Tungsten Clustering (for MySQL) 7.0 Manual

Continuent Ltd

Tungsten Clustering (for MySQL) 7.0 Manual

Continuent Ltd
Copyright © 2025 Continuent Ltd

Abstract

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

This manual includes information for 7.0, up to and including 7.0.3.

Build date: 2025-08-26 (81786476)

Up to date builds of this document: Tungsten Clustering (for MySQL) 7.0 Manual (Online), Tungsten Clustering (for MySQL) 7.0 Manual (PDF)

Table of Contents

Preface xxi
1. Legal Notice xxi
2. Conventions xxi
3. Quickstart Guide xxii
1. Introduction 23
1.1. Tungsten Replicator 23
1.1.1. Transaction History Log (THL) 24
1.2. Tungsten Manager 24
1.3. Tungsten Connector 25
2. Deployment 27
2.1. Host Types 27
2.1.1. Manager Hosts 27
2.1.2. Connector (Router) Hosts 27
2.1.3. Replicator Hosts 28
2.1.4. Active Witness Hosts 28
2.2. Deployment Sources 29
2.2.1. Using the TAR/GZipped files 29
2.2.2. Using the RPM package files 29
2.3. Common tpm Options During Deployment 30
2.4. Best Practices 30
2.4.1. Best Practices: Deployment 30
2.4.2. Best Practices: Upgrade 31
2.4.3. Best Practices: Operations 32
2.4.4. Best Practices: Maintenance 32
3. Deployment: MySQL Topologies 34
3.1. Deploying Standalone HA Clusters 34
3.1.1. Prepare: Standalone HA Cluster 34
3.1.2. Install: Standalone HA Cluster 35
3.2. Deploying Composite Active/Passive Clustering 37
3.2.1. Prepare: Composite Active/Passive Cluster 37
3.2.2. Install: Composite Active/Passive Cluster 37
3.2.3. Adding a remote Composite Cluster 40
3.3. Deploying Composite Active/Active Clusters 41
3.3.1. Prepare: Composite Active/Active Clusters 42
3.3.2. Install: Composite Active/Active Clusters 43
3.3.3. Best Practices: Composite Active/Active Clusters 46
3.3.4. Resetting a single dataservice 48
3.3.5. Resetting all dataservices 49
3.3.6. Dataserver maintenance 50
3.3.6.1. Fixing Replication Errors 50
3.3.7. Adding a Cluster to a Composite Active/Active Topology 52
3.3.7.1. Prerequisites 52
3.3.7.2. Backup and Restore 52
3.3.7.3. Update Existing Configuration 52
3.3.7.4. New Host Configuration 52
3.3.7.5. Install on new nodes 53
3.3.7.6. Update existing nodes 53
3.3.7.7. Start the new cluster 53
3.3.7.8. Validate and check 53
3.4. Deploying Composite Dynamic Active/Active 53
3.4.1. Enabling Composite Dynamic Active/Active 54
3.5. Deploying Multi-Site/Active-Active Clustering 56
3.5.1. Prepare: Multi-Site/Active-Active Clusters 57
3.5.2. Install: Multi-Site/Active-Active Clusters 57
3.5.3. Best Practices: Multi-Site/Active-Active Clusters 61
3.5.4. Resetting a single dataservice 63
3.5.5. Resetting all dataservices 65
3.5.6. Adding a new Cluster/Dataservice 66
3.5.7. Enabling SSL for Replicators Only 67
3.5.8. Dataserver maintenance 68
3.5.8.1. Fixing Replication Errors 69
3.6. Deploying Tungsten Connector Only 69
3.7. Deploying Additional Datasources, Managers, or Connectors 71
3.7.1. Adding Datasources to an Existing Deployment 71

3.7.2. Adding Active Witnesses to an Existing Deployment 72
3.7.3. Replacing an Active Witness as a Full Cluster Node 73
3.7.4. Replacing a Full Cluster Node as an Active Witness 75
3.7.5. Adding Connectors to an Existing Deployment 76
3.7.6. Converting from a single cluster to a composite cluster 77
3.7.6.1. Convert and add new nodes as a new service 77
3.7.6.2. Convert and move nodes to a new service 81
3.8. Replicating Data Into an Existing Dataservice 82
3.9. Replicating Data Out of a Cluster 86
3.9.1. Prepare: Replicating Data Out of a Cluster 86
3.9.2. Deploy: Replicating Data Out of a Cluster 86
3.10. Migrating and Seeding Data 89
3.10.1. Migrating from MySQL Native Replication 'In-Place' 89
3.10.2. Migrating from MySQL Native Replication Using a New Service 90
3.10.3. Seeding Data through MySQL 92
4. Deployment: Advanced 93
4.1. Deploying Parallel Replication 93
4.1.1. Application Prerequisites for Parallel Replication 93
4.1.2. Enabling Parallel Apply During Install 93
4.1.3. Channels 96
4.1.4. Parallel Replication and Offline Operation 96
4.1.4.1. Clean Offline Operation 96
4.1.4.2. Tuning the Time to Go Offline Cleanly 97
4.1.4.3. Unclean Offline 97
4.1.5. Adjusting Parallel Replication After Installation 97
4.1.5.1. How to Enable Parallel Apply After Installation 97
4.1.5.2. How to Change Channels Safely 98
4.1.5.3. How to Disable Parallel Replication Safely 99
4.1.5.4. How to Switch Parallel Queue Types Safely 100
4.1.6. Monitoring Parallel Replication 100
4.1.6.1. Useful Commands for Parallel Monitoring Replication 100
4.1.6.2. Parallel Replication and Applied Latency On Replicas 100
4.1.6.3. Relative Latency 101
4.1.6.4. Serialization Count 101
4.1.6.5. Maximum Offline Interval 102
4.1.6.6. Workload Distribution 102
4.1.7. Controlling Assignment of Shards to Channels 103
4.1.8. Disk vs. Memory Parallel Queues 104
4.2. Starting and Stopping Tungsten Components 105
4.2.1. Stopping/Starting Tungsten Replicator 105
4.2.2. Stopping/Starting Tungsten Connectors 106
4.2.3. Stopping/Starting Tungsten Managers 106
4.2.4. Restarting the Multi-Site/Active-Active Replicator Service 107
4.3. Configuring Startup on Boot 107
4.3.1. Configuring Multi-Site/Active-Active Replicator Startup on Boot 108
4.4. Removing Datasources, Managers or Connectors 108
4.4.1. Removing a Datasource from an Existing Deployment 108
4.4.1.1. Removing a Composite Datasource/Cluster from an Existing Deployment Using tpm update 109
4.4.1.2. Removing a Composite Datasource/Cluster from an Existing Deployment Using tpm delete-service 110
4.4.1.3. Removing a Composite Datasource/Cluster from an Existing Deployment Manually 110
4.4.2. Removing a Connector from an Existing Deployment 112
5. Deployment: Security 114
5.1. Enabling Security 115
5.1.1. Enabling Security using the Staging Method 115
5.1.2. Enabling Security using the INI Method 116
5.2. Disabling Security 118
5.3. Creating Suitable Certificates 118
5.3.1. Creating Tungsten Internal Certificates Manually 119
5.4. Installing from a Staging Host with Custom Certificates 119
5.4.1. Installing from a Staging Host with Manually-Generated Certificates 119
5.5. Installing via INI File with Custom Certificates 119
5.5.1. Installing via INI File with Manually-Generated Certificates 119
5.6. Installing via INI File with CA-Signed Certificates 119
5.7. Replacing the JGroups Certificate from a Staging Directory 122
5.8. Replacing the TLS Certificate from a Staging Directory 122
5.9. Removing JGroups Encryption from a Staging Directory 122
5.10. Removing JGroups Encryption via INI File 123

5.11. Removing TLS Encryption from a Staging Directory 123
5.12. Removing TLS Encryption via INI File 123
5.13. Enabling Tungsten<>Database Security 123
5.13.1. Enabling Database SSL 124
5.13.2. Configure Tungsten<>Database Secure Communication 124
5.13.3. Configuring Connector SSL 125
5.13.3.1. Enable and Test SSL encryption from the Connector to the Database 126
5.13.3.2. Test SSL encryption from the Application to the Database 128
6. Operations Guide 130
6.1. The Home Directory 130
6.2. Establishing the Shell Environment 130
6.3. Checking Dataservice Status 131
6.3.1. Latency or Relative Latency Display 132
6.3.2. Getting Detailed Information 134
6.3.3. Understanding Datasource Roles 136
6.3.4. Understanding Datasource States 136
6.3.4.1. `ONLINE` State 137
6.3.4.2. `OFFLINE` State 137
6.3.4.3. `FAILED` State 137
6.3.4.4. `SHUNNED` State 137
6.3.5. Understanding Replicator Roles 142
6.3.6. Checking Replication Status 143
6.3.6.1. Understanding Replicator States 145
6.3.6.2. Replicator States During Operations 146
6.3.6.3. Changing Replicator States 146
6.3.7. Changing Datasource States 147
6.3.7.1. Shunning a Datasource 147
6.3.7.2. Recover a Datasource 148
6.3.7.3. Offline a Datasource 148
6.3.7.4. Mark a Datasource as Standby 148
6.3.7.5. Mark a Datasource as Archive 148
6.3.8. Datasource Statuses 149
6.3.9. Datasource States and Policy Mode Interactions 149
6.4. Policy Modes 150
6.4.1. Setting Policy Modes 150
6.4.2. Setting Policy Modes in a Composite cluster 151
6.5. Switching Primary Hosts 151
6.5.1. Automatic Primary Failover 152
6.5.2. Manual Primary Switch 153
6.6. Datasource Recovery Steps 153
6.6.1. Recover a failed Replica 154
6.6.1.1. Recover a Replica from manually shunned state 155
6.6.1.2. Provision or Reprovision a Replica 155
6.6.1.3. Replica Datasource Extended Recovery 156
6.6.2. Recover a failed Primary 157
6.6.2.1. Recover when there are no Primaries 158
6.6.2.2. Manually Failing over a Primary in *`MAINTENANCE`* policy mode 159
6.6.2.3. Split-Brain Discussion 161
6.6.3. Handling Multiple Host Failures 161
6.7. Composite Cluster Switching, Failover and Recovery 163
6.7.1. Composite Cluster Site Switch 163
6.7.2. Composite Cluster Site Failover (Forced Switch) 165
6.7.3. Composite Cluster Site Recovery 167
6.7.4. Composite Cluster Relay Recovery 168
6.8. Composite Active/Active Recovery 168
6.9. Managing Transaction Failures 171
6.9.1. Identifying a Transaction Mismatch 171
6.9.2. Skipping Transactions 174
6.10. Creating a Backup 174
6.10.1. Using a Different Backup Tool 175
6.10.2. Automating Backups 176
6.10.3. Using a Different Directory Location 176
6.10.4. Creating an External Backup 176
6.11. Restoring a Backup 176
6.11.1. Restoring a Specific Backup 177
6.11.2. Restoring an External Backup 177
6.11.3. Restoring from Another Replica 178

6.11.4. Manually Recovering from Another Replica 179
6.11.5. Reprovision a MySQL Replica using rsync 180
6.11.6. Rebuilding a Lost Datasource 181
6.11.7. Resetting an Entire Dataservice from Filesystem Snapshots 182
6.12. Migrating and Seeding Data 182
6.12.1. Migrating from MySQL Native Replication 'In-Place' 182
6.12.2. Migrating from MySQL Native Replication Using a New Service 184
6.13. Resetting a Tungsten Cluster Dataservice 185
6.13.1. Reset a Single Site in a Multi-Site/Active-Active Topology 186
6.13.2. Reset All Sites in a Multi-Site/Active-Active topology 187
6.14. Replicator Fencing 187
6.14.1. Fencing a Replica Node Due to a Replication Fault 188
6.14.2. Fencing Primary Replicators 188
6.15. Upgrading Tungsten Cluster 188
6.15.1. Upgrading using the Staging Method 189
6.15.2. Upgrading when using INI-based configuration 191
6.15.2.1. Upgrading a Single Host using tpm 191
6.15.3. Upgrade/Convert: From Multi-Site/Active-Active (MSAA) to Composite Active/Passive (CAP) 191
6.15.3.1. Conversion Prerequisites 192
6.15.3.2. Step 1: Backups 192
6.15.3.3. Step 2: Redirect Client Connections 192
6.15.3.4. Step 3: Enter MAINTENANCE Mode 193
6.15.3.5. Step 4: Stop the Cross-site Replicators 193
6.15.3.6. Step 5: Export the tracking schema databases 194
6.15.3.7. Step 6: Uninstall the Cross-site Replicators 194
6.15.3.8. Step 7: Create Composite Tracking Schema 194
6.15.3.9. Step 8: Reload the tracking schema for Passive clusters 194
6.15.3.10. Step 9: Stop local cluster Replicators 194
6.15.3.11. Step 10: Remove THL 195
6.15.3.12. Step 11: Export the tracking schema database on Active cluster 195
6.15.3.13. Step 12: Reload the tracking schema for Active cluster 195
6.15.3.14. Step 13: Update Configuration 195
6.15.3.15. Step 14: Install the Software on Active Cluster 196
6.15.3.16. Step 15: Start Local Replicators on Active cluster 196
6.15.3.17. Step 16: Install the Software on remaining Clusters 196
6.15.3.18. Step 17: Start Local Replicators on remaining clusters 196
6.15.3.19. Step 18: Convert Datasource roles for Passive clusters 196
6.15.3.20. Step 19: Upgrade the Software on Connectors 197
6.15.4. Upgrade/Convert: From Multi-Site/Active-Active (MSAA) to Composite Active/Active (CAA) 197
6.15.4.1. Supported Upgrade Paths 198
6.15.4.2. Upgrade Prerequisites 198
6.15.4.3. Step 1: Backups 198
6.15.4.4. Step 2: Stop the Cross-site Replicators 198
6.15.4.5. Step 3: Export the tungsten_* Databases 199
6.15.4.6. Step 4: Uninstall the Cross-site Replicators 200
6.15.4.7. Step 5: Reload the tracking schema 200
6.15.4.8. Step 6: Update Configuration 200
6.15.4.9. Step 7: Enter MAINTENANCE Mode 201
6.15.4.10. Step 8: Stop Managers 201
6.15.4.11. Step 9: Install/Update the Software 201
6.15.4.12. Step 10: Start Managers 201
6.15.4.13. Step 11: Return to AUTOMATIC Mode 201
6.15.4.14. Step 12: Validate 201
6.15.5. Installing an Upgraded JAR Patch 203
6.15.6. Upgrading to v7 203
6.15.6.1. Background 203
6.15.6.2. Upgrade Decisions 204
6.15.6.3. Setup internal encryption and authentication 205
6.15.6.4. Enable Tungsten to Database Encryption 205
6.15.6.5. Enable Connector to Database Encryption 206
6.15.6.6. Enable MySQL SSL 206
6.15.6.7. Steps to upgrade using tpm 207
6.15.6.8. Optional Post-Upgrade steps to configure API 208
6.16. Performing Database or OS Maintenance 208
6.16.1. Performing Maintenance on a Single Replica 209
6.16.2. Performing Maintenance on a Primary 209
6.16.3. Performing Maintenance on an Entire Dataservice 210

