generated by MySQL. THL size is typically twice the size of the binary log. This space will be required on each machine in the cluster. The retention times and rotation of THL data can be controlled, see [Section D.1.5, “The thl Directory”](#) for more information, including how to change the retention time and move files during operation.

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 Replication is recommended to ensure that a full disk does not impact your OS or DBMS. Because the replicator reads and writes information using buffered I/O in a serial fashion, spinning disk and Network Attached Storage (NAS) is suitable for storing THL, as there is no random-access or seeking.

B.1.5. Java Requirements

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

- Oracle JVM/JDK 6
- Oracle JVM/JDK 7
- Oracle JVM/JDK 8
- OpenJDK 6
- OpenJDK 7
- OpenJDK 8

B.1.6. Cloud Deployment Requirements

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

Instance Types/Configuration

Attribute	Guidance	Amazon Example
Instance Type	Instance sizes and types are dependent on the workload, but larger instances are recommended for transactional databases.	m1.xlarge or better
Instance Boot Volume	Use block, not ephemeral storage.	EBS
Instance Deployment	Use standard Linux distributions and bases. For ease of deployment and configuration, use Puppet .	Amazon Linux AMIs

Development/QA nodes should always match the expected production environment.

AWS/EC2 Deployments

- Use Virtual Private Cloud (VPC) deployments, as these provide consistent IP address support.
- Multiple EBS-optimized volumes for data, using Provisioned IOPS for the EBS volumes depending on workload:

Parameter	tpm Option	tpm Value	MySQL my.cnf Option	MySQL Value
/ (root)				

Parameter	tpm Option	tpm Value	MySQL my.cnf Option	MySQL Value
MySQL Data	<code>datasource-mysql-data-directory</code> [337]	/volumes/mysql/data	<code>datadir</code>	/volumes/mysql/data
MySQL Binary Logs	<code>datasource-log-directory</code> [337]	/volumes/mysql/binlogs	<code>log-bin</code>	/volumes/mysql/bin-logs/mysql-bin
Transaction History Logs (THL)	<code>th1-directory</code> [368]	/volumes/mysql/thl		

Recommended Replication Formats

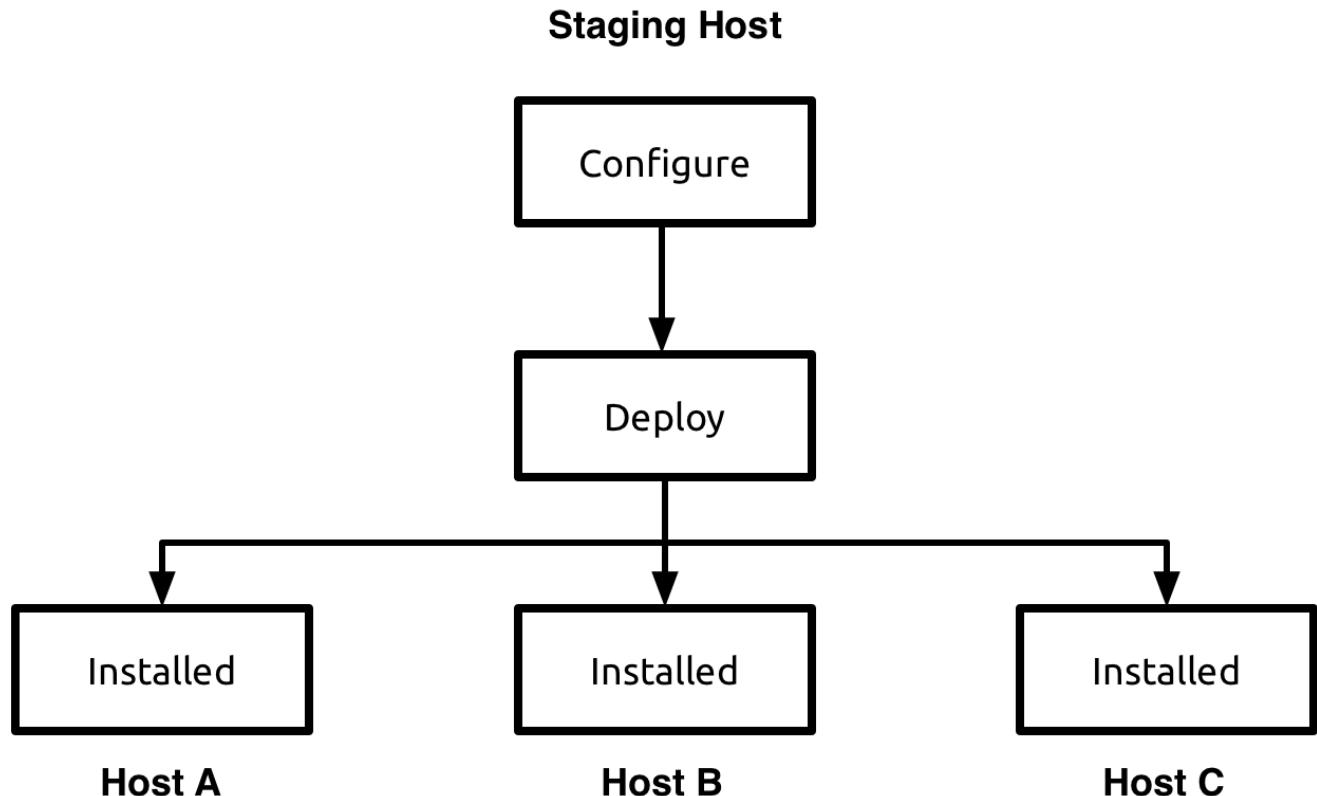
- `MIXED` is recommended for MySQL master/slave topologies (e.g., either single clusters or primary/data-recovery setups).
- `ROW` is strongly recommended for multi-master setups. Without `ROW`, data drift is a possible problem when using `MIXED` or `STATEMENT`. Even with `ROW` there are still cases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

B.2. Staging Host Configuration

The staging host will form the base of your operation for creating your cluster. The primary role of the staging host is to hold the Tungsten Replication™ software, and to install, transfer, and initiate the Tungsten Replication™ 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 Replication™ 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 Replication™. 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 tungsten
```

2. Login as the `tungsten` user:

```
shell> su - tungsten
```

3. Create an SSH key file, but do not configure a password:

```
tungsten:shell> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/tungsten/.ssh/id_rsa):
Created directory '/home/tungsten/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/tungsten/.ssh/id_rsa.
Your public key has been saved in /home/tungsten/.ssh/id_rsa.pub.
The key fingerprint is:
e3:fa:e9:7a:9d:d9:3d:81:36:63:85:cb:a6:f8:41:3b tungsten@staging
The key's randomart image is:
+---[ RSA 2048]----+
| |
| |
| . .
| S .. +
| . o .X .
| .EO + .
| .o=o. o
| o=+.. .
+-----+
```

This creates both a public and private keyfile; the public keyfile will be shared with the hosts in the cluster to allow hosts to connect to each other.

4. Within the staging server, profiles for the different cluster configurations are stored within a single directory. You can simplify the management of these different services by configuring a specific directory where these configurations will be stored. To set the directory, specify the directory within the `$CONTINUENT_PROFILES` environment variable, adding this variable to your shell startup script (`.bashrc`, for example) within your staging server.

```
shell> mkdir -p /opt/continuent/software/conf
shell> mkdir -p /opt/continuent/software/replicator.conf
shell> export CONTINUENT_PROFILES=/opt/continuent/software/conf
shell> export REPLICATOR_PROFILES=/opt/continuent/software/replicator.conf
```

We now have a staging server setup, an SSH keypair for our login information, and are ready to start setting up each host within the cluster.

B.3. Host Configuration

Each host in your cluster must be configured with the `tungsten` user, have the SSH key added, and then be configured to ensure the system and directories are ready for the Tungsten services to be installed and configured.

There are a number of key steps to the configuration process:

- Creating a user environment for the Tungsten service
- Creating the SSH authorization for the user on each host
- Configuring the directories and install locations
- Installing necessary software and tools
- Configuring `sudo` access to enable the configured user to perform administration commands

Important

The operations in the following sections must be performed on each host within your cluster. Failure to perform each step may prevent the installation and deployment of Tungsten cluster.

B.3.1. Creating the User Environment

The `tungsten` user should be created with a home directory that will be used to hold the Tungsten distribution files (not the installation files), and will be used to execute and create the different Tungsten services.

For Tungsten to work correctly, the `tungsten` user must be able to open a larger number of files/sockets for communication between the different components and processes as . You can check this by using `ulimit`:

```
shell> ulimit -n
core file size          (blocks, -c) 0
data seg size            (kbytes, -d) unlimited
file size                (blocks, -f) unlimited
max locked memory        (kbytes, -l) unlimited
max memory size          (kbytes, -m) unlimited
open files               (-n) 256
pipe size                (512 bytes, -p) 1
stack size               (kbytes, -s) 8192
cpu time                 (seconds, -t) unlimited
max user processes        (-u) 709
virtual memory            (kbytes, -v) unlimited
```

The system should be configured to allow a minimum of 65535 open files. You should configure both the `tungsten` user and the database user with this limit by editing the `/etc/security/limits.conf` file:

```
tungsten    -    nofile    65535
mysql       -    nofile    65535
```

In addition, the number of running processes supported should be increased to ensure that there are no restrictions on the running processes or threads:

```
tungsten    -    nproc     8096
mysql       -    nproc     8096
```

You must logout and log back in again for the `ulimit` changes to take effect.

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.mysql`):

```
/tmp/** rwx
```

B.3.2. Configuring Network and SSH Environment

The hostname, DNS, IP address and accessibility of this information must be consistent. For the cluster to operate successfully, each host must be identifiable and accessible to each other host, either by name or IP address.

Individual hosts within your cluster must be reachable and most conform to the following:

- Do not use the `localhost` or `127.0.0.1` addresses.
- Do not use Zeroconf (`.local`) addresses. These may not resolve properly or fully on some systems.
- The server hostname (as returned by the `hostname`) must match the names you use when configuring your service.
- The IP address that resolves on the hostname for that host must resolve to the IP address (not `127.0.0.1`). The default configuration for many Linux installations is for the hostname to resolve to the same as `localhost`:

```
127.0.0.1 localhost
127.0.0.1 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:

- Confirm the hostname:

```
shell> uname -n
```

Warning

The hostname cannot contain underscores.

- Confirm the IP address:

```
shell> hostname --ip-address
```

- Confirm that the hostnames of the other hosts in the cluster resolve correctly to a valid IP address. You should confirm on each host that you can identify and connect to each other host in the planned cluster:

```
shell> nslookup host1
shell> ping host1
```

If the host does not resolve, either ensure that the hosts are added to the DNS service, or explicitly add the information to the `/etc/hosts` file.

Warning

If using `/etc/hosts` then you must ensure that the information is correct and consistent on each host, and double check using the above method that the IP address resolves correctly for every host in the cluster.

B.3.2.1. Network Ports

The following network ports should be open between specific hosts to allow communication between the different components:

Component	Source	Destination	Port	Purpose
Database Service	Database Host	Database Host	7	Checking availability
#	#	#	2112	THL replication
#	#	#	10000-10001	Replication connection listener port

Component	Port	Purpose
#	2114	THL replication
#	10002-10003	Replication connection listener ports
Client Application	13306	MySQL port for Connectivity

Component	Port	Purpose
Manager Hosts	7	Communication between managers within multi-site, multi-master clusters

If a system has a firewall enabled, in addition to enabling communication between hosts as in the table above, the localhost must allow port-to-port traffic on the loopback connection without restrictions. For example, using `iptables` this can be enabled using the following command rule:

```
shell> iptables -A INPUT -i lo -m state --state NEW -j ACCEPT
```

B.3.2.2. SSH Configuration

For password-less SSH to work between the different hosts in the cluster, you need to copy both the public and private keys between the hosts in the cluster. This will allow the staging server, and each host, to communicate directly with each other using the designated login.

To achieve this, on each host in the cluster:

1. Copy the public (`.ssh/id_rsa.pub`) and private key (`.ssh/id_rsa`) from the staging server to the `~tungsten/.ssh` directory.
2. Add the public key to the `.ssh/authorized_keys` file.

```
shell> cat .ssh/id_rsa.pub >> .ssh/authorized_keys
```

3. Ensure that the file permissions on the `.ssh` directory are correct:

```
shell> chmod 700 ~./ssh
shell> chmod 600 ~./ssh/*
```

With each host configured, you should try to connecting to each host from the staging server to confirm that the SSH information has been correctly configured. You can do this by connecting to the host using `ssh`:

```
tungsten:shell> ssh tungsten@host
```

You should have logged into the host at the `tungsten` home directory, and that directory should be writable by the `tungsten` user.

B.3.3. Directory Locations and Configuration

On each host within the cluster you must pick, and configure, a number of directories to be used by Tungsten Replication™, 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 Replication™ needs to be installed in a specific directory. The recommended solution is to use `/opt/continuent`. This information will be required when you configure the cluster service.