6.16.4. Upgrading MySQL 210
6.16.4.1. Connector Behavior during upgrades 211
6.16.4.2. Upgrading between MySQL 5.x and MySQL 8.x 212
6.16.5. Upgrading or Updating your JVM 212
6.16.6. Making Schema Changes 213
6.16.7. Updating Database User Passwords 216
6.17. Monitoring Tungsten Cluster 217
6.17.1. Managing Log Files with logrotate 218
6.17.2. Monitoring Status Using cacti 218
6.17.3. Monitoring Status Using nagios 220
6.17.4. Monitoring Status Using Prometheus Exporters 220
6.17.4.1. Monitoring Status with Exporters Overview 220
6.17.4.2. Customizing the Prometheus Exporter Configuration 221
6.17.4.3. Disabling the Prometheus Exporters 222
6.17.4.4. Managing and Testing Exporters Using the tmonitor Command 222
6.17.4.5. Monitoring Node Status Using the External node_exporter 225
6.17.4.6. Monitoring MySQL Server Status Using the External mysqld_exporter 225
6.17.4.7. Monitoring Tungsten Replicator Status Using the Built-In Exporter 226
6.17.4.8. Monitoring Tungsten Manager Status Using the Built-In Exporter 226
6.17.4.9. Monitoring Tungsten Connector Status Using the Built-In Exporter 227
6.17.4.10. Alerting Using Prometheus Rules 227
6.18. Managing THL Files 228
6.18.1. Moving the THL File Location 228
6.18.1.1. Relocating THL Storage using Symbolic Links 229
6.18.1.2. Relocating THL Storage using Configuration Changes 229
6.18.2. Changing the THL Retention Times 230
6.18.3. Rebuilding THL on the Primary 230
6.18.4. THL Encryption and Compression 231
6.18.4.1. In-Flight Compression 232
6.18.4.2. Encryption and Compression On-Disk 232
7. Tungsten Connector 234
7.1. Connector/Application Basics 234
7.1.1. Connector Control Flow 237
7.2. Basic Connector Configuration 237
7.3. Clients and Deployment 238
7.3.1. Connection Pools 238
7.4. Routing Methods 238
7.4.1. Connector Routing Methods 240
7.4.2. Primary/Replica Selection 240
7.4.3. Connector Load Balancers 241
7.4.4. Specifying Required Latency 241
7.4.4.1. Applied and Relative Latency Comparison 242
7.4.4.2. Advanced Troubleshooting for Latency-based Routing 243
7.4.5. Setting Read Affinity and Direct Reads 243
7.4.6. Setting Read/Write Affinity in Composite Active/Active Environments 243
7.4.7. Setting Negative Affinity 244
7.4.8. Routing using embedded syntax in connect string 244
7.4.9. Connector Datasource Selection in Composite Clusters 244
7.4.10. Smartscale Routing 245
7.4.10.1. Specifying the Session ID 246
7.4.10.2. Enabling SmartScale Routing 246
7.4.10.3. Disabling SmartScale Routing 246
7.4.10.4. SmartScale Exploit 246
7.4.11. Direct Routing 246
7.4.11.1. Enabling Direct Routing 247
7.4.11.2. Limitations of Direct Routing 247
7.4.12. SQL Routing 247
7.4.12.1. Enabling SQL Routing 248
7.4.12.2. Limitations of SQL Routing 248
7.4.13. Host-based Routing 248
7.4.13.1. Enabling Host-based Routing 249
7.4.13.2. Limitations of Host-based Routing 249
7.4.14. Port-based Routing 249
7.4.14.1. Enabling Port-based Routing 250
7.4.15. Read-only Routing 250
7.5. Using Bridge Mode 250
7.5.1. Enabling Bridge Mode 251

7.6. User Authentication 252
7.6.1. `user.map` File Format 252
7.6.2. Dual-Paswords and MySQL 8 254
7.6.3. `user.map` Direct Routing 254
7.6.4. `user.map` Host Options 255
7.6.5. `user.map` Updates 255
7.6.6. Generating `user.map` Entries from a Script 255
7.6.7. Encrypting `user.map` Data 256
7.6.8. Synchronizing `user.map` Data 256
7.6.9. `user.map` Limitations 257
7.6.10. Host-based Authentication 257
7.7. Testing Connectivity Via the Connector 258
7.7.1. Testing Connectivity in Bridge Mode 258
7.7.2. Testing Connectivity in Proxy Mode with No R/W Splitting Enabled 258
7.7.3. Testing Connectivity in Proxy Mode with R/W Splitting Enabled (SmartScale or @direct) 258
7.8. Connector Operational States 258
7.8.1. Connections During Automatic Failure/Failover 259
7.8.2. Connections During Manual Switch 259
7.8.3. Connections During Connection Failures 259
7.8.4. Other Errors 260
7.8.5. Connector Keepalive 260
7.8.6. Connector Change User as Ping 260
7.9. Connector/Manager Interface 261
7.10. Connector Command-Line Interface 261
7.11. Connector Inline Command Interface 262
7.11.1. Connector tungsten cluster status Command 262
7.11.1.1. Connector connector cluster status on the Command-line 263
7.11.2. Connector tungsten connection count Command 264
7.11.3. Connector tungsten connection status Command 265
7.11.4. Connector tungsten flush privileges Command 265
7.11.5. Connector tungsten gc Command 265
7.11.6. Connector tungsten help Command 265
7.11.7. Connector tungsten mem info Command 266
7.11.8. Connector tungsten show [full] processlist Command 266
7.11.9. Connector show slave status Command 266
7.11.10. Connector tungsten show variables Command 267
7.12. Advanced Configuration 267
7.12.1. Working with Proxy Protocol v1 267
7.12.2. Using Multiple Dataservices 268
7.12.3. Advanced Listeners 269
7.12.4. Connector Automatic Reconnect 270
7.12.5. Using the Connector with HAProxy 271
7.12.5.1. Configuring HAProxy using the native Basic Check 272
7.12.5.2. Configuring HAProxy with a Check Script Via systemd 272
7.12.5.3. Configuring HAProxy with a Check Script Via xinetd 274
7.12.5.4. Configuring HAProxy using the native MySQL Check 276
7.12.5.5. Example HAProxy Configuration File 278
7.12.5.6. Monitoring HAProxy Status 278
7.12.6. Using Fall-Back Bridge Mode 279
7.12.6.1. Using Fall-Back SSL To Bridge Mode 279
7.12.7. Using the Max Connections Feature 280
7.12.8. Adjusting the Client Disconnect Delay During Manual Switch 280
7.12.9. Adjusting the Bridge Mode Forced Client Disconnect Timeout 281
7.12.10. Adjusting the Connnector Response to Resource Losses 282
7.12.10.1. Adjusting the Connnector Response to Datasource Loss 282
7.12.10.2. Adjusting the Connnector Response to Manager Loss 282
7.12.11. Connector Logging Configuration 284
7.12.11.1. Connector Logging to Syslog 284
7.12.12. Connector Audit Logging 285
7.12.13. Connector SSL Advertisement Configuration 285
7.12.14. Connector IP Address Configuration 286
7.12.15. Deploying a Connector through Docker 286
7.13. Connector General Limitations 288
8. Tungsten Manager 289
8.1. Tungsten Manager Introduction 289
8.1.1. How Does Failover Work? 289
8.1.1.1. Roles for Nodes and Clusters 289

8.1.1.2. Moving the Primary Role to Another Node or Cluster ... 290
8.1.2. Best Practices for Proper Cluster Failovers ... 291
8.1.3. Command-Line Monitoring Tools ... 291
8.2. Tungsten Manager Failover Behavior ... 293
8.2.1. Failover Replication State Scenarios ... 293
8.2.2. Recovery Behavior After Failover ... 294
8.2.3. Failover Response when MySQL Server Fails ... 295
8.2.4. Failover Response when Replica Applier is Latent ... 295
8.2.5. Understanding the FAILSAFE-SHUN state. ... 295
8.2.5.1. What causes a FAILSAFE-SHUN ... 296
8.2.5.2. How to tune the delay before FAILSAFE-SHUN ... 297
8.2.5.3. FAQ ... 297
8.3. Tungsten Manager Failover Tuning ... 299
8.4. Tungsten Manager Failover Internals ... 300
8.4.1. Manual Switch Versus Automatic Failover ... 300
8.4.2. Switch and Failover Steps for Local Clusters ... 301
8.4.3. Switch and Failover Steps for Composite Services ... 303
8.5. Tungsten Manager Fault Detection, Fencing and Recovery ... 304
8.5.1. Tungsten Manager Definitions ... 304
8.5.2. Cluster Monitoring and Notification Events ... 305
8.5.3. Rule Organization - Detection, Investigation, Fencing, Recovery ... 306
8.6. Cluster State Savepoints ... 308
9. Command-line Tools ... 310
9.1. The cctrl Command ... 310
9.1.1. cctrl Command-line Options ... 310
9.1.2. cctrl Modes ... 313
9.1.3. cctrl Commands ... 313
9.1.3.1. admin Command ... 314
9.1.3.2. cd Command ... 314
9.1.3.3. cluster Command ... 314
9.1.3.4. create composite Command ... 316
9.1.3.5. datasource Command ... 316
9.1.3.6. expert Command ... 320
9.1.3.7. failover Command ... 320
9.1.3.8. help Command ... 320
9.1.3.9. ls Command ... 321
9.1.3.10. members Command ... 322
9.1.3.11. physical Command ... 323
9.1.3.12. ping Command ... 323
9.1.3.13. quit Command ... 323
9.1.3.14. recover Command ... 323
9.1.3.15. replicator Command ... 323
9.1.3.16. router Command ... 324
9.1.3.17. service Command ... 324
9.1.3.18. set Command ... 325
9.1.3.19. show topology Command ... 325
9.1.3.20. switch Command ... 326
9.1.3.21. system summary Command ... 326
9.2. The check_tungsten_latency Command ... 327
9.3. The check_tungsten_online Command ... 328
9.4. The check_tungsten_policy Command ... 329
9.5. The check_tungsten_progress Command ... 329
9.6. The check_tungsten_services Command ... 330
9.7. The clean_release_directory Command ... 331
9.8. The cluster_backup Command ... 331
9.9. The connector Command ... 332
9.10. The deployall Command ... 335
9.11. The dsctl Command ... 336
9.11.1. dsctl get Command ... 336
9.11.2. dsctl set Command ... 337
9.11.3. dsctl reset Command ... 337
9.11.4. dsctl help Command ... 337
9.12. env.sh Script ... 338
9.13. The manager Command ... 338
9.14. The multi_trepctl Command ... 340
9.14.1. multi_trepctl Options ... 340
9.14.2. multi_trepctl Commands ... 342

9.14.2.1. multi_trepctl backups Command 343
9.14.2.2. multi_trepctl heartbeat Command 343
9.14.2.3. multi_trepctl masterof Command 343
9.14.2.4. multi_trepctl list Command 343
9.14.2.5. multi_trepctl run Command 343
9.15. The query Command 344
9.16. The replicator Command 344
9.17. The startall Command 346
9.18. The stopall Command 346
9.19. The tapi Command 347
9.20. The thl Command 350
9.20.1. thl Position Commands 350
9.20.2. thl dsctl Command 352
9.20.3. thl index Command 352
9.20.4. thl info Command 353
9.20.5. thl list Command 353
9.20.6. thl purge Command 356
9.20.6.1. Purging THL Log Information on a Replica 358
9.20.6.2. Purging THL Log Information on a Primary 358
9.20.7. thl tail Command 359
9.20.8. thl help Command 360
9.21. The tmonitor Command 360
9.22. The tpasswd Command 363
9.23. The tprovision Script 363
9.24. The trepctl Command 365
9.24.1. trepctl Options 365
9.24.2. trepctl Global Commands 366
9.24.2.1. trepctl kill Command 367
9.24.2.2. trepctl services Command 367
9.24.2.3. trepctl servicetable Command 369
9.24.2.4. trepctl thl Command 369
9.24.2.5. trepctl version Command 370
9.24.3. trepctl Service Commands 370
9.24.3.1. trepctl backup Command 371
9.24.3.2. trepctl capabilities Command 372
9.24.3.3. trepctl check Command 372
9.24.3.4. trepctl clear Command 373
9.24.3.5. trepctl clients Command 373
9.24.3.6. trepctl error Command 373
9.24.3.7. trepctl flush Command 374
9.24.3.8. trepctl heartbeat Command 374
9.24.3.9. trepctl load Command 376
9.24.3.10. trepctl offline Command 376
9.24.3.11. trepctl offline-deferred Command 376
9.24.3.12. trepctl online Command 377
9.24.3.13. trepctl pause Command 380
9.24.3.14. trepctl perf Command 380
9.24.3.15. trepctl properties Command 381
9.24.3.16. trepctl purge Command 382
9.24.3.17. trepctl qs Command 383
9.24.3.18. trepctl reset Command 384
9.24.3.19. trepctl restore Command 385
9.24.3.20. trepctl resume Command 385
9.24.3.21. trepctl setdynamic Command 385
9.24.3.22. trepctl setrole Command 386
9.24.3.23. trepctl shard Command 386
9.24.3.24. trepctl status Command 387
9.24.3.25. trepctl unload Command 395
9.24.3.26. trepctl wait Command 395
9.25. The tungsten_find_orphaned Command 395
9.26. The tungsten_find_position Command 397
9.27. The tungsten_find_seqno Command 397
9.28. The tungsten_get_mysql_datadir Script 398
9.29. The tungsten_get_status Script 398
9.30. The tungsten_get_ports Script 398
9.31. The tungsten_merge_logs Script 399
9.32. The tungsten_monitor Script 400

9.33. The tungsten_newrelic_event Command 401
9.34. The tungsten_post_process Command 402
9.35. The tungsten_prep_upgrade Script 402
9.36. The tungsten_purge_thl Command 404
9.37. The tungsten_reset_manager Command 404
9.38. The tungsten_send_diag Script 405
9.38.1. Updating tungsten_send_diag 406
9.39. The tungsten_show_processlist Script 407
9.40. The tungsten_skip_all Command 407
9.41. The tungsten_skip_seqno Script 408
9.42. The undeployall Command 409
9.43. The zabbix_tungsten_latency Command 409
9.44. The zabbix_tungsten_online Command 410
9.45. The zabbix_tungsten_progress Command 410
9.46. The zabbix_tungsten_services Command 411
10. The tpm Deployment Command 412
10.1. Processing Installs and Upgrades 413
10.2. tpm INI File Configuration 414
10.2.1. Creating an INI file 414
10.2.2. Installation with INI File 415
10.2.3. Upgrades with an INI File 415
10.2.4. Configuration Changes with an INI file 416
10.3. Converting from Staging to INI 416
10.3.1. Using the translatetoini.pl Script 417
10.4. tpm Staging Configuration 418
10.4.1. Configuring default options for all services 418
10.4.2. Configuring a single service 418
10.4.3. Configuring a single host 419
10.4.4. Reviewing the current configuration 419
10.4.5. Installation 419
10.4.5.1. Installing a set of specific services 419
10.4.5.2. Installing a set of specific hosts 419
10.4.6. Upgrades from a Staging Directory 419
10.4.7. Configuration Changes from a Staging Directory 420
10.5. tpm Commands 421
10.5.1. tpm ask Command 423
10.5.2. tpm check Command 424
10.5.2.1. tpm check ini Command 424
10.5.2.2. tpm check ports Command 424
10.5.3. tpm configure Command 424
10.5.4. tpm connector Command 424
10.5.4.1. tpm connector --hosts 425
10.5.4.2. tpm connector --dataservice-name 425
10.5.4.3. tpm connector --samples 425
10.5.5. tpm copy Command 425
10.5.6. tpm delete-service Command 426
10.5.7. tpm diag Command 428
10.5.8. tpm fetch Command 429
10.5.9. tpm firewall Command 429
10.5.10. tpm find-seqno Command 430
10.5.11. tpm generate-haproxy-for-api Command 430
10.5.12. tpm help Command 431
10.5.13. tpm install Command 431
10.5.14. tpm mysql Command 432
10.5.15. tpm policy Command 432
10.5.16. tpm post-process Command 432
10.5.17. tpm promote-connector Command 434
10.5.18. tpm purge-thl Command 434
10.5.19. tpm query Command 435
10.5.19.1. tpm query config 435
10.5.19.2. tpm query dataservices 435
10.5.19.3. tpm query deployments 435
10.5.19.4. tpm query modified-files 436
10.5.19.5. tpm query staging 436
10.5.19.6. tpm query topology 436
10.5.19.7. tpm query usermap 436
10.5.19.8. tpm query version 436

10.5.20. tpm report Command 436
10.5.21. tpm reverse Command 439
10.5.22. tpm uninstall Command 440
10.5.23. tpm update Command 440
10.5.24. tpm validate Command 441
10.5.25. tpm validate-update Command 441
10.6. tpm Common Options 442
10.7. tpm Validation Checks 444
10.8. tpm Configuration Options 462
10.8.1. A tpm Options 473
10.8.2. B tpm Options 475
10.8.3. C tpm Options 477
10.8.4. D tpm Options 484
10.8.5. E tpm Options 490
10.8.6. F tpm Options 493
10.8.7. H tpm Options 493
10.8.8. I tpm Options 494
10.8.9. J tpm Options 494
10.8.10. L tpm Options 497
10.8.11. M tpm Options 498
10.8.12. N tpm Options 504
10.8.13. O tpm Options 505
10.8.14. P tpm Options 505
10.8.15. R tpm Options 508
10.8.16. S tpm Options 513
10.8.17. T tpm Options 518
10.8.18. U tpm Options 521
10.8.19. W tpm Options 521
11. Tungsten REST APIv2 522
11.1. Getting Started with Tungsten REST API 522
11.1.1. Configuring the API 522
11.1.1.1. Network Ports 522
11.1.1.2. User Management 523
11.1.1.3. SSL/Encryption 523
11.1.1.4. Enabling and Disabling the API 524
11.1.2. How to Access the API 524
11.1.2.1. CURL calls and Examples 524
11.1.2.2. tapi 525
11.1.2.3. External Tools 525
11.1.3. Data Structures 525
11.1.3.1. Generic Payloads 525
11.1.3.2. INPUT and OUTPUT payloads 525
11.1.4. Summary of all API TPM options 526
11.2. Proxy (Connector) API Specifics 527
11.2.1. Proxy States 527
11.2.2. List and Set Proxy Configuration 527
11.2.3. User Map 528
11.2.4. Cluster Configuration Manipulation 528
11.2.4.1. Creating a Data Service 528
11.2.4.2. Adding a Data Source 528
11.2.4.3. Creating a Full Cluster Configuration 529
11.2.4.4. Data Source Changes, Switches and Failovers 529
11.2.4.5. Resetting the Cluster Configuration 529
11.3. Manager API Specifics 530
11.3.1. Manager Status 530
11.3.2. Cluster Topology 530
11.3.3. Service Status 532
11.3.4. Datasource Status 532
11.3.5. Cluster Control Commands 533
11.3.5.1. Ping a Datasource 533
11.3.5.2. Issue a Switch 533
11.4. Replicator API Specifics 533
11.4.1. Replicator Endpoints 534
11.4.1.1. services 534
11.4.1.2. status 534
11.4.1.3. version 537
11.4.1.4. offline/online 537

11.4.1.5. purge 538
11.4.1.6. reset 538
11.4.2. Service Endpoints 538
11.4.2.1. backupCapabilities 538
11.4.2.2. backup / restore 539
11.4.2.3. setrole 539
11.4.3. Service THL Endpoints 539
11.4.3.1. compression / encryption 540
11.4.3.2. genkey 540
12. Replication Filters 541
12.1. Enabling/Disabling Filters 542
12.2. Enabling Additional Filters 543
12.3. Filter Reference 544
12.3.1. `ansiquotes.js` Filter 545
12.3.2. BidiRemoteSlave (BidiSlave) Filter 545
12.3.3. `breadcrumbs.js` Filter 546
12.3.4. CaseTransform Filter 547
12.3.5. ColumnName Filter 548
12.3.6. ConvertStringFromMySQL Filter 549
12.3.7. DatabaseTransform (dbtransform) Filter 550
12.3.8. `dbrename.js` Filter 551
12.3.9. `dbselector.js` Filter 552
12.3.10. `dbupper.js` Filter 553
12.3.11. `dropcolumn.js` Filter 553
12.3.12. `dropcomments.js` Filter 555
12.3.13. `dropddl.js` Filter 555
12.3.14. `dropmetadata.js` Filter 556
12.3.15. `droprow.js` Filter 557
12.3.16. `dropstatementdata.js` Filter 558
12.3.17. `dropsqlmode.js` Filter 559
12.3.18. `dropxa.js` Filter 559
12.3.19. Dummy Filter 560
12.3.20. EnumToString Filter 560
12.3.21. EventMetadata Filter 561
12.3.22. `foreignkeychecks.js` Filter 561
12.3.23. Heartbeat Filter 562
12.3.24. `insertsonly.js` Filter 562
12.3.25. Logging Filter 563
12.3.26. MySQLSessionSupport (mysqlsessions) Filter 563
12.3.27. mapcharset Filter 563
12.3.28. NetworkClient Filter 564
12.3.28.1. Network Client Configuration 565
12.3.28.2. Network Filter Protocol 566
12.3.28.3. Sample Network Client 568
12.3.29. `nocreatedbifnotexists.js` Filter 569
12.3.30. OptimizeUpdates Filter 570
12.3.31. PrimaryKey Filter 571
12.3.31.1. Setting Custom Primary Key Definitions 572
12.3.32. PrintEvent Filter 574
12.3.33. Rename Filter 574
12.3.33.1. Rename Filter Examples 575
12.3.34. Replicate Filter 576
12.3.35. ReplicateColumns Filter 577
12.3.36. Row Add Database Name Filter 577
12.3.37. Row Add Transaction Info Filter 578
12.3.38. SetToString Filter 579
12.3.39. Shard Filter 581
12.3.40. `shardbyseqno.js` Filter 581
12.3.41. `shardbytable.js` Filter 582
12.3.42. SkipEventByType Filter 582
12.3.43. TimeDelay (delay) Filter 583
12.3.44. TimeDelayMsFilter (delayInMS) Filter 584
12.3.45. `tosingledb.js` Filter 584
12.3.46. `truncatetext.js` Filter 585
12.3.47. `zerodate2null.js` Filter 585
12.4. Standard JSON Filter Configuration 586
12.4.1. Rule Handling and Processing 587

12.4.2. Schema, Table, and Column Selection 587
12.5. JavaScript Filters 588
12.5.1. Writing JavaScript Filters 588
12.5.1.1. Implementable Functions 589
12.5.1.2. Getting Configuration Parameters 589
12.5.1.3. Logging Information and Exceptions 590
12.5.1.4. Exposed Data Structures 590
12.5.2. Installing Custom JavaScript Filters 595
12.5.2.1. Step 1: Copy JavaScript files 596
12.5.2.2. Step 2: Create Template Files 596
12.5.2.3. Step 3: (Optional) Copy json files 596
12.5.2.4. Step 4: Update Configuration 596
13. Performance, Tuning and Testing 597
13.1. Block Commit 597
13.1.1. Monitoring Block Commit Status 598
13.2. Improving Network Performance 598
13.3. Tungsten Replicator Block Commit and Memory Usage 599
13.4. Connector Memory Management 601
13.5. Functional Testing 601
13.5.1. Manual and Automatic Failover 602
13.5.2. Backup and Restore 602
13.5.3. Connectivity 603
13.5.4. Performance and Other Tests 603
A. Release Notes 605
A.1. Tungsten Clustering 7.0.3 GA (4 Apr 2023) 605
A.2. Tungsten Clustering 7.0.2 GA (9 Dec 2022) 608
A.3. Tungsten Clustering 7.0.1 GA (13 June 2022) 614
A.4. Tungsten Clustering 7.0.0 GA (29 Mar 2022) 618
B. Prerequisites 629
B.1. Requirements 629
B.1.1. Operating Systems Support 629
B.1.2. Database Support 629
B.1.3. RAM Requirements 631
B.1.4. Disk Requirements 631
B.1.5. Java Requirements 631
B.1.6. Cloud Deployment Requirements 632
B.1.7. Docker Support Policy 633
B.1.7.1. Overview 633
B.1.7.2. Background 633
B.1.7.3. Current State 633
B.1.7.4. Summary 633
B.2. Staging Host Configuration 634
B.3. Host Configuration 635
B.3.1. Operating System Version Support 635
B.3.2. Creating the User Environment 636
B.3.3. Configuring Network and SSH Environment 637
B.3.3.1. Network Ports 638
B.3.3.2. SSH Configuration 639
B.3.3.3. Host Availability Checks 640
B.3.4. Directory Locations and Configuration 640
B.3.5. Configure Software 640
B.3.6. sudo Configuration 641
B.3.7. SELinux Configuration 642
B.4. MySQL Database Setup 642
B.4.1. MySQL Version Support 642
B.4.2. MySQL Configuration 643
B.4.3. MySQL Configuration for Active/Active Deployments 645
B.4.4. MySQL User Configuration 646
B.4.5. MySQL Unprivileged Users 647
B.5. Prerequisite Checklist 648
C. Troubleshooting 650
C.1. Contacting Support 650
C.1.1. Support Request Procedure 650
C.1.2. Creating a Support Account 650
C.1.3. Open a Support Ticket 650
C.1.4. Open a Support Ticket via Email 650
C.1.5. Getting Updates for all Company Support Tickets 650