The directory should be created, and the owner and permissions set for the configured user:

```
shell> sudo mkdir /opt/continuent
shell> sudo chown tungsten /opt/continuent
shell> sudo chmod 700 /opt/continuent
```

- Home Directory

The home directory of the `tungsten` user must be writable by that user.

B.3.4. Configure Software

Tungsten Replication™ relies on the following software. Each host must use the same version of each tool.

Software	Versions Supported	Notes
Ruby	1.8.7, 1.9.3, or 2.0.0 or higher ^a	JRuby is not supported

Software	Versions Supported	Notes
Ruby OpenSSL Module	-	Checking using <code>ruby -ropenssl -e 'p "works"'</code>
Ruby Gems	-	
Ruby <code>io-console</code> module	-	Install using <code>gem install io-console</code>
Ruby <code>net-ssh</code> module	-	Install using <code>gem install net-ssh</code> ^b
Ruby <code>net-scp</code> module	-	Install using <code>gem install net-scp</code>
GNU tar	-	
Java Runtime Environment	Java SE 7 (or compatible), Java SE 8 (or compatible) is supported in 5.0.1 and higher	

^a Ruby 1.9.1 and 1.9.2 are not supported; these releases remove the execute bit during installation.

^b For Ruby 1.8.7 the minimum version of net-ssh is 2.5.2, install using `gem install net-ssh -v 2.5.2`

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
java version "1.7.0_21"
OpenJDK Runtime Environment (IcedTea 2.3.9) (7u21-2.3.9-1ubuntul)
OpenJDK 64-Bit Server VM (build 23.7-b01, mixed mode)
```

Tungsten Replication is known to work with Java 1.7 and using the following JVMs:

- Oracle JVM/JDK 7
- OpenJDK 7

On certain environments, a separate tool such as `alternatives` (RedHat/CentOS) or `update-alternatives` (Debian/Ubuntu) may need to be used to switch Java versions globally or for individual users. For example, within CentOS:

```
shell> alternatives --display
```

Important

It is recommended to switch off all automated software and operating system update procedures. These can automatically install and restart different services which may be identified as failures by Tungsten Replicator. Software and Operating System updates should be handled by following the appropriate [Section 8.13, “Performing Database or OS Maintenance”](#) procedures.

It also recommended to install `ntp` or a similar time synchronization tool so that each host in the cluster has the same physical time.

B.3.5. sudo Configuration

Tungsten requires that the user you have configured to run the server has `sudo` credentials so that it can run and install services as `root`.

Within Ubuntu you can do this by editing the `/etc/sudoers` file using `visudo` and adding the following lines:

```
Defaults:tungsten      !authenticate
...
## Allow tungsten to run any command
tungsten ALL=(ALL)    ALL
```

For a secure environment where `sudo` access is not permitted for all operations, a minimum configuration can be used:

```
tungsten ALL=(ALL)
```

`sudo` can also be configured to handle only specific directories or files. For example, when using `xtrabackup`, or additional tools in the Tungsten toolkit, such as `tungsten_provision_slave`, additional commands must be added to the permitted list:

```
tungsten ALL=(ALL) NOPASSWD: /sbin/service, /usr/bin/innobackupex, /bin/rm, >
```

```
/bin/mv, /bin/chown, /bin/chmod, /usr/bin/scp, /bin/tar, /usr/bin/which, »
/etc/init.d/mysql, /usr/bin/test, »
/apps/tungsten/continuent/tungsten/tungsten-replicator/scripts/xtrabackup.sh, »
/apps/tungsten/continuent/tungsten/tools/tpm, /usr/bin/innobackupex-1.5.1, »
/bin/cat, /bin/find
```

Within Red Hat Linux add the following line:

```
tungsten ALL=(root) NOPASSWD: ALL
```

For a secure environment where `sudo` access is not permitted for all operations, a minimum configuration can be used:

```
tungsten ALL=(root) NOPASSWD: /usr/bin/which, /etc/init.d/mysql
```

When using `xtrabackup`, or additional tools in the Tungsten toolkit, such as `tungsten_provision_slave`, additional commands must be added to the permitted list:

```
tungsten ALL=(root) NOPASSWD: /sbin/service, /usr/bin/innobackupex, /bin/rm, »
/bin/mv, /bin/chown, /bin/chmod, /usr/bin/scp, /bin/tar, /usr/bin/which, »
/etc/init.d/mysql, /usr/bin/test, »
/apps/tungsten/continuent/tungsten/tungsten-replicator/scripts/xtrabackup.sh, »
/apps/tungsten/continuent/tungsten/tools/tpm, /usr/bin/innobackupex-1.5.1, »
/bin/cat, /bin/find
```

Note

On some versions of `sudo`, use of `sudo` is deliberately disabled for `ssh` sessions. To enable support via `ssh`, comment out the requirement for `requiretty`:

```
#Defaults    requiretty
```

B.4. MySQL Database Setup

For replication between MySQL hosts, you must configure each MySQL database server to support the required user names and core MySQL configuration.

Note

Native MySQL replication should not be running when you install Tungsten Replication™. The replication service will be completely handled by Tungsten Replication™, and the normal replication, management and monitoring techniques will not provide you with the information you need.

B.4.1. MySQL Version Support

Database	Version	Support Status	Notes
MySQL	5.0, 5.1, 5.5, 5.6, 5.7	Primary platform	Statement and row based replication is supported. MyISAM and InnoDB table types are fully supported; MyISAM tables may introduce replication errors during failover scenarios. The JSON datatype is not supported.
Percona	5.5, 5.6, 5.7	Primary platform	Statement and row based replication is supported. MyISAM and InnoDB table types are fully supported; MyISAM tables may introduce replication errors during failover scenarios. The JSON datatype is not supported.
MariaDB	5.5, 10.0	Primary platform	

B.4.2. MySQL Configuration

Each MySQL Server should be configured identically within the system. Although binary logging must be enabled on each host, replication should not be configured, since Tungsten Replicator will be handling that process.

The configured `tungsten` user must be able to read the MySQL configuration file (for installation) and the binary logs. Either the `tungsten` user should be a member of the appropriate group (i.e. `mysql`), or the permissions altered accordingly.

Important

Parsing of `mysqld_multi` configuration files is not currently supported. To use a `mysqld_multi` installation, copy the relevant portion of the configuration file to a separate file to be used during installation.

To setup your MySQL servers, you need to do the following:

- Configure your `my.cnf` settings. The following changes should be made to the `[mysqld]` section of your `my.cnf` file:
 - By default, MySQL is configured only to listen on the localhost address (127.0.0.1). The `bind-address` parameter should be checked to ensure that it is either set to a valid value, or commented to allow listening on all available network interfaces:

```
#bind-address = 127.0.0.1
```

- Specify the server id

Each server must have a unique server id:

```
server-id = 1
```

- Ensure that the maximum number of open files matches the configuration of the database user. This was configured earlier at 65535 files.

```
open_files_limit = 65535
```

- Enable binary logs

Tungsten Replicator operates by reading the binary logs on each machine, so logging must be enabled:

```
log-bin = mysql-bin
```

- Set the `sync_binlog` parameter to 1 (one).

Note

In MySQL 5.7, the default value is 1.

The MySQL `sync_binlog` parameter sets the frequency at which the binary log is flushed to disk. A value of zero indicates that the binary log should not be synchronized to disk, which implies that only standard operating system flushing of writes will occur. A value greater than one configures the binary log to be flushed only after `sync_binlog` events have been written. This can introduce a delay into writing information to the binary log, and therefore replication, but also opens the system to potential data loss if the binary log has not been flushed when a fatal system error occurs.

Setting a value of value 1 (one) will synchronize the binary log on disk after each event has been written.

```
sync_binlog = 1
```

- Increase MySQL protocol packet sizes

The replicator can apply statements up to the maximum size of a single transaction, so the maximum allowed protocol packet size must be increase to support this:

```
max_allowed_packet = 52m
```

- Configure InnoDB as the default storage engine

Tungsten Replication 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 Replication.

```
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 Replication™ the value should never be set to 0, otherwise the cluster health may be affected during a failure or failover scenario.

- A value of **1 (one)** provides the best transaction stability by ensuring that all writes to disk are flushed and committed before the transaction is returned as complete. Using this setting implies an increased disk load and so may impact the overall performance.

When using Tungsten Replication™ 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 Replication™ 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 Replication™ 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 re-configuring and restarting MySQL Server. See [Section 8.13, “Performing Database or OS Maintenance”](#).

Optional configuration changes that can be made to your MySQL configuration:

- InnoDB Flush Method

```
innodb_flush_method=O_DIRECT
```

The InnoDB flush method can effect the performance of writes within MySQL and the system as a whole.

`O_DIRECT` is generally recommended as it eliminates double-buffering of InnoDB writes through the OS page cache. Otherwise, MySQL will be contending with Tungsten and other processes for pages there — MySQL is quite active and has a lot of hot pages for indexes and the like this can result lower i/o throughput for other processes.

Tungsten particularly depends on the page cache being stable when using parallel apply. There is one thread that scans forward over the THL pages to coordinate the channels and keep them from getting too far ahead. We then depend on those pages staying in cache for a while so that all the channels can read them — as you are aware parallel apply works like a bunch of parallel table scans that are traveling like a school of sardines over the same part of the THL. If pages get kicked out again before all the channels see them, parallel replication will start to serialize as it has to wait for the OS to read them back in again. If they stay in memory on the other hand, the reads on the THL are in-memory, and fast. For more information on parallel replication, see [Section 7.1, “Deploying Parallel Replication”](#).

- Increase InnoDB log file size

The default InnoDB log file size is 5MB. This should be increased to a larger file size, due to a known issue with `xtrabackup` during backup and restore operations.

To change the file size, read the corresponding information in the MySQL manual for configuring the file size information. See [MySQL 5.1](#), [MySQL 5.5](#), [MySQL 5.6](#), [MySQL 5.7](#).

- Binary Logging Format

Tungsten Replicator works with both statement and row-based logging, and therefore also mixed-based logging. The chosen format is entirely up to the systems and preferences, and there are no differences or changes required for Tungsten Replicator to operate. For native MySQL to MySQL master/slave replication, either format will work fine.

Depending on the exact use case and deployment, different binary log formats imply different requirements and settings. Certain deployment types and environments require different settings:

- For multi-master deployment, use row-based logging. This will help to avoid data drift where statements make fractional changes to the data in place of explicit updates.
- Use row-based logging for heterogeneous deployments. All deployments to Oracle, MongoDB, Vertica and others rely on row-based logging.
- Use mixed replication if warnings are raised within the MySQL log indicating that statement only is transferring possibly dangerous statements.
- Use statement or mixed replication for transactions that update many rows; this reduces the size of the binary log and improves the performance when the transaction are applied on the slave.

- Use row replication for transactions that have temporary tables. Temporary tables are replicated if statement or mixed based logging is in effect, and use of temporary tables can stop replication as the table is unavailable between transactions. Using row-based logging also prevents these tables entering the binary log, which means they do not clog and delay replication.

The configuration of the MySQL server can be permanently changed to use an explicit replication by modifying the configuration in the configuration file:

```
binlog-format = row
```

Note

In MySQL 5.7, the default format is `ROW`.

For temporary changes during execution of explicit statements, the binlog format can be changed by executing the following statement:

```
mysql> SET binlog-format = ROW;
```

You must restart MySQL after any changes have been made.

- Ensure the `tungsten` user can access the MySQL binary logs by either opening up the directory permissions, or adding the `tungsten` user to the group owner for the directory.

B.4.3. MySQL User Configuration

- Tungsten User Login

It is possible to use users with a lower-privilege level and without as many rights. For more information, see [Section B.4.4, “MySQL Unprivileged Users”](#).

The `tungsten` user connects to the MySQL database and applies the data from the replication stream from other datasources in the dataservice. The user must therefore be able execute any SQL statement on the server, including grants for other users. The user *must* have the following privileges in addition to privileges for creating, updating and deleting DDL and data within the database:

- `SUPER` privilege is required so that the user can perform all administrative operations including setting global variables.
- `GRANT OPTION` privilege is required so that users and grants can be updated.

To create a user with suitable privileges:

```
mysql> CREATE USER tungsten@'%' IDENTIFIED BY 'password';
mysql> GRANT ALL ON *.* TO tungsten@'%' WITH GRANT OPTION;
```

The connection will be made from the host to the local MySQL server. You may also need to create an explicit entry for this connection. For example, on the host `host1`, create the user with an explicit host reference:

```
mysql> CREATE USER tungsten@'host1' IDENTIFIED BY 'password';
mysql> GRANT ALL ON *.* TO tungsten@'host1' WITH GRANT OPTION;
```

The above commands enable logins from any host using the user name/password combination. If you want to limit the configuration to only include the hosts within your cluster you must create and grant individual user/host combinations:

```
mysql> CREATE USER tungsten@'client1' IDENTIFIED BY 'password';
mysql> GRANT ALL ON *.* TO tungsten@'client1' WITH GRANT OPTION;
```

Note

If you later change the cluster configuration and add more hosts, you will need to update this configuration with each new host in the cluster.