C.1.6. Support Severity Level Definitions 651
C.2. Support Tools 651
C.2.1. Generating Diagnostic Information 651
C.2.2. Generating Advanced Diagnostic Information 652
C.2.3. Using tungsten_upgrade_manager 653
C.3. Error/Cause/Solution 653
C.3.1. MySQL is incorrectly configured 653
C.3.2. WARNING: An illegal reflective access operation has occurred 654
C.3.3. element 'mysql_readonly' not found in path 654
C.3.4. Latency is high: master:ONLINE, progress=41331580333, THL latency=78849.733 654
C.3.5. Backup/Restore is not bringing my host back to normal 655
C.3.6. Attempt to write new log record with equal or lower fragno: seqno=3 previous stored fragno=32767 attempted new fragno=-32768 655
C.3.7. Triggers not firing correctly on Replica 656
C.3.8. ERROR backup.BackupTask Backup operation failed: null 656
C.3.9. cctrl reports MANAGER{state=STOPPED} 656
C.3.10. Starting replication after performing a restore because of an invalid restart sequence number 657
C.3.11. WARN [KeepAliveTimerTask] - Error while sending a KEEP_ALIVE query to connection. 657
C.3.12. [S1000][unixODBC][MySQL][ODBC 5.3(w) Driver]SSL connection error: unknown error number [ISQL]ERROR: Could not SQLConnect 658
C.3.13. MySQL 8.0+, User Roles and Smartscale 658
C.3.14. Connector shows errors with "java.net.SocketException: Broken pipe" 659
C.3.15. Services requires a reset 659
C.3.16. trepctl status hangs 659
C.3.17. Replicator reports an Out of Memory error 659
C.3.18. Backup agent name not found: xtrabackup-full 660
C.3.19. ERROR 1010 (HY000) at line 5094506: Error dropping database (can't rmdir './mysql-bin/', errno: 17) 660
C.3.20. ERROR >> host1 >> can't alloc thread 661
C.3.21. Too many open processes or files 661
C.3.22. Replicator fails to connect after updating password 661
C.3.23. Cluster remains in MAINTENANCE mode after tpm update 662
C.3.24. Replication latency very high 662
C.3.25. ERROR 2013 (HY000) at line 583: Lost connection to MySQL server during query 663
C.3.26. pendingExceptionMessage": "Unable to update last commit seqno: Incorrect datetime value: '2025-03-13 02:02:26' for column 'update_timestamp' at row 1 663
C.3.27. Lots of entries added to replicator log 663
C.3.28. Error: could not settle on encryption_client algorithm 664
C.3.29. Unable to update the configuration of an installed directory 665
C.3.30. ERROR 1580 (HY000) at line 5093787: You cannot 'DROP' a log table if logging is enabled 665
C.3.31. The Primary replicator stopped with a JDBC error. 665
C.3.32. Replicator runs out of memory 666
C.4. Known Issues 666
C.4.1. Triggers 666
C.5. Troubleshooting Timeouts 668
C.6. Troubleshooting Backups 668
C.7. Running Out of Diskspace 668
C.8. Troubleshooting SSH and tpm 668
C.9. Troubleshooting Data Differences 669
C.9.1. Identify Structural Differences 669
C.9.2. Identify Data Differences 669
C.10. Comparing Table Data 670
C.11. Troubleshooting Memory Usage 670
D. Files, Directories, and Environment 671
D.1. The Tungsten Install Directory 671
D.1.1. The `backups` Directory 671
D.1.1.1. Automatically Deleting Backup Files 671
D.1.1.2. Manually Deleting Backup Files 672
D.1.1.3. Copying Backup Files 672
D.1.1.4. Relocating Backup Storage 673
D.1.2. The `releases` Directory 674
D.1.3. The `service_logs` Directory 674
D.1.4. The `share` Directory 675
D.1.5. The `thl` Directory 675
D.1.6. The `tungsten` Directory 676
D.1.6.1. The `tungsten-connector` Directory 676
D.1.6.2. The `tungsten-manager` Directory 676
D.1.6.3. The `tungsten-replicator` Directory 676

D.2. Log Files 677
D.3. Environment Variables 677
E. Terminology Reference 678
E.1. Transaction History Log (THL) 678
E.1.1. THL Format 678
E.2. Generated Field Reference 681
E.2.1. Terminology: Fields *masterConnectUri* 681
E.2.2. Terminology: Fields *masterListenUri* 682
E.2.3. Terminology: Fields *accessFailures* 682
E.2.4. Terminology: Fields *active* 682
E.2.5. Terminology: Fields *activeSeqno* 682
E.2.6. Terminology: Fields *appliedLastEventId* 682
E.2.7. Terminology: Fields *appliedLastGoodLatency* 682
E.2.8. Terminology: Fields *appliedLastGoodSeqno* 682
E.2.9. Terminology: Fields *appliedLastSeqno* 682
E.2.10. Terminology: Fields *appliedLatency* 682
E.2.11. Terminology: Fields *applier.class* 683
E.2.12. Terminology: Fields *applier.name* 683
E.2.13. Terminology: Fields *applyTime* 683
E.2.14. Terminology: Fields *autoRecoveryEnabled* 683
E.2.15. Terminology: Fields *autoRecoveryTotal* 683
E.2.16. Terminology: Fields *averageBlockSize* 683
E.2.17. Terminology: Fields *blockCommitRowCount* 683
E.2.18. Terminology: Fields *cancelled* 683
E.2.19. Terminology: Fields *channel* 683
E.2.20. Terminology: Fields *channels* 683
E.2.21. Terminology: Fields *clusterName* 683
E.2.22. Terminology: Fields *commits* 683
E.2.23. Terminology: Fields *committedMinSeqno* 683
E.2.24. Terminology: Fields *criticalPartition* 683
E.2.25. Terminology: Fields *currentBlockSize* 684
E.2.26. Terminology: Fields *currentEventId* 684
E.2.27. Terminology: Fields *currentLastEventId* 684
E.2.28. Terminology: Fields *currentLastFragno* 684
E.2.29. Terminology: Fields *currentLastSeqno* 684
E.2.30. Terminology: Fields *currentTimeMillis* 684
E.2.31. Terminology: Fields *dataServerHost* 684
E.2.32. Terminology: Fields *discardCount* 684
E.2.33. Terminology: Fields *doChecksum* 684
E.2.34. Terminology: Fields *estimatedOfflineInterval* 684
E.2.35. Terminology: Fields *eventCount* 684
E.2.36. Terminology: Fields *extensions* 684
E.2.37. Terminology: Fields *extractTime* 684
E.2.38. Terminology: Fields *extractor.class* 684
E.2.39. Terminology: Fields *extractor.name* 684
E.2.40. Terminology: Fields *filter.#.class* 684
E.2.41. Terminology: Fields *filter.#.name* 684
E.2.42. Terminology: Fields *filterTime* 684
E.2.43. Terminology: Fields *flushIntervalMillis* 684
E.2.44. Terminology: Fields *fsyncOnFlush* 685
E.2.45. Terminology: Fields *headSeqno* 685
E.2.46. Terminology: Fields *intervalGuard* 685
E.2.47. Terminology: Fields *lastCommittedBlockSize* 685
E.2.48. Terminology: Fields *lastCommittedBlockTime* 685
E.2.49. Terminology: Fields *latestEpochNumber* 685
E.2.50. Terminology: Fields *logConnectionTimeout* 685
E.2.51. Terminology: Fields *logDir* 685
E.2.52. Terminology: Fields *logFileRetainMillis* 685
E.2.53. Terminology: Fields *logFileSize* 685
E.2.54. Terminology: Fields *maxChannel* 685
E.2.55. Terminology: Fields *maxDelayInterval* 685
E.2.56. Terminology: Fields *maxOfflineInterval* 685
E.2.57. Terminology: Fields *maxSize* 685
E.2.58. Terminology: Fields *maximumStoredSeqNo* 685
E.2.59. Terminology: Fields *minimumStoredSeqNo* 686
E.2.60. Terminology: Fields *name* 686
E.2.61. Terminology: Fields *offlineRequests* 686

E.2.62. Terminology: Fields *otherTime* 686
E.2.63. Terminology: Fields *pendingError* 686
E.2.64. Terminology: Fields *pendingErrorCode* 686
E.2.65. Terminology: Fields *pendingErrorEventId* 686
E.2.66. Terminology: Fields *pendingErrorSeqno* 686
E.2.67. Terminology: Fields *pendingExceptionMessage* 686
E.2.68. Terminology: Fields *pipelineSource* 686
E.2.69. Terminology: Fields *processedMinSeqno* 686
E.2.70. Terminology: Fields *queues* 686
E.2.71. Terminology: Fields *readOnly* 686
E.2.72. Terminology: Fields *relativeLatency* 686
E.2.73. Terminology: Fields *resourcePrecedence* 687
E.2.74. Terminology: Fields *rmiPort* 687
E.2.75. Terminology: Fields *role* 687
E.2.76. Terminology: Fields *seqnoType* 687
E.2.77. Terminology: Fields *serializationCount* 687
E.2.78. Terminology: Fields *serialized* 687
E.2.79. Terminology: Fields *serviceName* 687
E.2.80. Terminology: Fields *serviceType* 687
E.2.81. Terminology: Fields *shard_id* 687
E.2.82. Terminology: Fields *simpleServiceName* 687
E.2.83. Terminology: Fields *siteName* 687
E.2.84. Terminology: Fields *sourceId* 687
E.2.85. Terminology: Fields *stage* 687
E.2.86. Terminology: Fields *started* 687
E.2.87. Terminology: Fields *state* 687
E.2.88. Terminology: Fields *stopRequested* 688
E.2.89. Terminology: Fields *store.#* 688
E.2.90. Terminology: Fields *storeClass* 688
E.2.91. Terminology: Fields *syncInterval* 688
E.2.92. Terminology: Fields *taskCount* 688
E.2.93. Terminology: Fields *taskId* 688
E.2.94. Terminology: Fields *timeInCurrentEvent* 688
E.2.95. Terminology: Fields *timeInStateSeconds* 688
E.2.96. Terminology: Fields *timeoutMillis* 688
E.2.97. Terminology: Fields *totalAssignments* 688
E.2.98. Terminology: Fields *transitioningTo* 688
E.2.99. Terminology: Fields *uptimeSeconds* 688
E.2.100. Terminology: Fields *version* 688
F. Internals 689
F.1. Extending Backup and Restore Behavior 689
F.1.1. Backup Behavior 689
F.1.2. Restore Behavior 689
F.1.3. Writing a Custom Backup/Restore Script 690
F.1.4. Enabling a Custom Backup Script 691
F.2. Character Sets in Database and Tungsten Cluster 692
F.3. Understanding Replication of Date/Time Values 692
F.4. Tungsten Clustering Schemas 692
G. Frequently Asked Questions (FAQ) 695
G.1. General Questions 695
G.2. Cloud Deployment and Management 696

List of Figures

3.1. Topologies: Standalone HA Cluster 34
3.2. Topologies: Composite Active/Passive Cluster 37
3.3. Topologies: Composite Active/Active Clusters 42
3.4. Topologies: Composite Dynamic Active/Active Clusters 54
3.5. Topologies: Multi-Site/Active-Active Clusters 56
3.6. Topologies: Replicating into a Dataservice 82
3.7. Topologies: Replicating Data Out of a Cluster 86
3.8. Migration: Migrating Native Replication using a New Service 91
5.1. Security Internals: Cluster Communication Channels 114
6.1. Migration: Migrating Native Replication using a New Service 184
6.2. Cacti Monitoring: Example Graphs 219
7.1. Tungsten Connector Basic Architecture 234
7.2. Basic MySQL/Application Connectivity 235
7.3. Advanced MySQL/Application Connectivity 236
7.4. Using Tungsten Connector for MySQL/Application Connectivity 237
7.5. Tungsten Connector during a failed datasource 237
7.6. Tungsten Connector routing architecture 239
7.7. Tungsten Connector Bridge Mode Architecture 250
7.8. Tungsten Connector Authentication 252
8.1. Failover Scenario 1 293
8.2. Failover Scenario 2 294
8.3. Failover Scenario 3 294
10.1. Internals: Cluster Communication Channels 438
12.1. Filters: Pipeline Stages on Extractors 541
12.2. Filters: Pipeline Stages on Appliers 542
B.1. Tungsten Deployment 634

List of Tables

1.1. Key Terminology 23
2.1. Key Terminology 27
7.1. Routing Method Selection 240
7.2. Connector Command Line Sub-Commands 261
7.3. Inline Interface Commands 262
9.1. cctrl Command-line Options 310
9.2. cctrl Command-line Options 311
9.3. cctrl Command-line Options 311
9.4. cctrl Command-line Options 311
9.5. cctrl Command-line Options 311
9.6. cctrl Command-line Options 312
9.7. cctrl Command-line Options 312
9.8. cctrl Command-line Options 312
9.9. cctrl Command-line Options 312
9.10. cctrl Command-line Options 312
9.11. cctrl Command-line Options 313
9.12. cctrl Commands 313
9.13. cctrl datasource Commands 316
9.14. cctrl replicator Commands 323
9.15. check_tungsten_latency Options 327
9.16. check_tungsten_online Options 328
9.17. check_tungsten_policy Options 329
9.18. check_tungsten_progress Options 329
9.19. check_tungsten_services Options 330
9.20. cluster_backup Command-line Options 331
9.21. connector Commands 332
9.22. dsctl Commands 336
9.23. dsctl Command-line Options 336
9.24. dsctl Command-line Options 336
9.25. dsctl Command-line Options 337
9.26. manager Commands 338
9.27. multi_trepctl Command-line Options 340
9.28. multi_trepctl `--output` Option 341
9.29. multi_trepctl Commands 342
9.30. query Common Options 344
9.31. replicator Commands 344
9.32. replicator Commands Options for `condrestart` 345
9.33. replicator Commands Options for `console` 345
9.34. replicator Commands Options for `restart` 345
9.35. replicator Commands Options for `start` 346
9.36. tapi Generic Options 347
9.37. tapi CURL-related Options 347
9.38. tapi Nagios/NRPE/Zabbix-related Options 347
9.39. tapi Admin-related Options 348
9.40. tapi Filter-related Options 348
9.41. tapi API-related Options 348
9.42. tapi Status-related Options 349
9.43. tapi Backup and Restore-related Options 349
9.44. thl Options 350
9.45. tmonitor Common Options 360
9.46. tpasswd Common Options 363
9.47. tprovision Command-line Options 364
9.48. trepctl Command-line Options 365
9.49. trepctl Replicator Wide Commands 366
9.50. trepctl Service Commands 370
9.51. trepctl backup Command Options 371
9.52. trepctl clients Command Options 373
9.53. trepctl offline-deferred Command Options 376
9.54. trepctl online Command Options 378
9.55. trepctl pause Command Options 380
9.56. trepctl purge Command Options 382
9.57. trepctl reset Command Options 384
9.58. trepctl resume Command Options 385
9.59. trepctl setdynamic Command Options 385