B.4.4. MySQL Unprivileged Users

By default, the `tungsten` user needs to be given `SUPER` privileges within MySQL so that the user can apply, create and access all the tables and data within the MySQL database. In some situations, this level of access is not available within the MySQL environment, for example, when using a server that is heavily secured, or Amazon’s RDS service.

For this situation, the Tungsten Replication can be configured to use an ‘unprivileged’ user configuration. This configuration does not require the `SUPER` privilege, but instead needs explicit privileges on the schema created by Tungsten Replication, and on the schemas that it will update when applying events.

The capability can be enabled by using the following two options and behaviors:

- `--privileged-master=false` [360]

When privileged master is disabled:

- A master replicator will not attempt to suppress binlog writes during operations.
- A master replicator Will not issue a `FLUSH LOGS` command when the replicator starts.
- The current replicator position is not updated within the `trep_commit_seqno` table.

The `tungsten` user must that connects to the database must be configured to work with the MySQL service using the following grants:

```
mysql> GRANT ALL ON tungsten_batch.* TO tungsten@'%' IDENTIFIED BY 'secret';
mysql> GRANT SELECT ON *.* TO tungsten@'%' IDENTIFIED BY 'secret';
mysql> GRANT REPLICATION SLAVE ON *.* TO tungsten@'%' IDENTIFIED BY 'secret';
mysql> REVOKE SUPER ON *.* FROM tungsten@'%';
```

- `--privileged-slave=false` [360]

When privileged slave is disabled:

- The current replicator position is not updated within the `trep_commit_seqno` table.

```
mysql> GRANT ALL ON tungsten_batch.* TO tungsten@'%' IDENTIFIED BY 'secret';
mysql> GRANT SELECT,INSERT,UPDATE ON *.* TO tungsten@'%' IDENTIFIED BY 'secret';
mysql> GRANT REPLICATION SLAVE ON *.* TO tungsten@'%' IDENTIFIED BY 'secret';
mysql> REVOKE SUPER ON *.* FROM tungsten@'%';
```

Optionally, `INSERT` and `UPDATE` privileges can be explicitly added to the user permissions for the tables/databases that will be updated during replication.

B.5. Oracle Database Setup

B.5.1. Oracle Version Support

Database	Version	Support Status	Notes
Oracle	10g Release 2 (10.2.0.5), 11g	Primary Platform	Synchronous CDC is supported on Standard Edition only; Synchronous and Asynchronous are supported on Enterprise Editions

B.5.2. Oracle Environment Variables

Ensure the `tungsten` user being used for the master Tungsten Replicator service has the same environment setup as an Oracle database user. The user must have the following environment variables set:

Environment Variable	Sample Directory	Notes
<code>ORACLE_HOME</code>	<code>/home/oracle/app/oracle/product/11.2.0/dbhome_2</code>	The home directory of the Oracle installation.
<code>LD_LIBRARY_PATH</code>	<code>\$ORACLE_HOME/lib</code>	The library directory of the Oracle installation.
<code>ORACLE_SID</code>	<code>orcl</code>	Oracle System ID for this installation.
<code>JAVA_HOME</code>		The home of the Java installation.
<code>PATH</code>	<code>\$ORACLE_HOME/bin:\$JAVA_HOME/bin</code>	Must include the Oracle and Java binary directories.
<code>CLASSPATH</code>	<code>\$ORACLE_HOME/ucp/lib/ucp.jar:\$ORACLE_HOME/jdbc/lib/ojdbc6.jar:\$CLASSPATH</code>	Must include the key Oracle libraries the Oracle JDBC driver.

These should be set within the `.bashrc` or `.profile` to ensure these values are set correctly for all logins.

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
7. Provide steps already taken and commands already run to resolve the issue
8. Have you searched your previous support cases? <https://continuent.zendesk.com>.
9. Have you checked the Continuent documentation? <https://docs.continuent.com>
10. Have you checked our general knowledge base? For our Error/Cause/Solution guidance on specific issues and error messages, and how the reason can be identified and resolved, see [Section C.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
tungsten@host1:/home/tungsten/tungsten-replicator-5.0.1-136
```

```
shell> cd /home/tungsten/tungsten-replicator-5.0.1-136
shell> ./tools/tpm_diag
```

The process will create a file called `tungsten-diag-2014-03-20-10-21-29.zip`, with the corresponding date and time information replaced. This file should be included in the reported support issue as an attachment.

For a staging directory installation, `tpm diag` will collect together all of the information from each of the configured hosts in the cluster. For an INI file based installation, `tpm diag` will connect to all configured hosts if `ssh` is available. If a warning that `ssh` is not available is generated, `tpm diag` must be run individually on each host in the cluster.

- Manually Collecting Logs

If `tpm diag` cannot be used, or fails to return all the information, the information can be collected manually:

1. Run `tpm reverse` on all the hosts in the cluster:

```
shell> tpm reverse
```

2. Collect the logs from each host. Logs are available within the `service_logs` directory. This contains symbolic links to the actual log files. The original files can be included within a `tar` archive by using the `-h` option. For example:

```
shell> cd /opt/continuent
shell> tar zcfh host1-logs.tar.gz ./service_logs
```

The `tpm reverse` and log archives can then be submitted as attachments with the support query.

C.1.4. Open a Support Ticket

Login to the support portal and click on 'Submit a Request' at the top of the screen. You can access this page directly at <https://continuent.zendesk.com/requests/new>.

C.1.5. Open a Support Ticket via Email

Send an email to helpdesk@continuent.com from the email address that you used to create your support account. You can include a description and attachments to help us diagnose the problem.

C.1.6. Getting Updates for all Company Support Tickets

If multiple people in your organization have created support tickets, it is possible to get updates on any support tickets they open. You should see your organization name along the top of the support portal. It will be listed after the Check Your Existing Requests tab.

To see all updates for your organization, click on the organization name and then click the Subscribe link.

If you do not see your organization name listed in the headers, open a support ticket asking us to create the organization and list the people that should be included.

C.1.7. Support Severity Level Definitions

Summary of the support severity levels with initial response targets:

- Urgent: initial response within an hour

Represents a reproducible emergency condition (i.e. a condition that involves either data loss, data corruption, or lack of data availability) that makes the use or continued use of any one or more functions impossible. The condition requires an immediate solution. Continuent guarantees a maximum one (1) hour initial response time. Continuent will continue to work with Customer until Customer's database is back in production. The full resolution and the full root cause analysis will be provided when available.

- High: initial response within four (4) hours

Represents a reproducible, non-emergency condition (i.e. a condition that does not involve either data loss, data corruption or lack of database availability) that makes the use or continued use of any one or more functions difficult, and cannot be circumvented or avoided on a temporary basis by Customer. Continuent guarantees a maximum four (4) hours initial response time.

- Normal: initial response within one (1) business day

Represents a reproducible, limited problem condition that may be circumvented or avoided on a temporary basis by Customer. Continuent guarantees a maximum one (1) business day initial response time.

- Low: no guaranteed initial response interval

Represents minor problem conditions or documentation errors that are easily avoided or circumvented by Customer. Additional request for new feature suggestions, which are defined as new functionality in existing product, are also classified as low severity level. Continuent does not guarantee any particular initial response time, or a commitment to fix in any particular time frame unless Customer engages Continuent for professional services work to create a fix or a new feature.

C.2. Error/Cause/Solution

C.2.1. There were issues configuring the sandbox MySQL server

Last Updated: 2016-04-20

Condition or Error

- The command `tungsten_provision_thl` fails when using Percona Server.
- When running the command `tungsten_provision_thl`, you see the error:

```
There were issues configure the sandbox MySQL server
```

- MySQL Sandbox fails when using Percona Server.
- In the `$CONTINUENT_ROOT/service_logs/provision_thl.log` file, you see entries similar to:

```
mysqld: error while loading shared libraries: libssl.so.6: cannot open shared object file: No such file or directory
```

- In the `$CONTINUENT_ROOT/provision_thl.log` file, you see entries similar to:

```
mysql_install_db Error in my_thread_global_end(): 1 threads didn't exit
```

Causes

- This issue occurs because of a problem in Percona Server tarball distributions.

There are two issues with Percona Server tarball distributions, which depends on the version you have downloaded.

Look in the log file `$CONTINUENT_ROOT/service_logs/provision_thl.log` for:

- `mysqld: error while loading shared libraries: libssl.so.6`
- `mysql_install_db Error in my_thread_global_end()`

Rectifications

- To resolve this issue in Centos, install openssl by running the command:

```
shell> sudo yum install openssl098e
```

Alternatively, use Oracle MySQL or MariaDB which do not experience these issues.

Note

VMware does not endorse or recommend any particular third party utility.

More Information

[Section 9.18, "The `tungsten_provision_thl` Command"](#)

C.2.2. 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 Replication 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.2.3. 'subscription exists' when setting up CDC on Oracle

Last Updated: 2014-04-24

Condition or Error

When running `setupCDC.sh` an error regarding an existing subscription is received and `setupCDC.sh` fails to complete.

Causes

- This error indicates that `setupCDC.sh` has previously been executed and the CDC configuration and publishers and subscribers have been created, but the names of the tables or users may have been changed.

Rectifications

- The `cleanup_cdc_tables.sql` file can be used to cleanup an existing CDC installation. See [Section 5.7.7, "CDC Cleanup and Correction"](#).

C.2.4. 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.2.5. The session variable SQL_MODE when set to include ALLOW_INVALID_DATES does not apply statements correctly on the slave.

Last Updated: 2013-07-17

Condition or Error

Replication fails due to an incorrect SQL mode, `INVALID_DATES` being applied for a specific transaction.

Causes

- Due to a problem with the code, the `SQL_MODE` variable in MySQL when set to include `ALLOW_INVALID_DATES` would be identified incorrectly as `INVALID_DATES` from the information in the binary log.

Rectifications

- In affected versions, these statements can be bypassed by explicitly ignoring that value in the event by editing `tungsten-replicator/conf/replicator.properties` to include the following property line:

```
replicator.applier.dbms.ignoreSessionVars=autocommit|INVALID_DATES
```

C.2.6. 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 Replication user has too many open files, processes, or both.

Causes

- User limits for processes or files have either been exhausted, or recommended limits for user configuration have not been set.

Rectifications

- Check the output of `ulimit` and check the configure file and process limits:

```
shell> ulimit -a
core file size (blocks, -c) 0
data seg size (kbytes, -d) unlimited
file size (blocks, -f) unlimited
max locked memory (kbytes, -l) unlimited
max memory size (kbytes, -m) unlimited
open files (-n) 256
pipe size (512 bytes, -p) 1
stack size (kbytes, -s) 8192
cpu time (seconds, -t) unlimited
max user processes (-u) 709
virtual memory (kbytes, -v) unlimited
```

If the figures reported are less than the recommended settings, see [Section B.3.1, “Creating the User Environment”](#) for guidance on how these values should be changed.

More Information

[Section B.3.1, “Creating the User Environment”](#)

C.2.7. Replicator runs out of memory

Last Updated: 2016-05-18

Condition or Error

The replicator runs out of memory, triggers a stack trace indicator a memory condition, or the replicator fails to extract the transaction information from the MySQL binary log.

Causes

- The replicator operates by extracting (or applying) an entire transaction. This means that when extracting data from the binary log, and writing that to THL, or extracting from the THL in preparation for applying to the target, the entire transaction, or an entire statement within a multi-statement transaction, must be held in memory.

In the event of a very large transaction having to be extracted, this can cause a problem with the memory configuration. The actual configuration of how much memory is used is determined through a combination of the number of fragments, the size of the internal buffer used to store those fragments, and the overall fragment size.

Rectifications

- Although you can increase the overall memory allocated to the replicator, changing the internal sizes used can also improve the performance and ability to extract data.

First, try reducing the size of the buffer (`replicator.global.buffer.size`) used to hold the transaction fragments. The default for this value is 10, but reducing this to 5 or less will ease the required memory:

```
replicator.global.buffer.size=10
```

Altering the size of each fragment can also help, as it reduces the memory required to hold the data before it is written to disk and sent out over the network to slave replicators. Reducing the fragment size will reduce the memory footprint. The size is controlled by the `replicator.extractor.dbms.transaction_frag_size` parameter:

```
replicator.extractor.dbms.transaction_frag_size=1000000
```

Note that if you change the fragment size, you may need to reset the service on the extractor so that the binary log is parsed again. You can reset the service by using the `trepctl reset` command.

C.2.8. MySQLExtractException: unknown data type 0

Last Updated: 2014-04-15

Condition or Error

Replication fails to extract the data from the MySQL binary log, and the replicator will not go online again.

Causes

- The format of `DECIMAL` types between MySQL 4.x and MySQL 5.0 changed, however, the datatype was not automatically modified during an upgrade process. This means that tables that were created in MySQL 4.x and now exist within MySQL 5.0 using the `DECIMAL` generate an incompatible entry within the MySQL binary log. The upgrade and `mysql_upgrade` commands do not update the tables correctly. More detailed information on the change and issue can be located in [Bug #57166](#).

Rectifications

- The table definition must be manually upgraded to force the change of the columns using the older `DECIMAL` type. The recommended correction is to explicitly upgrade the `DECIMAL` columns. For example:

```
mysql> ALTER TABLE faulty MODIFY COLUMN faulty_column DECIMAL;
```

This should be performed on the master within your topology. To correct the error, you must use `tpm reset-thl` to regenerate the THL.

C.2.9. Services requires a reset

Last Updated: 2016-05-18

Condition or Error

The replicator service needs to be reset, for example if your MySQL service has been reconfigured, or when resetting a data warehouse or batch loading service after a significant change to the configuration.

Causes

- If the replicator stops replicating effectively, or the configuration and/or schema of a source or target in a datawarehouse loading solution has changed significantly. This will reset the service, starting extraction from the current point, and the target/slave from the new master position. It will also reset all the positions for reading and writing.

Rectifications

- To reset a service entirely, without having to perform a re-installation, you should follow these steps. This will reset both the THL, source database binary log reading position and the target THL and starting point.

- Take the slave offline:

```
slave-shell> trepctl offline
```

- Take the master offline:

```
slave-shell> trepctl offline
```

- Use `trepctl` to reset the service on the master and slave. You must use the service name explicitly on the command-line:

```
master-shell> trepctl -service alpha reset -y
slave-shell> trepctl -service alpha reset -y
```

- Put the slave online:

```
slave-shell> trepctl offline
```

- Put the master online:

```
slave-shell> trepctl offline
```

C.2.10. OptimizeUpdatesFilter cannot filter, because column and key count is different. Make sure that it is defined before filters which remove keys (eg. PrimaryKeyFilter)

Last Updated: 2014-07-28

Condition or Error

When using the `optimizeupdates` filter, replication stops with the error message in the output from `trepctl status` or when examining the log file.

Causes

- The `optimizeupdates` filter works by removing indexed columns from updates that are unnecessary when a primary key exists to locate the record. If the key information has already been removed (for example, by the `pkey` filter, then the columns cannot be effectively compared and optimized.

Rectifications

- If the `pkey` filter is required, change the order of the filters within the specified stage within the replicator so that the `optimizeupdates` filter is called *before* the `pkey` filter.

More Information

[Section 11.4.29, "PrimaryKey Filter"](#)

C.2.11. 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 Replication 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 Replication.

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.3. Known Issues

C.3.1. Triggers

Tungsten Replicator does not automatically shut off triggers on slaves. This creates problems on slaves when using row-based replication (RBR) as the trigger will run twice. Tungsten cannot do this because the setting required to do so is not available to MySQL client applications. Typical symptoms are duplicate key errors, though other problems may appear. Consider the following fixes:

- Drop triggers on slaves. This is practical in fan-in for reporting or other cases where you do not need to failover to the slave at a later time.
- Create an `is_master()` function that triggers can use to decide whether they are on the master or slave.
- Use statement replication. Beware, however, that even in this case you may find problems with triggers and auto-increment keys.

The `is_master()` approach is simple to implement. First, create a function like the following that returns 1 if we are using the Tungsten user, as would be the case on a slave.

```
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.

```
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.

C.4. Troubleshooting Timeouts

C.5. Troubleshooting Backups

- Operating system command failed

Backup directory does not exist.

```
...
INFO | jvm 1 | 2013/05/21 09:36:47 | Process timed out: false
INFO | jvm 1 | 2013/05/21 09:36:47 | Process exception null
INFO | jvm 1 | 2013/05/21 09:36:47 | Process stderr: Error: »
    The directory '/opt/continuent/backups/xtrabackup' is not writeable
...
```

- Backup Retention

C.6. Running Out of Diskspace

```
...
pendingError          : Event application failed: seqno=156847 »
  fragno=0 message=Unable to store event: seqno=156847
pendingErrorCode      : NONE
pendingErrorEventId   : mysql-bin.000025:0000000024735754;0
pendingErrorSeqno     : 156847
pendingExceptionMessage: Unable to store event: seqno=156847
...
```

The above indicates that the THL information could not be stored on disk. To recover from this error, make space available on the disk, or move the THL files to a different device with more space, then set the replicator service online again.

For more information on moving THL files to a different disk, see [Section D.1.5.3, “Moving the THL File Location”](#); for information on moving the backup file location, see [Section D.1.1.4, “Relocating Backup Storage”](#).

C.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.3.2.2, “SSH Configuration”](#) instructions.

- The most likely representation of this error will be when executing `tpm` during a deployment:

```
Error:
#####
Validation failed
#####
#####
Errors for host1
```

```
#####
ERROR>>host1>>Unable to SSH to host1 as root. (SSHLoginCheck)
Ensure that the host is running and that you can login as root via SSH using key authentication
tungsten-configure.log shows:
2012-05-23T11:10:37+02:00 DEBUG>>Execute `whoami` on host1 as root
2012-05-23T11:10:38+02:00 DEBUG>>RC: 0, Result: stdin: is not a tty
```

Try running the following command:

```
shell> ssh tungsten@host1 sudo whoami
```

If the SSH and `sudo` configurations have been configured correctly, it should return `root`. Any other value indicates a failure to configure the prerequisites properly.

- Check that none of the profile scripts (`.profile`, `.bash_profile`, `.bashrc`, etc.) do not contain a call to `mesg n`. This may fool the non-interactive `ssh` call; the call to this command should be changed to only be executed on interactive shells:

```
if `tty -s`; then
    mesg n
fi
```

- Check that firewalls and/or antivirus software are not blocking or preventing connectivity on port 22.

If `ssh` has been enabled on a non-standard port, use the `--net-ssh-option=port` option to specify the alternative port.

- Make sure that the user specified in the `--user` [370] to tpm is allowed to connect to your cluster nodes.

C.8. Troubleshooting Data Differences

It can sometimes become necessary to identify table and data differences due to unexpected behaviour or failures. There are a number of third party tools that can help identify and fix however a lot of them assume native replication is in place, the following explains the recommended methods for troubleshooting a Tungsten Environment based on MySQL as the source and target technologies.

C.8.1. Identify Structural Differences

If you suspect that there are differences to a table structure, a simple method to resolve this will be to compare schema DDL.

Extract DDL on the Master node, specifying the schema in place of {DB}:

```
shell> mysqldump -u root -p --no-data -h localhost --databases {DB} >master.sql
```

Repeat the same on the Slave node:

```
shell> mysqldump -u root -p --no-data -h localhost --databases {DB} >slave.sql
```

Now, using diff, you can compare the results

```
shell> diff master.sql slave.sql
```

Using the output of diff, you can then craft the necessary SQL statements to re-align your structure

C.8.2. Identify Data Differences

It is possible to use pt-table-checksum from the Percona Toolkit to identify data differences, providing you use the syntax described below for bypassing the native replication checks. First of all, it is advisable to familiarise yourself with the product by reading through the providers own documentation here:

<https://www.percona.com/doc/percona-toolkit/2.2/pt-table-checksum.html>

Once you are ready, ensure you install the latest version to the persona toolkit on all nodes, next execute the following on the Master node:

```
shell> pt-table-checksum --set-vars innodb_lock_wait_timeout=500 \
--recursion-method=none \
--ignore-databases=mysql \
--ignore-databases-regex=tungsten* \
h=localhost,u=tungsten,p=secret
```

On first run, this will create a database called percona, and within that database a table called checksums. The process will gather checksum information on every table in every database excluding the mysql and tungsten related schemas. You can now execute the following SQL Statement on the slave to identify tables with data differences:

```
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
/*percona-toolkit src_db:employees src_tbl:departments src_dsn:P=13306,h=db1,p=...,u=tungsten
dst_db:employees dst_tbl:departments dst_dsn:P=13306,h=db2,p=...,u=tungsten
lock:0 transaction:1 changing_src:0 replicate:0 bidirectional:0 pid:24524 user:tungsten host:db1*/;
```

The UPDATE staments 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> <userinput>SELECT db, tbl, SUM(this_cnt) AS total_rows, COUNT(*) AS chunks \
    FROM percona.checksums WHERE ( master_cnt <> this_cnt OR master_crc \
    <> this_crc OR ISNULL(master_crc) <> ISNULL(this_crc)) GROUP BY db, tbl;</userinput>
```

Any differences will be reported and will need to manually corrected.

C.10. Troubleshooting Memory Usage

Appendix D. Files, Directories, and Environment

D.1. The Tungsten Replication Install Directory

Any Tungsten Replication™ installation creates an installation directory that contains the software and the additional directories where active information, such as the transaction history log and backup data is stored. A sample of the directory is shown below, and a description of the individual directories is provided in [Table D.1, “Continuent Tungsten Directory Structure”](#).

```
shell> ls -al /opt/continuent
total 40
drwxr-xr-x 9 tungsten root    4096 Mar 21 18:47 .
drwxr-xr-x 3 root      root    4096 Mar 21 18:00 ..
drwxrwxr-x 2 tungsten tungsten 4096 Mar 21 18:44 backups
drwxrwxr-x 2 tungsten tungsten 4096 Mar 21 18:47 conf
drwxrwxr-x 3 tungsten tungsten 4096 Mar 21 18:44 relay
drwxrwxr-x 4 tungsten tungsten 4096 Mar 21 18:47 releases
drwxrwxr-x 2 tungsten tungsten 4096 Mar 21 18:47 service_logs
drwxrwxr-x 2 tungsten tungsten 4096 Mar 21 18:47 share
drwxrwxr-x 3 tungsten tungsten 4096 Mar 21 18:44 thl
lrwxrwxrwx 1 tungsten tungsten   62 Mar 21 18:47 tungsten -> /opt/continuent/releases/tungsten-replicator-5.0.1-136_pid31409
```

The directories shown in the table are relative to the installation directory, the recommended location is `/opt/continuent`. For example, the THL files would be located in `/opt/continuent/thl`.

Table D.1. Continuent Tungsten Directory Structure

Directory	Description
<code>backups</code>	Default directory for backup file storage
<code>conf</code>	Configuration directory with a copy of the current and past configurations
<code>relay</code>	Location for relay logs if relay logs have been enabled.
<code>releases</code>	Contains one or more active installations of the Continuent Tungsten software, referenced according to the version number and active process ID.
<code>service-logs</code>	Logging information for the active installation
<code>share</code>	Active installation information, including the active JAR for the MySQL connection
<code>thl</code>	The Transaction History Log files, stored in a directory named after each active service.
<code>tungsten</code>	Symbolic link to the currently active release in <code>releases</code> .

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 Replication™. The backup data and metadata for each backup will be stored in this directory.

An example of the directory content is shown below:

```
shell> ls -al /opt/continuent/backups/
total 130788
drwxrwxr-x 2 tungsten tungsten    4096 Apr  4 16:09 .
drwxrwxr-x 3 tungsten tungsten    4096 Apr  4 11:51 ..
-rw-r--r-- 1 tungsten tungsten     71 Apr  4 16:09 storage.index
-rw-r--r-- 1 tungsten tungsten 133907646 Apr  4 16:09 store-000000001-mysqldump_2013-04-04_16-08_42.sql.gz
-rw-r--r-- 1 tungsten tungsten    317 Apr  4 16:09 store-000000001.properties
```

The `storage.index` contains the backup file index information. The actual backup data is stored in the GZipped file. The properties of the backup file, including the tool used to create the backup, and the checksum information, are located in the corresponding `.properties` file. Note that each backup and property file is uniquely numbered so that you can identify and restore a specific backup.

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` [330] 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 back-

ups 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 `--rep1-backup-directory` [330].

The backup retention will only remove files starting with `store`. If you are using a backup method that creates additional information then those files may not be fully removed until the next backup process begins. This includes `xtrabackup-full`, `xtrabackup-incremental` and any snapshot based backup methods. You may manually clean these excess files if space is needed before the next backup method. If you delete information associated with an existing backup, any attempts to restore it will fail.

D.1.1.2. Manually Deleting Backup Files

If you no longer need one or more backup files, you can delete the files from the filesystem. You must delete both the SQL data, and the corresponding properties file. For example, from the following directory:

```
shell> ls -al /opt/continuent/backups
total 764708
drwxrwxr-x 2 tungsten tungsten 4096 Apr 16 13:57 .
drwxrwxr-x 3 tungsten tungsten 4096 Apr 16 13:54 ..
-rw-r--r-- 1 tungsten tungsten 71 Apr 16 13:56 storage.index
-rw-r--r-- 1 tungsten tungsten 517170 Apr 15 18:02 store-000000004-mysqldump-1332463738918435527.sql
-rw-r--r-- 1 tungsten tungsten 311 Apr 15 18:02 store-000000004.properties
-rw-r--r-- 1 tungsten tungsten 517170 Apr 15 18:06 store-000000005-mysqldump-2284057977980000458.sql
-rw-r--r-- 1 tungsten tungsten 310 Apr 15 18:06 store-000000005.properties
-rw-r--r-- 1 tungsten tungsten 781991444 Apr 16 13:57 store-000000006-mysqldump-3081853249977885370.sql
-rw-r--r-- 1 tungsten tungsten 314 Apr 16 13:57 store-000000006.properties
```

To delete the backup files for index 4:

```
shell> rm /opt/continuent/backups/alpha/store-000000004*
```

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 be copied to another directory or system using any suitable means. Once the backup has been completed, the files will not be modified or updated and are therefore safe to be moved or actively copied to another location without fear of corruption of the backup information.