9.60. trepctl setrole Command Options 386
9.61. trepctl shard Command Options 386
9.62. trepctl status Command Options 387
9.63. trepctl wait Command Options 395
9.64. tungsten_find_orphaned Options 396
9.65. tungsten_find_position Options 397
9.66. tungsten_get_mysql_datadir Command-line Options 398
9.67. tungsten_get_ports Options 398
9.68. tungsten_merge_logs Command-line Options 399
9.69. tungsten_monitor Command-line Options 400
9.70. tungsten_newrelic_event Command-line Options 401
9.71. tungsten_prep_upgrade Command-line Options 403
9.72. tungsten_reset_manager Options 404
9.73. tungsten_send_diag Command-line Options 405
9.74. tungsten_show_processlist Command-line Options 407
9.75. tungsten_skip_all Options 408
9.76. tungsten_skip_seqno Command-line Options 408
9.77. zabbix_tungsten_latency Options 409
9.78. zabbix_tungsten_online Options 410
9.79. zabbix_tungsten_progress Options 410
9.80. check_tungsten_services Options 411
10.1. tpm Core Options 421
10.2. tpm Commands 422
10.3. tpm ask Arguments 423
10.4. tpm ask Context Options 424
10.5. tpm copy Common Options 425
10.6. tpm delete-service Common Options 426
10.7. tpm diag Common Options 428
10.8. tungsten_find_seqno Options 430
10.9. tpm generate-haproxy-for-api Common Options 430
10.10. tpm policy Options 432
10.11. tpm-post-process Options 432
10.12. tpm purge-thl Options 434
10.13. tpm report Common Options 437
10.14. tpm Common Options 442
10.15. tpm Validation Checks 445
10.16. tpm Configuration Options 463
B.1. Tungsten OS Support 629
B.2. MySQL/Tungsten Version Support 630
C.1. tungsten_upgrade_manager Options 653
D.1. Continuent Tungsten Directory Structure 671
D.2. Continuent Tungsten `tungsten` Sub-Directory Structure 676
E.1. THL Event Format 679

Preface

This manual documents Tungsten Cluster 7.0 up to and including 7.0.3 build 141. Differences between minor versions are highlighted stating the explicit minor release version, such as 7.0.3.

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` [494] both link automatically to the corresponding reference page.
- Parameters or information explicitly used to set values to commands or options is formatted using *`this style`*.
- Option values, for example on the command-line are marked up using this format: `--help`. Where possible, all option values are directly linked to the reference information for that option.
- Commands, including sub-commands to a command-line tool are formatted using Text in this style. Commands are also automatically linked to their respective documentation page when this is known. For example, tpm links automatically to the corresponding reference page.
- `Text in this style` indicates literal or character sequence text used to show a specific value.
- Filenames, directories or paths are shown like this `/etc/passwd`. Filenames and paths are automatically linked to the corresponding reference page if available.

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

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

```
shell> cd /opt/continuent/software
shell> tar zxvf tungsten-clustering-7.0.3-141.tar.gz
```

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

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

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

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

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

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

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

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

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

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

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

3. Quickstart Guide

- Are you planning on completing your first installation?
  - Do you know the Section B.1, “Requirements”?
  - Have you followed the Appendix B, *Prerequisites*?
  - Have you chosen your deployment type from Chapter 2, *Deployment*? Is this a Primary/Replica deployment?
- Do you want to upgrade to the latest version?

  See Section 10.5.23, “tpm update Command”.
- Are you trying to update or change the configuration of your system?

  See Section 10.5.23, “tpm update Command”.
- Has your system suffered a failure?

  For recovery methods and instructions, see Section 6.6, “Datasource Recovery Steps”.
- Would you like to perform database or operating system maintenance?

  See Section 6.16, “Performing Database or OS Maintenance”.
- Do you need to backup or restore your system?

  For backup instructions, see Section 6.10, “Creating a Backup”, and to restore a previously made backup, see Section 6.11, “Restoring a Backup”.

Chapter 1. Introduction

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

- Tungsten Replicator

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

- Tungsten Manager

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

- Tungsten Connector (or Tungsten Proxy)

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

While there is no specific SLA because every customer's environment is different, we strive to deliver a very low RTO and a very high RPO. For example, a cluster failover normally takes around 30 seconds depending on load, so the RTO is typically under 1 minute. Additionally, the RPO is 100%, since we keep copies of the database on Replica nodes, so that a failover happens with zero data loss under the vast majority of conditions.

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

Table 1.1. Key Terminology

Continuent Term	Traditional Term	Description
composite dataservice	Multi-Site Cluster	A configured Tungsten Cluster service consisting of multiple dataservices, typically at different physical locations.
dataservice	Cluster	The collection of machines that make up a single Tungsten Dataservice. Individual hosts within the dataservice are called datasources. Each dataservice is identified by a unique name, and multiple dataservices can be managed from one server.
dataserver	Database	The database on a host.
datasource	Host or Node	One member of a dataservice and the associated Tungsten components.
staging host	-	The machine (and directory) from which Tungsten Cluster™ is installed and configured. The machine does not need to be the same as any of the existing hosts in the dataservice.
active witness	-	A machine in the dataservice that runs the manager process but is not running a database server. This server will be used to establish quorum in the event that a datasource becomes unavailable.
coordinator		The datasource or active witness in a dataservice that is responsible for making decisions on the state of the dataservice. The coordinator is usually the member that has been running the longest. It will not always be the Primary. When the manager process on the coordinator is stopped, or no longer available, a new coordinator will be chosen from the remaining members.

1.1. Tungsten Replicator

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

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

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

- Extractor

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

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

- Applier

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

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

- Transaction History Log (THL)

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

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

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

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

1.1.1. Transaction History Log (THL)

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

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

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

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

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

1.2. Tungsten Manager

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

Primarily, the Tungsten Manager handles the following tasks:

- Monitors the replication status of each datasource (node) within the cluster.

- Communicates and updates Tungsten Connector with information about the status of each datasource. In the event of a change of status, Tungsten Connectors are notified so that queries can be redirected accordingly.
- Manages all the individual components of the system. Using the Java JMX system the manager is able to directly control the different components to change status, control the replication process, and
- Checks to determine the availability of datasources by using the system ping utility (The default) or Echo TCP/IP on port 7 to determine whether a host is available. The configuration of the protocol to be used can be made by adjusting the manager properties. For more information, see Section B.3.3.3, "Host Availability Checks".
- Includes an advanced rules engine. The rule engine is used to respond to different events within the cluster and perform the necessary operations to keep the dataservice in optimal working state. During any change in status, whether user-selected or automatically triggered due to a failure, the rules are used to make decisions about whether to restart services, swap Primaries, or reconfigure connectors.

Please see the Tungsten Manager documentation section Chapter 8, *Tungsten Manager* for more information.

1.3. Tungsten Connector

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

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

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

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

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

The supported modes are:

- *Bridge Mode (Default)*

  Bridge Mode is the default mode of operation when no other options are supplied at install time. Bridge mode is the fastest mode of operation for the Connector. In this mode, the connector acts as a simple router connecting the network sockets between client application and MySQL servers. Bridge mode provides a simpler method for connecting clients to MySQL, but with reduced facilities.

- *Port Based Routing*

  Port based routing employs a second port on the connector host. All connections to this port are sent to an available Replica.

- *Direct Reads*

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

- *SmartScale*

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

- *Host Based Routing*

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

- *SQL Based Routing*

SQL based routing employs packet inspection to identify key strings within the query to determine where the packets should be routed.

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

For much more detailed documentation on all of the various Connector features and configurations available, see Chapter 7, *Tungsten Connector*

> Internally, Tungsten Connector supports the native MySQL protocol, and accepts the raw packet data from the client and sends those packets directly to the datasource. Because it is the native network packets that are being forwarded between hosts the performance is kept high, without requiring any additional overhead or intelligence within the application.
>
> The connector handles the distribution of packets between datasources, allowing clients to remain connected to Tungsten Connector even while the underlying datasources may become disconnected, or expanded as new datasources are added to the cluster.

Chapter 2. Deployment

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

2.1. Host Types

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

Table 2.1. Key Terminology

Tungsten Term	Traditional Term	Description
composite dataservice	Multi-Site Cluster	A configured Tungsten Cluster service consisting of multiple dataservices, typically at different physical locations.
dataservice	Cluster	A configured Tungsten Cluster service consisting of dataservers, datasources and connectors.
dataserver	Database	The database on a host. Datasources include MySQL, PostgreSQL or Oracle.
datasource	Host or Node	One member of a dataservice and the associated Tungsten components.
staging host	-	The machine from which Tungsten Cluster is installed and configured. The machine does not need to be the same as any of the existing hosts in the cluster.
staging directory	-	The directory where the installation files are located and the installer is executed. Further configuration and updates must be performed from this directory.
connector	-	A connector is a routing service that provides management for connectivity between application services and the underlying dataserver.
Active Witness host	-	A witness host is a host that runs the manager process but is not running a database server. This server will be used to establish quorum in the event that a datasource becomes unavailable

2.1.1. Manager Hosts

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

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

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

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

Within a typical Tungsten Cluster deployment there are multiple managers and these keep in constant contact with each other, and the other services. When a failure occurs, multiple managers are involved in decisions. For example, if a host is no longer visible to one manager, it does not make the decision to disable the service on it's own; only when a majority of managers identify the same result is the decision made. For this reason, there should be an odd number of managers (to prevent deadlock), or managers can be augmented through the use of witness hosts.

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

Checks to determine the availability of hosts are performed by using either the system ping protocol or the Echo TCP/IP protocol on port 7 to determine whether a host is available. The configuration of the protocol to be used can be made by adjusting the manager properties. For more information, see Section B.3.3.3, "Host Availability Checks".

2.1.2. Connector (Router) Hosts

Connectors (known as routers within the dataservice) provide a routing mechanism between client applications and the dataservice. The Tungsten Connector component automatically routes database operations to the Primary or Replica, and takes account of the current cluster

status as communicated to it by the Tungsten Manager. This functionality solves three primary issues that might normally need to be handled by the client application layer:

- Datasource role redirection (i.e. Primary and Replica). This includes read/write splitting, and the ability to read data from a Replica that is up to date with a corresponding write.
- Datasource failure (high-availability), including the ability to redirect client requests in the event of a failure or failover. This includes maintenance operations.
- Dataservice topology changes, for example when expanding the number of datasources within a dataservice

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

Depending on the deployment environment and client application requirements, the connector can be installed either on the client application servers, the database servers, or independent hosts. For more information, see Section 7.3, “Clients and Deployment”.

Connectors can also be installed independently on specific hosts. The list of enabled connectors is defined by the `--connectors` [484] option to tpm. A Tungsten Cluster dataservice can be installed with more connector servers than datasources or managers.

2.1.3. Replicator Hosts

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

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

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

Replication within Tungsten Cluster is supported by Tungsten Replicator™ and this supports a wide range of additional deployment topologies, and heterogeneous deployments including MongoDB, Vertica, and Oracle. Replication to and from a dataservice are supported. For more information on replicating out of an existing dataservice, see:

- Section 3.9, “Replicating Data Out of a Cluster”

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

2.1.4. Active Witness Hosts

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

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

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

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

Active witnesses are the solution for deployments where you have only require a two-database-node deployment.

Tungsten Cluster Quorum Requirements

- There should be at least three managers (including any active witnesses)
- There should be, in total, an odd number of managers and witnesses, to prevent deadlocks.
- If the dataservice contains only two hosts, at least one active witness must be installed.

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

The rules for witness selection are as follows:

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

To enable active witnesses, the `enable-active-witnesses=true` [490] option must be specified and the hosts that will act as active witnesses must be added to the list of hosts provided to the `members` [499] option. This enables all specified witnesses to be enabled as active witnesses:

```
[alpha]
...
master=hostA
members=hostA,hostB,hostC
enable-active-witnesses=true
witnesses=hostC
 ...
```

2.2. Deployment Sources

- Tar/GZip

  Using the TAR/GZip package creates a local directory that enables you to perform installs and updates from the extracted 'staging' directory.

- RPM Packages

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

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

```
tungsten-clustering-7.0.3-141.tar.gz
```

The version number is `7.0.3` and build number `141`. Build numbers indicate which build a particular release version is based on, and may be useful when installing patches provided by support.

2.2.1. Using the TAR/GZipped files

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

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-clustering-7.0.3-141.tar.gz
```

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

```
shell> cd tungsten-clustering-7.0.3-141
shell> tools/tpm install
```

2.2.2. Using the RPM package files

Installation

Installing the RPM package will do the following:

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

If no `tungsten.ini` was found, then after creation you will need to manually install the software, using the following steps as an example:

```
shell> cd tungsten-clustering-7.0.3-141
shell> tools/tpm install
```

Although the RPM packages complete a number of the prerequisite 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/tungsten.ini` file and then installing the package. Any output from the tpm command will go to `/opt/continuent/service_logs/rpm.output`.

> **Note**
>
> If you download the package files directly, you may need to add the signing key to your environment before the package will load properly.
>
> For yum platforms (RHEL/CentOS/Amazon Linux), the rpm command is used :
>
> ```
> root-shell> rpm --import http://www.continuent.com/RPM-GPG-KEY-continuent
> ```
>
> For Ubuntu/Debian platforms, the gpg command is used :
>
> ```
> root-shell> gpg --keyserver keyserver.ubuntu.com --recv-key 7206c924
> ```

Upgrades

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

1. Unpack the software into `/opt/continuent/software`
2. Run tools/tpm update --replace-release 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.

2.3. Common tpm Options During Deployment

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

Changes to the configuration should be made with tpm update. This continues the procedure of using tpm install during installation. See Section 10.5.23, “tpm update Command” for more information on using tpm update.

- `datasource-systemctl-service` [487]

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

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

  ```
  shell> systemctl mysql stop
  ```

  Different commands must follow the same basic structure, the command configured by `datasource-systemctl-service` [487], the service-name, and the status (i.e. `stop`).

2.4. Best Practices

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

2.4.1. Best Practices: Deployment

- Standardize the OS and database prerequisites. There are Ansible modules available for immediate use within AWS, or as a template for modifications. The product is also available to subscribe to via the AWS and GCP Marketplaces for customers without an existing annual subscription license.

More information on the Ansible method is available in this blog article.

- Ensure that the output of the `hostname`command and the nodename entries in the Tungsten configuration match exactly prior to installing Tungsten.

  The configuration keys that define nodenames are: `slaves` [514], `dataservice-slaves` [514], `members` [499], `master` [499], `dataservice-master-host` [499], `masters` [499] and `relay` [499]

- 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.3.4, "Directory Locations and Configuration".
  - When using ssh and/or SSL, ensure that the ssh key or certificates are suitably protected. See Section B.3.3.2, "SSH Configuration".
  - Use a firewall, such as iptables to protect the network ports that you need to use. The best solution is to ensure that only known hosts can connect to the required ports for Tungsten Cluster. For more information on the network ports required for Tungsten Cluster operation, see Section B.3.3.1, "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 Chapter 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 there are an even number of database servers in the cluster, configure the cluster with a witness host. See Section 2.1.4, "Active Witness Hosts" for more details on how to configure them.
- It is strongly advised to include `install=true` [494] to your configuration to ensure all components are correctly registered with your OS systemctl (or equivalent) processes, ensuring automatic restart should a host reboot.

  Alternatively, you should run deployall following installation.

- If you are using `ROW` replication, any triggers that run additional `INSERT`/`UPDATE`/`DELETE` operations must be updated so they do not run on the Replica servers.
- Make sure you know the structure of the Tungsten Cluster home directory and how to initialize your environment for administration. See Section 6.1, "The Home Directory" and Section 6.2, "Establishing the Shell Environment".
- Prior to migrating applications to Tungsten Cluster test failover and recovery procedures from Chapter 6, *Operations Guide*. Be sure to try recovering a failed Primary and reprovisioning failed Replicas.
- When deciding on the Service Name for your configurations, keep them simple and short and only use alphanumerics (Aa-Zz,0-9) and underscores (_).

2.4.2. Best Practices: Upgrade

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

- From time to time, certain changes in the product may result in switches failing between higher/lower versions, therefore the recommended best practice is to perform the upgrade in place on the all nodes without a primary switch. Providing the cluster is placed in *`MAINTENANCE`* mode there will be no disruption to the cluster operation.
- See Section 10.2.3, "Upgrades with an INI File" for more information.
- Here is the sequence of events for a proper Tungsten upgrade on a 3-node cluster:
  - Login to the Customer Downloads Portal and get the latest version of the software.
  - Copy the file (i.e. `tungsten-clustering-7.0.3-141.tar.gz`) to each host that runs a Tungsten component.
  - Set the cluster to policy *`MAINTENANCE`*
  - On every host:
    - Extract the tarball under /opt/continuent/software/ (i.e. create `/opt/continuent/software/tungsten-clustering-7.0.3-141`)
    - cd to the newly extracted directory

- Run the Tungsten Package Manager tool:

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

- For example, here are the steps in order:

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

On EVERY Tungsten host (Preferably at the same time):
shell> cd /opt/continuent/software
shell> tar xvzf tungsten-clustering-7.0.3-141.tar.gz
shell> cd tungsten-clustering-7.0.3-141

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

To perform the upgrade and delay the restart of the Connectors to a later time:
shell> tools/tpm update --replace-release --no-connectors

When it is time for the Connector to be promoted to the new version, perhaps after taking it out of the load balancer:
shell> tpm promote-connector

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

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

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

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

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

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

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

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

2.4.3. Best Practices: Operations

- Setup proper monitoring for all servers as described in Section 6.17, "Monitoring Tungsten Cluster".
- Configure the Tungsten services to startup and shutdown along with the server. See Section 4.3, "Configuring Startup on Boot".
- Schedule the Section 9.8, "The cluster_backup Command" tool on each database server at least each night. The script will take a backup of at least one server. Skip this step if you have another backup method scheduled that takes consistent snapshots of your server.

2.4.4. Best Practices: Maintenance

- Deploy an installation that matches your production enviornment and test all operations and maintenance operations there.
- Schedule regular tests for local and DR failover. This should at least include switching the Primary server to another host in the local cluster. If possible, the DR cluster should be tested once per quarter.
- Disable any automatic operating system patching processes. The use of automatic patching will cause issues when all database servers automatically restart without coordination.

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

Chapter 3. Deployment: MySQL Topologies

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

All deployments operate through the tpm command. tpm operates in two different modes:

- tpm `INI` configuration — tpm uses an `INI` to configure the service on the local host. The `INI` file must be created on each host that will be part of the cluster. tpm only manages the services on the local host; in a multi-host deployment, upgrades, updates, and configuration must be handled separately on each host.
- tpm staging configuration — a tpm configuration is created by defining the command-line arguments that define the deployment type, structure and any additional parameters. tpm then installs all the software on all the required hosts by using ssh to distribute Tungsten Cluster and the configuration, and optionally automatically starts the services on each host. tpm manages the entire deployment, configuration and upgrade procedure.

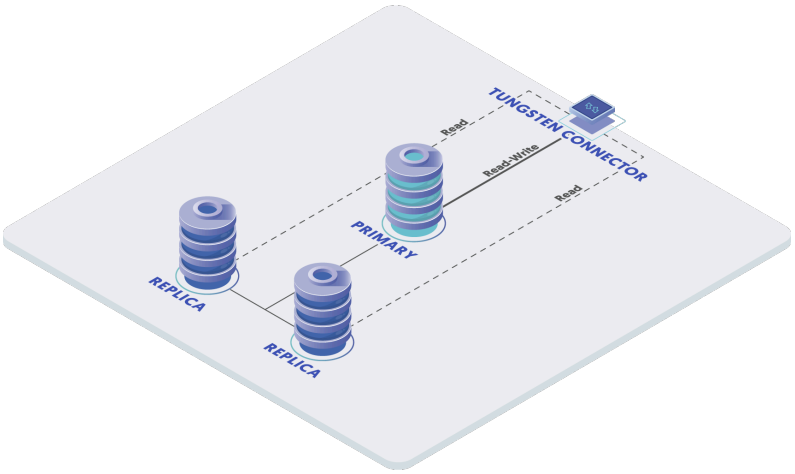
> **Important**
>
> Whilst the staging method of deployment is still available, it will be deprecated in a future release and therefore all new deployments should follow the INI method of deployment.

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

3.1. Deploying Standalone HA Clusters

Within a Primary/Replica service, there is a single Primary which replicates data to the Replicas. The Tungsten Connector handles connectivity by the application and distributes the load to the datasources in the dataservice.

Figure 3.1. Topologies: Standalone HA Cluster



3.1.1. Prepare: Standalone HA Cluster

Before continuing with the deployment you will need the following:

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

All servers must be prepared with the proper prerequisites. See Appendix B, *Prerequisites* for additional details.

3.1.2. Install: Standalone HA Cluster

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

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-clustering-7.0.3-141.tar.gz
```

2. Change to the Tungsten Cluster directory:

```
shell> cd tungsten-clustering-7.0.3-141
```

3. Create the `/etc/tungsten/tungsten.ini` using the example below as a template. Once created to suit your needs, run the tpm to perform the installation.

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

```
[defaults]
user=tungsten
install-directory=/opt/continuent
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=password
replication-port=13306
application-user=app_user
application-password=secret
application-port=3306
rest-api-admin-user=apiuser
rest-api-admin-password=secret
install=true
connector-rest-api-address=0.0.0.0
manager-rest-api-address=0.0.0.0
replicator-rest-api-address=0.0.0.0

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

Configuration group **`defaults`**

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

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

- `user=tungsten` [521]

  System User

- `install-directory=/opt/continuent` [494]

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

- `profile-script=~/.bash_profile` [507]

  Append commands to include env.sh in this profile script

- `replication-user=tungsten` [510]

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

- `replication-password=password` [510]

  The password to be used when connecting to the database using the corresponding `--replication-user` [510].

- `replication-port=13306` [510]

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

- `application-user=app_user` [474]

  Database username for the connector

- `application-password=secret` [473]

Database password for the connector

- application-port=3306 [473]

  Port for the connector to listen on

- rest-api-admin-user=apiuser [512]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- rest-api-admin-password=secret [512]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

- install=true [494]

  Install service start scripts

- connector-rest-api-address=0.0.0.0 [482]

  Address for the API to bind too.

- manager-rest-api-address=0.0.0.0 [498]

  Address for the API to bind too.

- replicator-rest-api-address=0.0.0.0 [511]

  Address for the API to bind too.

Configuration group **alpha**

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

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

- topology=clustered [520]

  Replication topology for the dataservice.

- master=host1 [499]

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

- members=host1,host2,host3 [499]

  Hostnames for the dataservice members

- connectors=host4 [484]

  Hostnames for the dataservice connectors

> **Note**
>
> If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API Access. This must be done by specifying the following options in your configuration:
>
> ```
> rest-api-admin-user=tungsten
> rest-api-admin-pass=secret
> ```
>
> For more information on using and configuring the REST API, see Section 11.1, “Getting Started with Tungsten REST API”

Run tpm to validate the installation. Providing no errors are return, you can then continue to install the software.

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

During the validation, startup and installation, tpm will notify you of any problems that need to be fixed before the service can be correctly installed and started.

4. Initialize your PATH and environment.

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

5. If you supplied `start-and-report=true` [514] the software will be running, if this wasn't included, you can now start the software as follows:

```
shell > startall
```

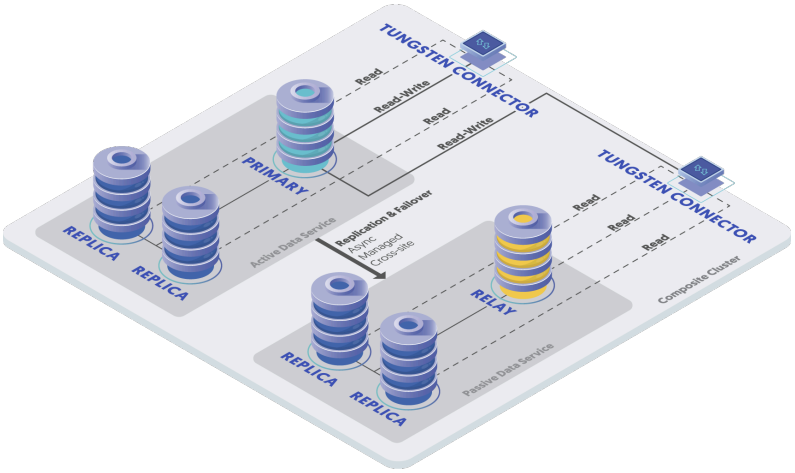
> **Important**
>
> Do not include `start-and-report` [514] if you are taking over for MySQL native replication. See Section 3.10.1, "Migrating from MySQL Native Replication 'In-Place'" for next steps after completing installation.

For Best Practices, follow the guidelines in Section 2.4, "Best Practices".

3.2. Deploying Composite Active/Passive Clustering

Tungsten Cluster supports the creation of composite clusters. This includes multiple active/passive dataservices tied together. One of the dataservices is identified as the active, containing the Primary node and all other dataservices (passive) replicate from it.

Figure 3.2. Topologies: Composite Active/Passive Cluster



3.2.1. Prepare: Composite Active/Passive Cluster

Before continuing with deployment you will need the following:

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

All servers must be prepared with the proper prerequisites. See Appendix B, *Prerequisites* for additional details.

3.2.2. Install: Composite Active/Passive Cluster

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

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-clustering-7.0.3-141.tar.gz
```

2. Change to the Tungsten Cluster directory:

```
shell> cd tungsten-clustering-7.0.3-141
```

3. Create the `/etc/tungsten/tungsten.ini` using the example below as a template. Once created to suit your needs, run the tpm to perform the installation.