There are multiple files associated with each backup. The number of files will depend on the backup method that was used. All backups will use at least two files in the `/opt/continuent/backups` directory.

```
shell> cd /opt/continuent/backups
shell> scp store-[0]*6[\.-]* host3:$PWD/
store-000000001-full_xtrabackup_2014-08-16_15-44_86
store-000000001.properties
100%   70      0.1KB/s  00:00
100%  314      0.3KB/s  00:00
```

Note

Check the ownership of files if you have trouble transferring files or restoring the backup. They should be owned by the Tungsten system user to ensure proper operation.

If `xtrabackup-full` method was used, you must transfer the corresponding directory from `/opt/continuent/backups/xtrabackup`. In this example that would be `/opt/continuent/backups/xtrabackup/full_xtrabackup_2014-08-16_15-44_86`.

```
shell> cd /opt/continuent/backups/xtrabackup
shell> rsync -aze ssh full_xtrabackup_2014-08-16_15-44_86 host3:$PWD/
```

If the `xtrabackup-incremental` method was used, you must transfer multiple directories. In addition to the corresponding directory from `/opt/continuent/backups/xtrabackup` you must transfer all `xtrabackup-incremental` directories since the most recent `xtrabackup-full` backup and then transfer that `xtrabackup-full` directory. See the example below for further explanation :

```
shell> ls -altr /opt/continuent/backups/xtrabackup/
total 32
drwxr-xr-x 7 tungsten tungsten 4096 Oct 16 20:55 incr_xtrabackup_2014-10-16_20-55_73
drwxr-xr-x 7 tungsten tungsten 4096 Oct 17 20:55 full_xtrabackup_2014-10-17_20-55_1
drwxr-xr-x 7 tungsten tungsten 4096 Oct 18 20:55 incr_xtrabackup_2014-10-18_20-55_38
drwxr-xr-x 7 tungsten tungsten 4096 Oct 19 20:57 incr_xtrabackup_2014-10-19_20-57_76
```

```
drwxr-xr-x 7 tungsten tungsten 4096 Oct 20 20:58 full_xtrabackup_2014-10-20_20-57_41
drwxr-xr-x 8 tungsten tungsten 4096 Oct 21 20:58 .
drwxr-xr-x 7 tungsten tungsten 4096 Oct 21 20:58 incr_xtrabackup_2014-10-21_20-58_97
drwxrwxr-x 3 tungsten tungsten 4096 Oct 21 20:58 ..
```

In this example there are two instances of **xtrabackup-full** backups and four **xtrabackup-incremental** backups.

- To restore either of the **xtrabackup-full** backups then they would be copied to the target host on their own.
- To restore **incr_xtrabackup_2014-10-21_20-58_97**, it must be copied along with **full_xtrabackup_2014-10-20_20-57_41**.
- To restore **incr_xtrabackup_2014-10-19_20-57_76**, it must be copied along with **incr_xtrabackup_2014-10-18_20-55_38** and **full_xtrabackup_2014-10-17_20-55_1**.

D.1.1.4. Relocating Backup Storage

If the filesystem on which the main installation directory is running out of space and you need to increase the space available for backup files without interrupting the service, you can use symbolic links to relocate the backup information.

Note

When using an NFS mount point when backing up with **xtrabackup**, the command must have the necessary access rights and permissions to change the ownership of files within the mounted directory. Failure to update the permissions and ownership will cause the **xtrabackup** command to fail. The following settings should be made on the directory:

- Ensure the **no_root_squash** option on the NFS export is not set.
- Change the group and owner of the mount point to the **tungsten** user and **mysql** group:

```
shell> chown tungsten /mnt/backups
shell> chgrp mysql /mnt/backups
```

Owner and group IDs on NFS directories must match across all the hosts using the NFS mount point. Inconsistencies in the owner and group IDs may lead to backup failures.

- Change the permissions to permit at least owner and group modifications::

```
shell> chmod 770 /mnt/backups
```

- Mount the directory:

```
shell> mount host1:/exports/backups /mnt/backups
```

The backup directory can be changed using two different methods:

- [Section D.1.1.4.1, "Relocating Backup Storage using Symbolic Links"](#)
- [Section D.1.1.4.2, "Relocating Backup Storage using Configuration Changes"](#)

D.1.1.4.1. Relocating Backup Storage using Symbolic Links

To relocate the backup directory using symbolic links:

1. Ensure that no active backup is taking place of the current host. Your service does not need to be offline to complete this operation.
2. Create a new directory, or attach a new filesystem and location on which the backups will be located. You can use a directory on another filesystem or connect to a SAN, NFS or other filesystem where the new directory will be located. For example:

```
shell> mkdir /mnt/backupdata/continuent
```

3. *Optional*

Copy the existing backup directory to the new directory location. For example:

```
shell> rsync -r /opt/continuent/backups/* /mnt/backupdata/continuent/
```

4. Move the existing directory to a temporary location:

```
shell> mv /opt/continuent/backups /opt/continuent/old-backups
```

5. Create a symbolic link from the new directory to the original directory location:

```
shell> ln -s /mnt/backupdata/continuent /opt/continuent/backups
```

The backup directory has now been moved. If you want to verify that the new backup directory is working, you can optionally run a backup and ensure that the backup process completes correctly.

D.1.1.4.2. Relocating Backup Storage using Configuration Changes

To relocate the backup directory by reconfiguration:

1. Ensure that no active backup is taking place of the current host. Your service does not need to be offline to complete this operation.
2. Create a new directory, or attach a new filesystem and location on which the backups will be located. You can use a directory on another filesystem or connect to a SAN, NFS or other filesystem where the new directory will be located. For example:

```
shell> mkdir /mnt/backupdata/continuent
```

3. *Optional*

Copy the existing backup directory to the new directory location. For example:

```
shell> rsync -r /opt/continuent/backups/* /mnt/backupdata/continuent/
```

4. Following the directions for [tpm update](#) to apply the `--backup-directory=/mnt/backupdata/continuent` [330] setting.

The backup directory has now been moved. If you want to verify that the new backup directory is working, you can optionally run a backup and ensure that the backup process completes correctly.

D.1.2. The `releases` Directory

The `releases` directory contains a copy of each installed release. As new versions are installed and updated (through [tpm update](#)), a new directory is created with the corresponding version of the software.

For example, a number of releases are listed below:

```
shell> ll /opt/continuent/releases/
total 20
drwxr-xr-x  5 tungsten mysql 4096 May 23 16:19 .
drwxr-xr-x  9 tungsten mysql 4096 May 23 16:19 ../
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 tungsten-replicator-5.0.1-136_pid16184/
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 tungsten-replicator-5.0.1-136_pid14577/
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 tungsten-replicator-5.0.1-136_pid23747/
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 tungsten-replicator-5.0.1-136_pid24978/
```

The latest release currently in use can be determined by checking the symbolic link, `tungsten` within the installation directory. For example:

```
shell> ll /opt/continuent
total 40
drwxr-xr-x  9 tungsten mysql 4096 May 23 16:19 .
drwxr-xr-x  3 root      root  4096 Apr 29 16:09 ../
drwxr-xr-x  2 tungsten mysql 4096 May 30 13:27 backups/
drwxr-xr-x  2 tungsten mysql 4096 May 23 16:19 conf/
drwxr-xr-x  3 tungsten mysql 4096 May 10 19:09 relay/
drwxr-xr-x  5 tungsten mysql 4096 May 23 16:19 releases/
drwxr-xr-x  2 tungsten mysql 4096 May 10 19:09 service_logs/
drwxr-xr-x  2 tungsten mysql 4096 May 23 16:18 share/
drwxr-xr-x  3 tungsten mysql 4096 May 10 19:09 th1/
lrwxrwxrwx  1 tungsten mysql   63 May 23 16:19 tungsten -> /opt/continuent/releases/tungsten-replicator-5.0.1-136_pid24978/
```

If multiple services are running on the host, search for `.pid` files within the installation directory to determine which release directories are currently in use by an active service:

```
shell> find /opt/continuent -name "*.pid"
/opt/continuent/releases/tungsten-replicator-5.0.1-136_pid24978/tungsten-replicator/var/treplicator.pid
/opt/continuent/releases/tungsten-replicator-5.0.1-136_pid24978/tungsten-connector/var/tconnector.pid
/opt/continuent/releases/tungsten-replicator-5.0.1-136_pid24978/tungsten-manager/var/tmanager.pid
```

Directories within the `releases` directory that are no longer being used can be safely removed.

D.1.3. The `service_logs` Directory

The `service_logs` directory contains links to the log files for the currently active release. The directory contains the following links:

- `trepsvc.log` — a link to the Tungsten Replicator log.

D.1.4. The `share` Directory

The `share` directory contains information that is shared among all installed releases and instances of Tungsten Replication. Unlike other directories, the `share` directory is not overwritten or replaced during installation or update using `tpm`. This means that the directory can be used to hold information, such as filter configurations, without the contents being removed when the installation is updated.

D.1.5. The `thl` Directory

The transaction history log (THL) retains a copy of the SQL statements from each master host, and it is the information within the THL that is transferred between hosts and applied to the database. The THL information is written to disk and stored in the `thl` directory:

```
shell> ls -al /opt/continuent/thl/alpha/
total 2291984
drwxrwxr-x 2 tungsten tungsten 4096 Apr 16 13:44 .
drwxrwxr-x 3 tungsten tungsten 4096 Apr 15 15:53 ..
-rw-r--r-- 1 tungsten tungsten 0 Apr 15 15:53 disklog.lck
-rw-r--r-- 1 tungsten tungsten 100137585 Apr 15 18:13 thl.data.0000000001
-rw-r--r-- 1 tungsten tungsten 100134069 Apr 15 18:18 thl.data.0000000002
-rw-r--r-- 1 tungsten tungsten 100859685 Apr 15 18:26 thl.data.0000000003
-rw-r--r-- 1 tungsten tungsten 100515215 Apr 15 18:28 thl.data.0000000004
-rw-r--r-- 1 tungsten tungsten 100180770 Apr 15 18:31 thl.data.0000000005
-rw-r--r-- 1 tungsten tungsten 100453094 Apr 15 18:34 thl.data.0000000006
-rw-r--r-- 1 tungsten tungsten 100379260 Apr 15 18:35 thl.data.0000000007
-rw-r--r-- 1 tungsten tungsten 100294561 Apr 16 12:21 thl.data.0000000008
-rw-r--r-- 1 tungsten tungsten 100133258 Apr 16 12:24 thl.data.0000000009
-rw-r--r-- 1 tungsten tungsten 100293278 Apr 16 12:32 thl.data.0000000010
-rw-r--r-- 1 tungsten tungsten 100819317 Apr 16 12:34 thl.data.0000000011
-rw-r--r-- 1 tungsten tungsten 100250972 Apr 16 12:35 thl.data.0000000012
-rw-r--r-- 1 tungsten tungsten 100337285 Apr 16 12:37 thl.data.0000000013
-rw-r--r-- 1 tungsten tungsten 100535387 Apr 16 12:38 thl.data.0000000014
-rw-r--r-- 1 tungsten tungsten 100378358 Apr 16 12:40 thl.data.0000000015
-rw-r--r-- 1 tungsten tungsten 100198421 Apr 16 13:32 thl.data.0000000016
-rw-r--r-- 1 tungsten tungsten 100136955 Apr 16 13:34 thl.data.0000000017
-rw-r--r-- 1 tungsten tungsten 100490927 Apr 16 13:41 thl.data.0000000018
-rw-r--r-- 1 tungsten tungsten 100684346 Apr 16 13:41 thl.data.0000000019
-rw-r--r-- 1 tungsten tungsten 100225119 Apr 16 13:42 thl.data.0000000020
-rw-r--r-- 1 tungsten tungsten 100390819 Apr 16 13:43 thl.data.0000000021
-rw-r--r-- 1 tungsten tungsten 100418115 Apr 16 13:43 thl.data.0000000022
-rw-r--r-- 1 tungsten tungsten 100388812 Apr 16 13:44 thl.data.0000000023
-rw-r--r-- 1 tungsten tungsten 38275509 Apr 16 13:47 thl.data.0000000024
```

THL files are created on both the master and slaves within the cluster. THL data can be examined using the `thl` command.