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

```
[defaults]
user=tungsten
install-directory=/opt/continuent
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=secret
replication-port=13306
application-user=app_user
application-password=secret
application-port=3306
rest-api-admin-user=apiuser
rest-api-admin-password=secret
connector-rest-api-address=0.0.0.0
manager-rest-api-address=0.0.0.0
replicator-rest-api-address=0.0.0.0

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

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

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

Configuration group **defaults**

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

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

- `user=tungsten` [521]

  System User

- `install-directory=/opt/continuent` [494]

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

- `profile-script=~/.bash_profile` [507]

  Append commands to include env.sh in this profile script

- `replication-user=tungsten` [510]

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

- `replication-password=secret` [510]

  The password to be used when connecting to the database using the corresponding `--replication-user` [510].

- `replication-port=13306` [510]

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

- `application-user=app_user` [474]

  Database username for the connector

- `application-password=secret` [473]

Database password for the connector

- `application-port=3306` [473]

  Port for the connector to listen on

- `rest-api-admin-user=apiuser` [512]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `rest-api-admin-password=secret` [512]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

- `connector-rest-api-address=0.0.0.0` [482]

  Address for the API to bind too.

- `manager-rest-api-address=0.0.0.0` [498]

  Address for the API to bind too.

- `replicator-rest-api-address=0.0.0.0` [511]

  Address for the API to bind too.

Configuration group **`alpha`**

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

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

- `topology=clustered` [520]

  Replication topology for the dataservice.

- `master=host1.alpha` [499]

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

- `members=host1.alpha,host2.alpha,host3.alpha` [499]

  Hostnames for the dataservice members

- `connectors=host1.alpha,host2.alpha,host3.alpha` [484]

  Hostnames for the dataservice connectors

Configuration group **`beta`**

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

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

- `topology=clustered` [520]

  Replication topology for the dataservice.

- `relay=host1.beta` [499]

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

- `members=host1.beta,host2.beta,host3.beta` [499]

  Hostnames for the dataservice members

- `connectors=host1.beta,host2.beta,host3.beta` [484]

  Hostnames for the dataservice connectors

- `relay-source=alpha` [508]

  Dataservice name to use as a relay source

Configuration group `gamma`

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

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

- `composite-datasources=alpha,beta` [478]

  Data services that should be added to this composite data service

  > **Note**
  >
  > If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API Access. This must be done by specifying the following options in your configuration:
  >
  > ```
  > rest-api-admin-user=tungsten
  > rest-api-admin-pass=secret
  > ```

Run tpm to validate the installation. Providing no errors are return, you can then continue to install the software.

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

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

4. Initialize your `PATH` and environment.

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

The Composite Active/Passive Cluster should be installed and ready to use.

3.2.3. Adding a remote Composite Cluster

Adding an entire new cluster provides significant level of availability and capacity. The new cluster nodes that form the cluster will be fully aware of the original cluster(s) and communicate with existing managers and datasources within the cluster.

The following steps guide you through updating the configuration to include the new hosts and services you are adding.

1. On the new host(s), ensure the Appendix B, *Prerequisites* have been followed.
2. Let's assume that we have a composite cluster dataservice called `global` with two clusters, `east` and `west`, with three nodes each.

   In this worked exmple, we show how to add an additional cluster service called `north` with three new nodes.
3. Set the cluster to *`MAINTENANCE`* mode using cctrl:

   ```
   shell> cctrl
   [LOGICAL] / > use global
   [LOGICAL] /global > set policy maintenance
   ```
4. Using the following as an example, update the `/etc/tungsten/tungsten.ini` on the existing nodes to include the new cluster and update the additional composite service block. Then, copy the ini file to all the new nodes in the new cluster.

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

   ```
   [north]
   ...
   connectors=db7,db8,db9
   relay-source=east
   relay=db7
   slaves=db8,db9
   topology=clustered

   [global]
   ...
   composite-datasources=east,west,north
   ```

   Run the tpm command to update the software with the INI-based configuration:

   ```
   shell> tpm query staging
   tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141

   shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
   ```

```
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-7.0.3-141

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update --no-connectors --replace-release
```

For information about making updates when using an INI file, please see Section 10.2.4, "Configuration Changes with an INI file".

5. Using the `--no-connectors` [504] option updates the current deployment without restarting the existing connectors.
6. On all nodes in the new cluster, download and unpack the software, and install

```
shell> cd /opt/continunent/software
shell> tar zxvf tungsten-clustering-7.0.3-141.tar.gz
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> tools/tpm install
```

7. On every node in the original clusters, make sure all replicators are online:

```
shell> trepctl online; trepctl services
```

8. On all nodes in the new cluster start the software

```
shell> startall
```

9. The next steps will involve provisioning the new cluster nodes. An alternative approach to using this method would be to take a backup of a Replica from the existing cluster, and manually restoring it to ALL nodes in the new cluster PRIOR to issuing the install step above. If you take this approach then you can skip the next two re-provision steps.
10. Go to the relay (Primary) node of the new cluster (i.e. db7) and provision it from any Replica in the original cluster (i.e. db2):

```
shell> tprovision -s db2 -m xtrabackup
```

11. Go to a Replica node of the new cluster (i.e. db8) and provision it from the relay node of the new cluster (i.e. db7):

```
shell> tprovision -s db7 -m xtrabackup
```

12. Repeat the process for the renamining Replicas nodes in the new cluster.

> **Note**
>
> For a full explanation of using tprovison see Section 9.23, "The tprovision Script"

13. Set the composite cluster to *`AUTOMATIC`* mode using cctrl:

```
shell> cctrl
[LOGICAL] / > use global
[LOGICAL] /global > set policy automatic
```

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

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

3.3. Deploying Composite Active/Active Clusters

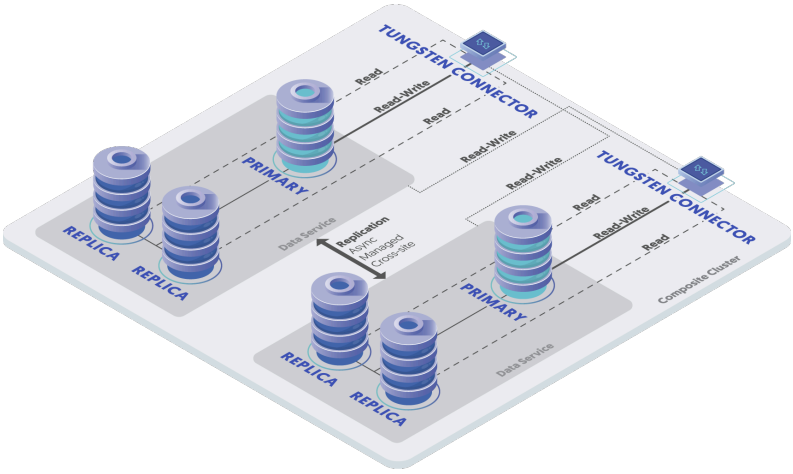
A Composite Active/Active (CAA) Cluster topology provides all the benefits of a typical dataservice at a single location, but with the benefit of also replicating the information to another site. The underlying configuration within Tungsten Cluster uses two services within each node; one provides the replication within the cluster, and the second provides replication from the remote cluster. Both are managed by the Tungsten Manager

> **Note**
>
> Composite Active/Active Clusters were previously referred to as Multi-Site/Active-Active (MSAA) clusters. The name has been updated to reflect the nature of these clusters as part of an overall active/active deployment using clusters, where the individual clusters could be in the same or different locations.
>
> Whilst the older Multi-Site/Active-Active topology is still valid and supported, it is recommended that this newer Composite Active/Active topology is adopted from version 6 of Tungsten Cluster onwards. For details on the older topology, see Section 3.5, "Deploying Multi-Site/Active-Active Clustering"

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

A sample display of how this operates is provided in Figure 3.3, "Topologies: Composite Active/Active Clusters".

Figure 3.3. Topologies: Composite Active/Active Clusters



The service can be described as follows:

- Tungsten Cluster Service: `east`

  Replicates data between `east1`, `east2` and `east3`.

- Tungsten Cluster Service: `west`

  Replicates data between `west1`, `west2` and `west3`.

- Tungsten Cluster Service: `west_from_east`

  Defines the replication service using a secondary sub-service within the cluster. This service reads THL FROM `east` and writes to the `relay` node in `west`, subsequently, the `replica` nodes within `west` are then replicated to from there.

- Tungsten Replicator Service: `east_from_west`

  Defines the replication service using a secondary sub-service within the cluster. This service reads THL FROM `west` and writes to the `relay` node in `east`, subsequently, the `replica` nodes within `east` are then replicated to from there.

A new Composite Dynamic Active/Active topology was introduced from version 7.0.0 of Tungsten Cluster

Composite Dynamic Active/Active builds on the foundation of the Composite Active/Active topology and the cluster continues to operate and be configured in the same way.

The difference is, with Composite Dynamic Active/Active, the cluster instructs the Proxy layer to behave like a Composite Active/Passive cluster.

For more information on this topology and how to enable it, see Section 3.4, “Deploying Composite Dynamic Active/Active”

3.3.1. Prepare: Composite Active/Active Clusters

Some considerations must be taken into account for any active/active scenarios:

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

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

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

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

  ```
  binlog-format = row
  ```

- Beware of triggers. Triggers can cause problems during replication because if they are applied on the Replica as well as the Primary you can get data corruption and invalid data. Tungsten Cluster cannot prevent triggers from executing on a Replica, and in an active/active

topology there is no sensible way to disable triggers. Instead, check at the trigger level whether you are executing on a Primary or Replica. For more information, see Section C.4.1, “Triggers”.

3.3.2. Install: Composite Active/Active Clusters

Deployment of Composite Active/Active clusters is only supported using the INI method of deployment.

Configuration and deployment of the cluster works as follows:

- Creates two basic Primary/Replica clusters.
- Creates a composite service that includes the Primary/Replica clusters within the definition.

The resulting configuration within the example builds the following deployment:

- One cluster, `east`, with three hosts.
- One cluster, `west`, with three hosts.
- All six hosts in the two clusters will have a manager, replicator and connector installed.
- Each replicator has two replication services, one service that replicates the data within the cluster. The second service, replicates data from the other cluster to this host.

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

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

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

```
[defaults]
user=tungsten
install-directory=/opt/continuent
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=secret
replication-port=13306
application-user=app_user
application-password=secret
application-port=3306
rest-api-admin-user=apiuser
rest-api-admin-password=secret
connector-rest-api-address=0.0.0.0
manager-rest-api-address=0.0.0.0
replicator-rest-api-address=0.0.0.0

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

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

[global]
topology=composite-multi-master
composite-datasources=east,west
```

Configuration group **`defaults`**

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

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

- `user=tungsten` [521]

  System User

- `install-directory=/opt/continuent` [494]

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

- `profile-script=~/.bash_profile` [507]

  Append commands to include env.sh in this profile script

- `replication-user=tungsten` [510]

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

- `replication-password=secret` [510]

  The password to be used when connecting to the database using the corresponding `--replication-user` [510].

- `replication-port=13306` [510]

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

- `application-user=app_user` [474]

  Database username for the connector

- `application-password=secret` [473]

  Database password for the connector

- `application-port=3306` [473]

  Port for the connector to listen on

- `rest-api-admin-user=apiuser` [512]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `rest-api-admin-password=secret` [512]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

- `connector-rest-api-address=0.0.0.0` [482]

  Address for the API to bind too.

- `manager-rest-api-address=0.0.0.0` [498]

  Address for the API to bind too.

- `replicator-rest-api-address=0.0.0.0` [511]

  Address for the API to bind too.

Configuration group **`east`**

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

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

- `topology=clustered` [520]

  Replication topology for the dataservice.

- `master=east1` [499]

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

- `members=east1,east2,east3` [499]

  Hostnames for the dataservice members

- `connectors=east1,east2,east3` [484]

  Hostnames for the dataservice connectors

Configuration group **`west`**

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

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

- `topology=clustered` [520]

  Replication topology for the dataservice.

- `master=west1` [499]

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

- `members=west1,west2,west3` [499]

  Hostnames for the dataservice members

- `connectors=west1,west2,west3` [484]

  Hostnames for the dataservice connectors

Configuration group **`global`**

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

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

- `topology=composite-multi-master` [520]

  Replication topology for the dataservice.

- `composite-datasources=east,west` [478]

  Data services that should be added to this composite data service

The configuration above defines two clusters, `east` and `west`, which are both part of a composite cluster service, `global`.

> **Note**
>
> If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API Access. This must be done by specifying the following options in your configuration:
>
> ```
> rest-api-admin-user=tungsten
> rest-api-admin-pass=secret
> ```

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

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

- A parent composite service, `global` in this example, with child services as listed, `east` and `west`.
- Replication services between each child service, using the service name `a_from_b`, for example, `east_from_west` and `west_from_east`.

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

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

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

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

   ```
   shell> cd /opt/continuent/software
   shell> tar zxf tungsten-clustering-7.0.3-141.tar.gz
   ```

   b. Change to the Tungsten Cluster staging directory:

```
shell> cd tungsten-clustering-7.0.3-141
```

c. Run tpm to install the software:

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

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

If you included the `start-and-report` [514] option in your configuration, the software will be started for you. If you excluded this option then you will need to manually start the software by continuing with the steps below.

3. Initialize your `PATH` and environment:

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

4. Start the software, if not already running:

```
shell> startall
```

The Composite Active/Active clustering should be installed and ready to use.

3.3.3. Best Practices: Composite Active/Active Clusters

> **Note**
>
> In addition to this information, follow the guidelines in Section 2.4, “Best Practices”.

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

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

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

```
Tungsten Clustering 7.0.3 build 141
east: session established, encryption=false, authentication=false
jgroups: unencrypted, database: unencrypted
[LOGICAL] / > ls
global
  east
    east_from_west
  west
    west_from_east
```

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

```
[LOGICAL] / > use global
[LOGICAL] /global > ls

COORDINATOR[east1:AUTOMATIC:ONLINE]
   east:COORDINATOR[east1:AUTOMATIC:ONLINE]
   west:COORDINATOR[west2:AUTOMATIC:ONLINE]