The THL is written into individual files, which are by default, no more than 1 GByte in size each. From the listing above, you can see that each file has a unique file index number. A new file is created when the file size limit is reached, and given the next THL log file number. To determine the sequence number that is stored within log, use the `thl` command:

```
shell> thl index
LogIndexEntry thl.data.0000000001(0:106)
LogIndexEntry thl.data.0000000002(107:203)
LogIndexEntry thl.data.0000000003(204:367)
LogIndexEntry thl.data.0000000004(368:464)
LogIndexEntry thl.data.0000000005(465:561)
LogIndexEntry thl.data.0000000006(562:658)
LogIndexEntry thl.data.0000000007(659:755)
LogIndexEntry thl.data.0000000008(756:1251)
LogIndexEntry thl.data.0000000009(1252:1348)
LogIndexEntry thl.data.0000000010(1349:1511)
LogIndexEntry thl.data.0000000011(1512:1609)
LogIndexEntry thl.data.0000000012(1610:1706)
LogIndexEntry thl.data.0000000013(1707:1803)
LogIndexEntry thl.data.0000000014(1804:1900)
LogIndexEntry thl.data.0000000015(1901:1997)
LogIndexEntry thl.data.0000000016(1998:2493)
LogIndexEntry thl.data.0000000017(2494:2590)
LogIndexEntry thl.data.0000000018(2591:2754)
LogIndexEntry thl.data.0000000019(2755:2851)
LogIndexEntry thl.data.0000000020(2852:2948)
LogIndexEntry thl.data.0000000021(2949:3045)
LogIndexEntry thl.data.0000000022(3046:3142)
LogIndexEntry thl.data.0000000023(3143:3239)
LogIndexEntry thl.data.0000000024(3240:3672)
```

The THL files are retained for seven days by default, although this parameter is configurable. Due to the nature and potential size required to store the information for the THL, you should monitor the disk space and usage.

The purge is continuous and is based on the date the log file was written. Each time the replicator finishes the current THL log file, it checks for files that have exceeded the defined retention configuration and spawns a job within the replicator to delete files older than the retention policy. Old files are only removed when the current THL log file rotates.

D.1.5.1. Purging THL Log Information on a 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. Put the replication service offline using `trepctl`:

```
shell> trepctl -service alpha offline
```

3. Use the `thl` command to purge the logs up to the specified transaction sequence number. You will be prompted to confirm the operation:

```
shell> thl purge --high 3670
WARNING: The purge command will break replication if you delete all events or >
         delete events that have not reached all slaves.
Are you sure you wish to delete these events [y/N]?
Y
Deleting events where SEQ# <=3670
2013-04-16 14:09:42,384 [- main] INFO thl.THLManagerCtrl Transactions deleted
```

4. Put the replication service online using `trepctl`:

```
shell> trepctl -service alpha online
```

You can now check the current THL file information:

```
shell> thl index
LogIndexEntry thl.data.0000000024(3240:3672)
```

For more information on purging events using `thl`, see [Section 9.15.3, “thl purge Command”](#).

D.1.5.2. Purging THL Log Information on a Master

Warning

Purging the THL on a Master node can potentially remove information that has not yet been applied to the slave databases. Please check and ensure that the THL data that you are purging has been applied to the database on all slaves before continuing.

Important

If the situation allows, it may be better to switch the Master role to a current, up-to-date slave, then perform the steps to purge THL from a slave on the old master host using [Section D.1.5.1, “Purging THL Log Information on a Slave”](#).

Warning

Follow the below steps with great caution! Failure to follow best practices will result in slaves unable to apply transactions, forcing a full re-provisioning. For those steps, please see [Section 8.5, “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:

- Determine the highest sequence number from the THL that you want to delete. To purge the logs up until the latest sequence number, you can use `trepctl` to determine the highest applied sequence number:

```
shell> trepctl services
Processing services command...
NAME          VALUE
----          -----
appliedLastSeqno: 3675
appliedLatency : 0.835
role          : master
serviceName   : alpha
serviceType   : local
started       : true
state         : ONLINE
Finished services command...
```

- Put the replication service offline using `trepctl`:

```
shell> trepctl -service alpha offline
```

- Use the `thl` command to purge the logs up to the specified transaction sequence number. You will be prompted to confirm the operation:

```
shell> thl purge -high 3670
WARNING: The purge command will break replication if you delete all events or >
          delete events that have not reached all slaves.
Are you sure you wish to delete these events [y/N]?
y
Deleting events where SEQ# <=3670
2013-04-16 14:09:42,384 [- main] INFO thl.THLManagerCtrl Transactions deleted
```

- Put the replication service online using `trepctl`:

```
shell> trepctl -service alpha online
```

You can now check the current THL file information:

```
shell> thl index
LogIndexEntry thl.data.0000000024(3240:3672)
```

For more information on purging events using `thl`, see [Section 9.15.3, “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:

- Put the replication service offline using `trepctl`:

```
shell> trepctl -service alpha offline
```

- 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
```

- Copy the existing THL directory to the new directory location. For example:

```
shell> rsync -r /opt/continuent/thl/* /mnt/data/thl/
```

- Move the existing directory to a temporary location:

```
shell> mv /opt/continuent/thl /opt/continuent/old-thl
```

- Create a symbolic link from the new directory to the original directory location:

```
shell> ln -s /mnt/data/thl /opt/continuent/thl
```

- Put the replication service online using `trepctl`:

```
shell> repctl -service alpha online
```

D.1.5.3.2. Relocating THL Storage using Configuration Changes

To permanently change the directory currently holding the THL information can 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:

- Put the replication service offline using `trepctl`:

```
shell> repctl -service alpha offline
```

- 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
```

- Copy the existing THL directory to the new directory location. For example:

```
shell> rsync -r /opt/continuent/thl/* /mnt/data/thl/
```

- Change the directory location using `tpm` to update the configuration for a specific host:

```
shell> tpm update --thl-directory=/mnt/data/thl --host=host1
```

- Put the replication service online using `trepctl`:

```
shell> repctl -service alpha online
```

D.1.5.4. Changing the THL Retention Times

THL files are by default retained for seven days, but the retention period can be adjusted according to 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 Replication. 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` [369] setting. The replication service will be restarted on each host with updated retention configuration.

D.1.6. The `tungsten` Directory

```
shell> ls -l /opt/continuent/tungsten/
total 72
drwxr-xr-x  9 tungsten mysql  4096 May 23 16:18 bristlecone
drwxr-xr-x  6 tungsten mysql  4096 May 23 16:18 cluster-home
drwxr-xr-x  4 tungsten mysql  4096 May 23 16:18 cookbook
-rw-r--r--  1 tungsten mysql   681 May 23 16:18 INSTALL
-rw-r--r--  1 tungsten mysql 19974 May 23 16:18 README.LICENSES
drwxr-xr-x  3 tungsten mysql  4096 May 23 16:18 tools
-rw-r--r--  1 tungsten mysql 19724 May 23 16:18 tungsten.cfg
drwxr-xr-x 11 tungsten mysql  4096 May 23 16:18 tungsten-replicator
```

Table D.2. Continuent Tungsten `tungsten` Sub-Directory Structure

Directory	Description
<code>bristlecone</code>	Contains the bristlecone load-testing tools.

Directory	Description
<code>cluster-home</code>	Home directory for the main tools, configuration and libraries of the Tungsten Replication installation.
<code>cookbook</code>	Cookbook installation and testing tools.
<code>INSTALL</code>	Text file describing the basic installation process for Tungsten Replication
<code>README.LICENSES</code>	Software license information.
<code>tools</code>	Directory containing the tools for installing and configuring Tungsten Replication.
<code>tungsten-replicator</code>	Installed directory of the Tungsten Replicator installation.

D.1.6.1. The `tungsten-replicator` Directory

This directory holds all of the files, libraries, configuration and other information used to support the installation of product.

D.1.6.1.1. The `tungsten-replicator/lib` Directory

This directory holds library files specific to Tungsten Replicator. When performing patches or extending functionality specifically for Tungsten Replicator, for example when adding JDBC libraries for other databases, the JAR files can be placed into this directory.

D.1.6.1.2. The `tungsten-replicator/scripts` Directory

This directory contains scripts used to support Tungsten Replicator operation.

D.2. Log Files

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 10.3, “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 10.3, “tpm Staging Configuration”](#).

- `$CONTINUENT_ROOT`

The `$CONTINUENT_ROOT` variable is created by the `env.sh` file that is created when installing Tungsten Replication. When defined, the variable will contain the installation directory of the corresponding Tungsten Replication installation.

On hosts where multiple installations have been created, the variable can be used to point to different installations.

Appendix E. Terminology Reference

Tungsten Replication 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 Replication.

E.1. Transaction History Log (THL)

The Transaction History Log (THL) stores transactional data from different data servers in a universal format that is then used to exchange and transfer the information between replicator instances. Because the THL is stored and independently managed from the data servers that it reads and writes, the data can be moved, exchanged, and transmuted during processing.

The THL is created by any replicator service acting as a master, where the information is read from the database using the native format, such as the MySQL binary log, or Oracle Change Data Capture (CDC), writing the information to the THL. Once in the THL, the THL data can be exchanged with other processes, including transmission over the network, and then applied to a destination database. Within Tungsten Replicator, this process is handled through the pipeline stages that read and write information between the THL and internal queues.

Information stored in THL is recorded in a series of event records in sequential format. The THL therefore acts as a queue of the transactions. On a replicator reading data from a database, the THL represents the queue of transactions applied on the source database. On a replicator applying that information to a database, the THL represents the list of the transactions to be written. The THL has the following properties:

- THL is a sequential list of events
- THL events are written to a THL file through a single thread (to enforce the sequential nature)
- THL events can be read from individually or sequentially, and multiple threads can read the same THL at the same time
- THL events are immutable; once stored, the contents of the THL are never modified or individually deleted (although entire files may be deleted)
- THL is written to disk without any buffering to prevent software failure causing a problem; the operating system buffers are used.

THL data is stored on disk within the `thl` directory of your Tungsten Replicator installation. The exact location can be configured using `logDir` parameter of the THL component. A sample directory is shown below:

```
total 710504
-rw-r--r-- 1 tungsten tungsten      0 May  2 10:48 disklog.lck
-rw-r--r-- 1 tungsten tungsten 100042900 Jun  4 10:10 thl.data.0000000013
-rw-rw-r-- 1 tungsten tungsten 101025311 Jun  4 11:41 thl.data.0000000014
-rw-rw-r-- 1 tungsten tungsten 100441159 Jun  4 11:43 thl.data.0000000015
-rw-rw-r-- 1 tungsten tungsten 100898492 Jun  4 11:44 thl.data.0000000016
-rw-rw-r-- 1 tungsten tungsten 100305613 Jun  4 11:44 thl.data.0000000017
-rw-rw-r-- 1 tungsten tungsten 100035516 Jun  4 11:44 thl.data.0000000018
-rw-rw-r-- 1 tungsten tungsten 101690969 Jun  4 11:45 thl.data.0000000019
-rw-rw-r-- 1 tungsten tungsten 23086641 Jun  5 21:55 thl.data.0000000020
```

The THL files have the format `thl.data.#####`, and the sequence number increases for each new log file. The size of each log file is controlled by the `--thl-log-file-size` [369] 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` [369] 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 9.15, “The thl Command”](#).

E.1.1. THL Format

The THL is stored on disk in a specific format that combines the information about the SQL and row data, metadata about the environment in which the row changes and SQL changes were made (metadata), and the log specific information, including the source, database, and timestamp of the information.

A sample of the output is shown below, the information is taken from the output of the `thl` command:

```
SEQ# = 0 / FRAG# = 0 (last frag)
- TIME = 2013-03-21 18:47:39.0
- EPOCH# = 0
- EVENTID = mysql-bin.000010:0000000000004397:0
- SOURCEID = host1
- METADATA = [mysql server_id=10;dbms type=mysql;is_metadata=true;service=dsone;»
```

```

shard=tungsten_firstcluster;heartbeat=MASTER_ONLINE]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [##charset = ISO8859_1, autocommit = 1, sql_auto_is_null = 0, »
    foreign_key_checks = 1, unique_checks = 1, sql_mode = '', character_set_client = 8, »
    collation_connection = 8, collation_server = 8]
- SCHEMA = tungsten_dson
- SQL(0) = UPDATE tungsten_dson.heartbeat SET source_tstamp= '2013-03-21 18:47:39', salt= 1, »
    name= 'MASTER_ONLINE' WHERE id= 1 /* __SERVICE__ = [firstcluster] */

```

The sample above shows the information for the SQL executed on a MySQL server. The [EVENTID \[460\]](#) shows the MySQL binary log from which the statement has been read. The MySQL server has stored the information in the binary log using [STATEMENT](#) or [MIXED](#) mode; log events written in [ROW](#) mode store the individual row differences. A summary of the THL stored format information, including both hidden values and the information included in the [thl](#) command output is provided in [Table E.1, “THL Event Format”](#).

Table E.1. THL Event Format

Displayed Field	Internal Name	Data type	Size	Description
-	<code>record_length</code>	Integer	4 bytes	Length of the full record information, including this field
-	<code>record_type</code>	Byte	1 byte	Event record type identifier
-	<code>header_length</code>	Unsigned int	4 bytes	Length of the header information
SEQ# [459]	<code>seqno</code>	Unsigned long	8 bytes	Log sequence number, a sequential value given to each log entry
FRAG# [459]	<code>fragno</code>	Unsigned short	2 bytes	Event fragment number. An event can consist of multiple fragments of SQL or row log data
-	<code>last_frag</code>	Byte	1 byte	Indicates whether the fragment is the last fragment in the sequence
EPOCH# [460]	<code>epoch_number</code>	Unsigned long	8 bytes	Event epoch number. Used to identify log sections within the master THL
SOURCEID [460]	<code>source_id</code>	UTF-8 String	Variable (null terminated)	Event source ID, the hostname or identity of the dataserver that generated the event
EVENTID [460]	<code>event_id</code>	UTF-8 String	Variable (null terminated)	Event ID; in MySQL, for example, the binlog filename and position that contained the original event
SHARDID [461]	<code>shard_id</code>	UTF-8 String	Variable (null terminated)	Shard ID to which the event belongs
TIME [460]	<code>tstamp</code>	Unsigned long	8 bytes	Time of the commit that triggered the event
-	<code>data_length</code>	Unsigned int	4 bytes	Length of the included event data
-	<code>event</code>	Binary	Variable	Serialized Java object containing the SQL or ROW data
METADATA [460]	Part of <code>event</code>	-	-	Metadata about the event
TYPE [460]	Part of <code>event</code>	-	-	Internal storage type of the event
OPTIONS [460]	Part of <code>event</code>	-	-	Options about the event operation
SCHEMA [461]	Part of <code>event</code>	-	-	Schema used in the event
SQL [461]	Part of <code>event</code>	-	-	SQL statement or row data
-	<code>crc_method</code>	Byte	1 byte	Method used to compute the CRC for the event.
-	<code>crc</code>	Unsigned int	4 bytes	CRC of the event record (not including the CRC value)

- [SEQ# \[459\]](#) and [FRAG# \[459\]](#)

Individual events within the log are identified by a sequential [SEQUENCE \[459\]](#) 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# \[460\]](#)

The [EPOCH \[460\]](#) value is used a check to ensure that the logs on the slave and the master match. The [EPOCH \[460\]](#) is stored in the THL, and a new [EPOCH \[460\]](#) is generated each time a master goes online. The [EPOCH \[460\]](#) value is then written and stored in the THL alongside each individual event. The [EPOCH \[460\]](#) acts as an additional check, beyond the sequence number, to validate the information between the slave and the master. The [EPOCH \[460\]](#) 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 \[460\]](#) acts as a check so that when the master rejoins as the slave, the [EPOCH \[460\]](#) 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 \[460\]](#) prevents the slave from accepting events that happen to match only the sequence number, but not the corresponding [EPOCH \[460\]](#).

Each time a Tungsten Replicator master goes online, the [EPOCH \[460\]](#) number is incremented. When the slave connects, it requests the [SEQUENCE \[459\]](#) and [EPOCH \[460\]](#), and the master confirms that the requested [SEQUENCE \[459\]](#) has the requested [EPOCH \[460\]](#). If not, the request is rejected and the slave gets a validation error:

```
pendingExceptionMessage: Client handshake failure: Client response validation failed: »  
  Log epoch numbers do not match: client source ID=west-db2 seqno=408129 »  
    server epoch number=408128 client epoch number=189069
```

When this error occurs, the THL should be examined and compared between the master and slave to determine if there really is a mismatch between the two databases. For more information, see [Section 8.4, "Managing Transaction Failures"](#).

- [SOURCEID \[460\]](#)

The [SOURCEID \[460\]](#) is a string identifying the source of the event stored in the THL. Typically it is the hostname or host identifier.

- [EVENTID \[460\]](#)

The [EVENTID \[460\]](#) is a string identifying the source of the event information in the log. Within a MySQL installed, the [EVENTID \[460\]](#) contains the binary log name and position which provided the original statement or row data.

Note

The event ID shown is the end of the corresponding event stored in the THL, not the beginning. When examining the mysqlbinlog for an sequence ID in the THL, you should check the EVENTID of the previous THL sequence number to determine where to start looking within the binary log.

- [TIME \[460\]](#)

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 \[460\]](#) value as stored in the THL is used to compute latency information when reading and applying data on a slave.

- [METADATA \[460\]](#)

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 \[460\]](#)

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 \[460\]](#)

Part of the [EVENT](#) binary payload, the [OPTIONS](#) [460] 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](#) [460] block for each [SQL](#) [461] statement stored in the event.

- [SCHEMA](#) [461]

Part of the [EVENT](#) structure, the [SCHEMA](#) [461] provides the database or schema name in which the statement or row data was applied.

- [SHARDID](#) [461]

When using parallel apply, provides the generated shard ID for the event when it is applied by the parallel applier thread. data.

- [SQL](#) [461]

For statement based events, the SQL of the statement that was recorded. Multiple individual SQL statements as part of a transaction can be contained within a single event fragment.

For example, the MySQL statement:

```
mysql> INSERT INTO user VALUES (null, 'Charles', now());
Query OK, 1 row affected (0.01 sec)
```

Stores the following into the THL:

```
SEQ# = 3583 / FRAG# = 0 (last frag)
- TIME = 2013-05-27 11:49:45.0
- EPOCH# = 2500
- EVENTID = mysql-bin.000007:000000625753960:0
- 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](#) [461], [TABLE](#) [461] and individual ROW data. For each ROW, there may be one or more [COL](#) [461] (column) and identifying [KEY](#) [461] event to identify the row on which the action is to be performed.

The same statement when recorded in [ROW](#) [461] format:

```
SEQ# = 3582 / FRAG# = 0 (last frag)
- TIME = 2013-05-27 11:45:19.0
- EPOCH# = 2500
- EVENTID = mysql-bin.000007:000000625753710:0
- 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 Replication 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. For example, when extracting from MySQL, the output from `treptl` shows the MySQL binary log file and position within the log where the transaction was extracted:

```
shell> treptl status
Processing status command...
NAME VALUE
-----
appliedLastEventId : mysql-bin.000064:000000002757461;0
...
```

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.

```
appliedLatency : 0.828
```

The latency is measure in seconds. Increasing latency may indicate that the destination database is unable to keep up with the transactions from the 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` [329]. 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.2.14. Terminology: Fields *cancelled*

E.2.15. Terminology: Fields *channel*

E.2.16. Terminology: Fields *channels*

The number of channels being used to apply transactions to the target dataserver. In a standard replication setup there is typically only one channel. When parallel replication is in effect, there will be more than one channel used to apply transactions.

```
channels : 1
```

E.2.17. Terminology: Fields *clusterName*

The name of the cluster. This information is different to the service name and is used to identify the cluster, rather than the individual service information being output.

E.2.18. Terminology: Fields *commits*

E.2.19. Terminology: Fields *committedMinSeqno*

E.2.20. Terminology: Fields *criticalPartition*

E.2.21. Terminology: Fields *currentBlockSize*

E.2.22. Terminology: Fields *currentEventId*

Event ID of the transaction currently being processed

E.2.23. Terminology: Fields *currentLastEventId*

E.2.24. Terminology: Fields *currentLastFragno*

E.2.25. Terminology: Fields *currentLastSeqno*

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 *appliedLatency* figure.

E.2.27. Terminology: Fields *dataServerHost*

E.2.28. Terminology: Fields *discardCount*

E.2.29. Terminology: Fields *docChecksum*

E.2.30. Terminology: Fields `estimatedOfflineInterval`

E.2.31. Terminology: Fields `eventCount`

E.2.32. Terminology: Fields `extensions`

E.2.33. Terminology: Fields `extractTime`

E.2.34. Terminology: Fields `extractor.class`

E.2.35. Terminology: Fields `extractor.name`

E.2.36. Terminology: Fields `filter.#.class`

E.2.37. Terminology: Fields `filter.#.name`

E.2.38. Terminology: Fields `filterTime`

E.2.39. Terminology: Fields `flushIntervalMillis`

E.2.40. Terminology: Fields `fSyncOnFlush`

E.2.41. Terminology: Fields `headSeqno`

E.2.42. Terminology: Fields `intervalGuard`

E.2.43. Terminology: Fields `lastCommittedBlockSize`

The `lastCommittedBlockSize` contains the size of the last block that was committed as part of the block commit procedure. The value is only displayed on applicers and defines the number of events in the last block. By comparing this value to the configured block commit size, the commit type can be determined.

For more information, see [Section 12.1, “Block Commit”](#).

E.2.44. Terminology: Fields `lastCommittedBlockTime`

The `lastCommittedBlockTime` contains the duration since the last committed block. The value is only displayed on applicers and defines the number of seconds since the last block was committed. By comparing this value to the configured block interval, the commit type can be determined.

For more information, see [Section 12.1, “Block Commit”](#).

E.2.45. Terminology: Fields `latestEpochNumber`

E.2.46. Terminology: Fields `logConnectionTimeout`

E.2.47. Terminology: Fields *logDir*

E.2.48. Terminology: Fields *logFileRetainMillis*

E.2.49. Terminology: Fields *logFileSize*

E.2.50. Terminology: Fields *masterConnectUri*

The URI being used to extract THL information. On a master, the information may be empty, or may contain the reference to the underlying extractor source where information is being read.

On a slave, the URI indicates the host from which THL data is being read:

```
masterConnectUri : thl://host1:2112/
```

In a secure installation where SSL is being used to exchange data, the URI protocol will be `thls`:

```
masterConnectUri : thls://host1:2112/
```

E.2.51. Terminology: Fields *masterListenUri*

The URI on which the replicator is listening for incoming slave requests. On a master, this is the URI used to distribute THL information.

```
masterListenUri : thls://host1:2112/
```

E.2.52. Terminology: Fields *maxChannel*

E.2.53. Terminology: Fields *maxDelayInterval*

E.2.54. Terminology: Fields *maxOfflineInterval*

E.2.55. Terminology: Fields *maxSize*

E.2.56. Terminology: Fields *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> treptl status
...
minimumStoredSeqNo : 0
offlineRequests : Offline at sequence number: 5262;Offline at time: 2014-01-01 00:00:00 EST
pendingError : NONE
```

E.2.60. Terminology: Fields *otherTime*

E.2.61. Terminology: Fields *pendingError*

E.2.62. Terminology: Fields *pendingErrorCode*

E.2.63. Terminology: Fields *pendingErrorEventId*

E.2.64. Terminology: Fields *pendingErrorSeqno*

The sequence number where the current error was identified

E.2.65. Terminology: Fields *pendingExceptionMessage*

The current error message that caused the current replicator offline

E.2.66. Terminology: Fields *pipelineSource*

The source for data for the current pipeline. On a master, the pipeline source is the database that the master is connected to and extracting data from. Within a slave, the pipeline source is the master replicator that is providing THL data.

E.2.67. Terminology: Fields *processedMinSeqno*

E.2.68. Terminology: Fields *queues*

E.2.69. Terminology: Fields *readOnly*

E.2.70. Terminology: Fields *relativeLatency*

The relativeLatency is the latency between now and timestamp of the last event written into the local THL. This information gives an indication of how fresh the incoming THL information is. On a master, it indicates whether the master is keeping up with transactions generated on the master database. On a slave, it indicates how up to date the THL read from the master is.

A large value can either indicate that the database is not busy, that a large transaction is currently being read from the source database, or from the master replicator, or that the replicator has stalled for some reason.

An increasing *relativeLatency* on the slave may indicate that the replicator may have stalled and stopped applying changes to the dataserver.

E.2.71. Terminology: Fields *resourcePrecedence*

E.2.72. Terminology: Fields *rmiPort*

E.2.73. Terminology: Fields *role*

The current role of the host in the corresponding service specification. Primary roles are *master* (in [Tungsten Clustering for MySQL 5.0 Manual]) and *slave* (in [Tungsten Clustering for MySQL 5.0 Manual]).

E.2.74. Terminology: Fields *seqnoType*

The internal class used to store the transaction ID. In MySQL replication, the sequence number is typically stored internally as a Java Long (`java.lang.Long`). In heterogeneous replication environments, the type used may be different to match the required information from the source database.

E.2.75. Terminology: Fields *serializationCount*

E.2.76. Terminology: Fields *serialized*

E.2.77. Terminology: Fields *serviceName*

The name of the configured service, as defined when the deployment was first created through `tpm`.

```
serviceName : alpha
```

A replicator may support multiple services. The information is output to confirm the service information being displayed.

E.2.78. Terminology: Fields *serviceType*

The configured service type. Where the replicator is on the same host as the database, the service is considered to be `local`. When reading or write to a remote dataserver, the service is `remote`.

E.2.79. Terminology: Fields *shard_id*

E.2.80. Terminology: Fields *simpleServiceName*

A simplified version of the `serviceName`.

E.2.81. Terminology: Fields *siteName*

E.2.82. Terminology: Fields *sourceId*

E.2.83. Terminology: Fields *stage*

E.2.84. Terminology: Fields *started*

E.2.85. Terminology: Fields *state*

E.2.86. Terminology: Fields *stopRequested*

E.2.87. Terminology: Fields *store.#*

E.2.88. Terminology: Fields *storeClass*

E.2.89. Terminology: Fields *syncInterval*

E.2.90. Terminology: Fields *taskCount*

E.2.91. Terminology: Fields *taskID*

E.2.92. Terminology: Fields *timeInStateSeconds*

E.2.93. Terminology: Fields *timeoutMillis*

E.2.94. Terminology: Fields *totalAssignments*

E.2.95. Terminology: Fields *transitioningTo*

E.2.96. Terminology: Fields *uptimeSeconds*

E.2.97. Terminology: Fields *version*

Appendix F. Internals

Tungsten Replication 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 Replication:

- [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 Replication”](#) — 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 Replication.

F.1. Extending Backup and Restore Behavior

The backup and restore system within Tungsten Replication 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 Replication 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` [207] 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:

```
#!/usr/bin/env ruby
require "/opt/continuent/tungsten/cluster-home/lib/ruby/tungsten"
require "/opt/continuent/tungsten/tungsten-replicator/lib/ruby/backup"
class MyCustomBackupScript < TungstenBackupScript
  def backup
    TU.info("Take a backup with arg1 = #{@options[:arg1]} and myarg = #{@options[:myarg]}")
    storage_file = "/opt/continuent/backups/backup_" +
      Time.now.strftime("%Y-%m-%d-%H-%M") + "_" + rand(100).to_s()
    # Take a backup of the server and store the information to
    storage_file
    TU.cmd_result("echo 'my backup' > #{storage_file}")
    # Write the filename to the final storage file
    TU.cmd_result("echo \"file=#{storage_file}\" > #"
```

```

{@options[:properties]}")
end
  def restore
    storage_file = TU.cmd_result(". #{@options[:properties]}; echo
$file")
    TU.info("Restore a backup from #{@storage_file} with arg1 = #
{@options[:arg1]} and myarg = #{@options[:myarg]}")
    # Process the contents of storage_file to restore into the database
server
end

```

An alternative script using Perl is provided below:

```

#!/usr/bin/perl

use strict;
use warnings;
use Getopt::Long;
use IO::File;

my $argstring = join(' ',@ARGV);

my ($backup,$restore,$properties,$options) = (0,0,'','');

my $result = GetOptions("backup" => \$backup,
"restore" => \$restore,
"properties=s" => \$properties,
"options=s" => \$options,
);

if ($backup)
{
  my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time);
  my $backupfile = sprintf('mcbackup.%04d%02d%02d%02d%02d-%02d.dump',
  ($year+1900),$mon,$mday,$hour,$min,$sec,$$);

  my $out = IO::File->new($backupfile,'w') or die "Couldn't open the backup file: $backupfile";

# Fake backup data

  print $out "Backup data!\n";
  $out->close();

# Update the properties file
  my $propfile = IO::File->new($properties,'w') or die "Couldn't write to the properties file";
  print $propfile "file=$backupfile\n";
  $propfile->close();
}

if ($restore)
{
  warn "Would be restoring information using $argstring\n";
}

exit 0;

```

F.1.4. Enabling a Custom Backup Script

To enable a custom backup script, the installation must be updated through `tpm` to use the script backup method. To update the configuration:

1. Create or copy the backup script into a suitable location, for example `/opt/continuent/share`.
2. Copy the script to each of the datasources within your dataservice.
3. Update the configuration using `tpm`. The `--repl-backup-method` [330] should be set to `script`, and the directory location set using the `--repl-backup-script` [331] 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` [330] option indicates whether the backup script operates in online or offline mode. If set to false, replicator must be in the offline state because the backup process is started.

To pass additional arguments or options to the script, use the `replicator.backup.agent.script.options` property to supply a list of ampersand separate key/value pairs, for example:

```
--property=replicator.backup.agent.script.options="arg1=val1&myarg=val2"
```

These are the custom parameters which are supplied to the script as the value of the `-options` parameter when the script is called.

Once the backup script has been enabled within the configuration it can be used when performing a backup through the standard backup or restore interface:

```
shell> trepctl -host host2 backup -backup script
```

Note

Note that the name of the backup method is `script`, not the actual name of the script being used.

F.2. Character Sets in Database and Tungsten Replication

Character sets within the databases and within the configuration for Java and the wrappers for Tungsten Replication must match to enable the information to be extracted and viewed.

For example, if you are extracting with the UTF-8 character set, the data must be applied to the target database using the same character set. In addition, the Tungsten Replicator should be configured with a corresponding matching character set. For installations where replication is between identical database flavours (for example, MySQL or MySQL) no explicit setting should be made. For heterogeneous deployments, the character set should be set explicitly.

When installing and using Tungsten Replication, be aware of the following aspects when using character sets:

- When installing Tungsten Replication, use the `--java-file-encoding` [348] to `tpm` to configure the character set.
- When using the `thl` command, the character set may need to be explicitly stated to view the content correctly:

```
shell> thl list -charset utf8
```

For more information on setting character sets within your database, see your documentation for the database:

- [MySQL](#)
- [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.propertiesx` file but is not recommended other than for problem diagnosis or specialized testing.
- Replicas store a time zone on statements and row changes extracted from MySQL.
- Replicators use UTC as the session time zone when applying to MySQL replicas.
- Replicators similarly default to UTC when applying transactions to data warehouses like Hadoop, Vertica, or Amazon Redshift.
- The `thl` utility prints time-related data using the default GMT time zone. This can be altered using the `-timezone` option.

Best Practices

We recommend the following steps to ensure successful replication of time-related data.

- Standardize all DBMS server and host time zones to UTC. This minimizes time zone inconsistencies between applications and data stores. The recommendation is particularly important when replicating between different DBMS types, such as MySQL to Hadoop.
- Use the default time zone settings for Tungsten replicator. Do not change the time zones unless specifically recommended by VMware support.
- If you cannot standardize on UTC at least ensure that time zones are set consistently on all hosts and applications.

Arbitrary time zone settings create a number of corner cases for database management beyond replication. Standardizing on UTC helps minimize them, hence is strongly recommended.

F.4. Memory Tuning and Performance

Different areas of Tungsten Replication use memory in different ways, according to the operation and requirements of the component. Specific information on how memory is used by different components and how it is used is available below:

- [Tungsten Replicator](#) — Memory performance and tuning options.

F.4.1. Understanding Tungsten Replicator Memory Tuning

Replicators are implemented as Java processes, which use two types of memory: stack space, which is allocated per running thread and holds objects that are allocated within individual execution stack frames, and heap memory, which is where objects that persist across individual method calls live. Stack space is rarely a problem for Tungsten as replicators rarely run more than 200 threads and use limited recursion. The Java defaults are almost always sufficient. Heap memory on the other hand runs out if the replicator has too many transactions in memory at once. This results in the dreaded Java OutOfMemory exception, which causes the replicator to stop operating. When this happens you need to look at tuning the replicator memory size.

To understand replicator memory usage, we need to look into how replicators work internally. Replicators use a "pipeline" model of execution that streams transactions through 1 or more concurrently executing stages. As you can see from the attached diagram, a slave pipeline might have a stage to read transactions from the master and put them in the THL, a stage to read them back out of the THL into an in-memory queue, and a stage to apply those transactions to the slave. This model ensures high performance as the stages work independently. This streaming model is quite efficient and normally permits Tungsten to transfer even exceedingly large transactions, as the replicator breaks them up into smaller pieces called transaction fragments.

The pipeline model has consequences for memory management. First of all, replicators are doing many things at one, hence need enough memory to hold all current objects. Second, the replicator works fastest if the in-memory queues between stages are large enough that they do not ever become empty. This keeps delays in upstream processing from delaying things at the end of the pipeline. Also, it allows replicators to make use of block commit. Block commit is an important performance optimization in which stages try to commit many transactions at once on slaves to amortize the cost of commit. In block commit the end stage continues to commit transactions until it either runs out of work (i.e., the upstream queue becomes empty) or it hits the block commit limit. Larger upstream queues help keep the end stage from running out of work, hence increase efficiency.

Bearing this in mind, we can alter replicator behavior in a number of ways to make it use less memory or to handle larger amounts of traffic without getting a Java OutOfMemory error. You should look at each of these when tuning memory:

- Property `wrapper.java.memory` in file [wrapper.conf](#). This controls the amount of heap memory available to replicators. 1024 MB is the minimum setting for most replicators. Busy replicators, those that have multiple services, or replicators that use parallel apply should consider using 2048 MB instead. If you get a Java OutOfMemory exception, you should first try raising the current setting to a higher value. This is usually enough to get past most memory-related problems. You can set this at installation time as the `--repl-java-mem-size` [349] parameter.

If you set the heap memory to a very large value (e.g. over 3 GB), you should also consider enabling concurrent garbage collection. Java by default uses mark-and-sweep garbage collection, which may result in long pauses during which network calls to the replicator may fail. Concurrent garbage collection uses more CPU cycles and reduces on-going performance a bit but avoids periods of time during which the replicator is non-responsive. You can set this using the `--repl-java-enable-concurrent-gc` [348] parameter at installation time.)

- Property `replicator.global.buffer.size`. This controls two things, the size of in-memory queues in the replicator as well as the block commit size. If you still have problems after increasing the heap size, try reducing this value. It reduces the number of objects simultaneously stored on the Java heap. A value of 2 is a good setting to try to get around temporary problems. This can be set at installation time as the `--repl-buffer-size` [331] parameter.
- Property `replicator.stage.q-to-dbms.blockCommitRowCount` in the replicator properties file. This parameter sets the block commit count in the final stage in a slave pipeline. If you reduce the global buffer size, it is a good idea to set this to a fixed size, such as 10, to avoid reducing the block commit effect too much. Very low block commit values in this stage can cut update rates on slaves by 50% or more in some cases. This is available at installation time as the `--repl-svc-applier-buffer-size` parameter.
- Property `replicator.extractor.dbms.transaction_frag_size` in the `replicator.properties` file. This parameter controls the size of fragments for long transactions. Tungsten automatically breaks up long transactions into fragments. This parameter controls the number of bytes of binlog per transaction fragment. You can try making this value smaller to reduce overall memory usage if many transactions are simultaneously present. Normally however this value has minimal impact.

Finally, it is worth mentioning that the main cause of out-of-memory conditions in replicators is large transactions. In particular, Tungsten cannot fragment individual statements or row changes, so changes to very large column values can also result in OutOfMemory conditions. For now the best approach is to raise memory, as described above, and change your application to avoid such transactions.

F.5. Tungsten Replicator Stages

F.6. Tungsten Replication Schemas

Appendix G. Frequently Asked Questions (FAQ)

- G.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. 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.3. Do you handle bandwidth/traffic management to the DB servers?

This is not something we have looked at.

- G.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:

```
shell> ntpdate pool.ntp.org
```

Once the command has been executed across all the hosts, trying sending a heartbeat on the master to slaves and checking the latency:

```
shell> trepctl heartbeat
```

- G.5. 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.6. What effect do triggers have on replication, either row or statement-based?

Tungsten Replication does not automatically shut off triggers on slaves. This creates problems on slaves when using row-based replication (RBR) as the trigger will run twice. Tungsten cannot do this because the setting required to do so is not available to MySQL client applications. Typical symptoms are duplicate key errors, though other problems may appear. Consider the following fixes:

- Drop triggers on slaves. This is practical in fan-in for reporting or other cases where you do not need to failover to the slave at a later time.
- Create an "is_master()" function that triggers can use to decide whether they are on the master or slave.
- Use statement replication; not an option for heterogeneous replication.

The `is_master()` approach is simple to implement. First, create a function like the following that returns 1 if we are using the Tungsten user, as would be the case on a slave.

```
create function is_master()
returns boolean
deterministic
return if(substring_index(user(),'@',1) != 'tungsten',true, false);
```

Next add a check to triggers that should not run on the slave. This suppresses trigger action to insert into table bar except on the master.

```
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.

Appendix H. Ecosystem Support

In addition to the core utilities provided by Tungsten Replication, 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 Replication during deployment, monitoring, and management.

H.1. Continuent Github Repositories

In addition to the core product releases, Continuent also support a number of repositories within the [Github](#) system.

To access these repositories and use the tools and information within them, use the `git` command (available from [git-scm.com](#)). To copy the repository to a machine, use the `clone` command, specifying the repository URL:

```
shell> git clone https://github.com/continuent/continuent-tools-hadoop.git
```

This will create a clone of the repository within the current directory.

To keep the copy up to date with any changes in the main Github repository:

```
shell> cd continuent-tools-hadoop
shell> git pull
```

The following tools and functionality are available within these repositories:

- [continuent-tools-hadoop](#)

Provides tools to support the processing and materialization of view data within Hadoop. The repository contains two primary tools, [load-reduce-check](#) and [materialize](#).

Appendix I. Configuration Property Reference