ROUTERS:
+------------------------------------------------------------------------------+
|connector@east1[8925](ONLINE, created=0, active=0)                            |
|connector@east2[8770](ONLINE, created=0, active=0)                            |
|connector@east3[8801](ONLINE, created=0, active=0)                            |
|connector@west1[8801](ONLINE, created=0, active=0)                            |
|connector@west2[8770](ONLINE, created=0, active=0)                            |
|connector@west3[8768](ONLINE, created=0, active=0)                            |
+------------------------------------------------------------------------------+

DATASOURCES:
+------------------------------------------------------------------------------+
|east(composite master:ONLINE, global progress=12, max latency=0.995)         |
|STATUS [OK] [2025/01/07 11:30:23 AM UTC]                                      |
+------------------------------------------------------------------------------+
|  east(master:ONLINE, progress=10, max latency=0.229)                         |
|  east_from_west(relay:ONLINE, progress=2, max latency=0.995)                 |
```

```
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|west(composite master:ONLINE, global progress=12, max latency=0.988)       |
|STATUS [OK] [2025/01/07 11:30:27 AM UTC]                                    |
+----------------------------------------------------------------------------+
|  west(master:ONLINE, progress=2, max latency=0.988)                        |
|  west_from_east(relay:ONLINE, progress=10, max latency=0.260)              |
+----------------------------------------------------------------------------+
```

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

- `|east(composite master:ONLINE, global progress=12, max latency=0.995)            |`

  This line indicates:

  - The name and type of the composite cluster, and whether the Primary in the cluster is online.
  - The global progress. This is a counter that combines the local progress of the cluster, and the replication of data from this cluster to the remote clusters in the composite to this cluster. For example, if data is inserted into `west`
  - The maximum latency within the cluster.

- `|STATUS [OK] [2025/01/07 11:30:23 AM UTC]                                    |`

  The status and date within the Primary of the cluster.

- `|  east(master:ONLINE, progress=10, max latency=0.229)                        |`

  The status and progress of the cluster.

- `|  east_from_west(relay:ONLINE, progress=2, max latency=0.995)                |`

  The status and progress of remote replication from the cluster.

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

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

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

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

ROUTERS:
+----------------------------------------------------------------------------+
|connector@east1[8925](ONLINE, created=0, active=0)                          |
|connector@east2[8770](ONLINE, created=0, active=0)                          |
|connector@east3[8801](ONLINE, created=0, active=0)                          |
|connector@west1[8801](ONLINE, created=0, active=0)                          |
|connector@west2[8770](ONLINE, created=0, active=0)                          |
|connector@west3[8768](ONLINE, created=0, active=0)                          |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|east1(master:ONLINE, progress=10, THL latency=0.134)                        |
|STATUS [OK] [2025/01/07 01:19:12 PM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|east2(slave:ONLINE, progress=10, latency=0.229)                             |
|STATUS [OK] [2025/01/07 11:30:26 AM UTC]                                    |
+----------------------------------------------------------------------------+
```

```
|  MANAGER(state=ONLINE)                                                          |
|  REPLICATOR(role=slave, master=east1, state=ONLINE)                             |
|  DATASERVER(state=ONLINE)                                                       |
|  CONNECTIONS(created=0, active=0)                                               |
+---------------------------------------------------------------------------------+
+---------------------------------------------------------------------------------+
|east3(slave:ONLINE, progress=10, latency=0.159)                                  |
|STATUS [OK] [2025/01/07 01:19:18 PM UTC]                                         |
+---------------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                          |
|  REPLICATOR(role=slave, master=east1, state=ONLINE)                             |
|  DATASERVER(state=ONLINE)                                                       |
|  CONNECTIONS(created=0, active=0)                                               |
+---------------------------------------------------------------------------------+
```

Within each cluster, cctrl can be used to monitor the current status. For more information on checking the status and controlling operations, see Section 6.3, “Checking Dataservice Status”.

To monitor all services and the current status, you can also use the multi_trepctl command. This generates a unified status report for all the hosts and services configured:

```
shell> multi_trepctl --by-service
| host  | servicename    | role   | state  | appliedlastseqno | appliedlatency |
| east1 | east           | master | ONLINE |                5 |          0.440 |
| east2 | east           | slave  | ONLINE |                5 |          0.538 |
| east3 | east           | slave  | ONLINE |                5 |          0.517 |
| east1 | east_from_west | relay  | ONLINE |               23 |          0.074 |
| east2 | east_from_west | slave  | ONLINE |               23 |          0.131 |
| east3 | east_from_west | slave  | ONLINE |               23 |          0.111 |
| west1 | west           | master | ONLINE |               23 |          0.021 |
| west2 | west           | slave  | ONLINE |               23 |          0.059 |
| west3 | west           | slave  | ONLINE |               23 |          0.089 |
| west1 | west_from_east | relay  | ONLINE |                5 |          0.583 |
| west2 | west_from_east | slave  | ONLINE |                5 |          0.562 |
| west3 | west_from_east | slave  | ONLINE |                5 |          0.592 |
```

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

> **Note**
>
> For the multi_trepctl command to work, you will need to have enabled passwordless ssh between all nodes within the cluster.

To monitor individual servers and/or services, use trepctl, using the correct servicename. For example, on `east1` to check the status of the replicator within the Tungsten Cluster service, use the trepctl services command to get the status of both the local and cross-cluster services:

```
shell> trepctl status
Processing services command...
NAME              VALUE
----              -----
appliedLastSeqno: 10
appliedLatency  : 0.134
role            : master
serviceName     : east
serviceType     : local
started         : true
state           : ONLINE
NAME              VALUE
----              -----
appliedLastSeqno: 2
appliedLatency  : 0.987
role            : relay
serviceName     : east_from_west
serviceType     : local
started         : true
state           : ONLINE
Finished services command...
```

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

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

3.3.4. Resetting a single dataservice

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

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

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

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

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

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

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

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

```
shell west> trepctl -service west_from_east offline
shell west> trepctl -service west_from_east reset -all -y
```

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

```
shell east> trepctl -service east offline
shell east> trepctl -service east reset -all -y
```

5. Restore a backup on each host (`east{1,2,3}`) in the failed `east` service from a host in the `west` service, first on the host that will be the Primary node within the east cluster, for example `east1`:

```
shell east> tprovision -s west1 -m xtrabackup -c
```

When complete, proceed to restore the remaining hosts within `east` using the newly restored `east1` as the source:

```
shell east> tprovision -s east1 -m xtrabackup
```

> **Note**
>
> For a full explanation of using tprovison see Section 9.23, “The tprovision Script”

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

```
shell west> trepctl -service west_from_east online
```

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

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

8. Set the policy back to *`AUTOMATIC`*:

```
cctrl> set policy automatic
```

3.3.5. Resetting all dataservices

To reset all of the dataservices:

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

Place both clusters into *`MAINTENANCE`* mode:

```
shell> cctrl
[LOGICAL] /east > use global
[LOGICAL] /global > set policy maintenance
```

2. On all nodes in both clusters:

```
shell> trepctl -all-services offline

shell> trepctl -all-services reset -all -y

shell> trepctl -all-services online
```

3. Return both clusters to *`AUTOMATIC`* mode:

```
shell> cctrl
[LOGICAL] /east > use global
[LOGICAL] /global > set policy maintenance
```

3.3.6. Dataserver maintenance

3.3.6.1. Fixing Replication Errors

In the event of a replication fault, the standard cctrl, trepctl and other utility commands in Chapter 9, *Command-line Tools* can be used to bring the dataservice back into operation. All the tools are safe to use.

If you have to perform any updates or modifications to the stored MySQL data, ensure binary logging has been disabled using:

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

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

3.3.6.1.1. Recovering Cross Site Services

In a Composite Active/Active topology, a switch or a failover not only promotes a Replica to be a new Primary, but also will require the ability to reconfigure cross-site communications. This process therefore assumes that cross-site communication is online and working. In some situations, it may be possible that cross-site communication is down, or for some reason cross-site replication is in an `OFFLINE:ERROR` [145] state - for example a DDL or DML statement that worked in the local cluster may have failed to apply in the remote.

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

The guide explains how to recover from this situation.

- The examples are based on a 2-cluster topology, named `NYC` and `LONDON` and the composite dataservice named `GLOBAL`.
- The cluster is configured with the following dataservers:
  - `NYC : db1 (Primary), db2 (Replica), db3 (Replica)`
  - `LONDON: db4 (Primary), db5 (Replica), db6 (Replica)`
- The cross site replicators in both clusters are in an `OFFLINE:ERROR` [145] state due to failing DDL.
- A switch was then issued, promoting db3 as the new Primary in NYC and db5 as the new Primary in `LONDON`

When the cluster enters a state where the cross-site services are in an error, output from cctrl will look like the following:

```
shell> cctrl -expert -multi
[LOGICAL:EXPERT] / > use london_from_nyc
london_from_nyc: session established, encryption=false, authentication=false
[LOGICAL:EXPERT] /london_from_nyc > ls
COORDINATOR[db6:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@db1[26248](ONLINE, created=0, active=0)                           |
|connector@db2[14906](ONLINE, created=0, active=0)                           |
|connector@db3[15035](ONLINE, created=0, active=0)                           |
|connector@db4[27813](ONLINE, created=0, active=0)                           |
|connector@db5[4379](ONLINE, created=0, active=0)                            |
|connector@db6[2098](ONLINE, created=0, active=0)                            |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|db5(relay:SHUNNED(SUBSERVICE-SWITCH-FAILED), progress=6, latency=0.219)     |
|STATUS [SHUNNED] [2025/01/07 10:27:24 AM UTC]                               |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=relay, master=db3, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db4(slave:SHUNNED(SUBSERVICE-SWITCH-FAILED), progress=6, latency=0.252)     |
|STATUS [SHUNNED] [2025/01/07 10:27:25 AM UTC]                               |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db5, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db6(slave:SHUNNED(SUBSERVICE-SWITCH-FAILED), progress=6, latency=0.279)     |
```

```
|STATUS [SHUNNED] [2025/01/07 10:27:25 AM UTC]                                  |
+-------------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                        |
|  REPLICATOR(role=slave, master=db4, state=ONLINE)                             |
|  DATASERVER(state=ONLINE)                                                     |
|  CONNECTIONS(created=0, active=0)                                             |
+-------------------------------------------------------------------------------+
```

In the above example, you can see that all services are in the `SHUNNED(SUBSERVICE-SWITCH-FAILED)` state, and partial reconfiguration has happened.

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

1. The first step is to ensure the initial replication errors have been resolved and that the replicators are in an online state, the steps to resolve the replicators will depend on the reason for the error, for further guidance on resolving these issues, see Chapter 6, *Operations Guide*.

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

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

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

   ```
   cctrl> use london_from_nyc
   ```

4. Next, place the service into *MAINTENANCE* Mode

   ```
   cctrl> set policy maintenance
   ```

5. Enable override of commands issued

   ```
   cctrl> set force true
   ```

6. Bring the relay datasource online

   ```
   cctrl> datasource db5 online
   ```

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

   ```
   cctrl> replicator db5 offline
   ```

8. Change the source of the relay replicator

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

9. Bring the replicator online

   ```
   cctrl> replicator db5 online
   ```

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

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

    For example:

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

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

    ```
    cctrl> cluster welcome
    ```

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

    ```
    cctrl> datasource datasource online
    ```

    For example:

    ```
    cctrl> datasource db4 online
    ```

13. Once all nodes are online, we can then return the cluster to *AUTOMATIC*

    ```
    cctrl> set policy automatic
    ```

14. Repeat this process for the other cross-site subservice if required

3.3.7. Adding a Cluster to a Composite Active/Active Topology

This procedure explains how to add additional clusters to an existing Composite Active/Active configuration.

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

3.3.7.1. Prerequisites

Ensure the new nodes have all the necessary prerequisites in place, specifically paying attention to the following:

- MySQL auto_increment parameters set appropriately on existing and new clusters
- All new nodes have full connectivity to the existing nodes and the hosts file contains correct hostnames
- All existing nodes have full connectivity to the new nodes and hosts file contains correct hostnames

3.3.7.2. Backup and Restore

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

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

```
db6-shell> cctrl
cctrl> datasource db6 shun
cctrl> replicator db6 offline
cctrl> exit
db6-shell> stopall
db6-shell> sudo service mysqld stop
```

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

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

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

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

5. Start mysql on the new nodes

3.3.7.3. Update Existing Configuration

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

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

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

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

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

3.3.7.4. New Host Configuration

To avoid any differences in configuration, once the changes have been made to the tungsten.ini on the existing hosts, copy this file from one of the nodes to all the nodes in the new cluster.

Ensure `start-and-report` [514] is `false` or not set in the config.

3.3.7.5. Install on new nodes

On the 3 new nodes, validate the software:

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> tools/tpm validate
```

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

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

3.3.7.6. Update existing nodes

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

1. Put entire cluster into *MAINTENANCE*

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

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

   ```
   shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
   shell> tools/tpm update --replace-release
   ```

3.3.7.7. Start the new cluster

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

```
shell> startall
```

3.3.7.8. Validate and check

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

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

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

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

Place entire cluster back into *AUTOMATIC*

```
shell> cctrl
cctrl> use {composite-dataservice}
cctrl> set policy automatic
cctrl> ls
COORDINATOR[db2:AUTOMATIC:ONLINE]
   london:COORDINATOR[db5:AUTOMATIC:ONLINE]
   nyc:COORDINATOR[db2:AUTOMATIC:ONLINE]
   tokyo:COORDINATOR[db8:AUTOMATIC:ONLINE]
cctrl> exit
```

3.4. Deploying Composite Dynamic Active/Active

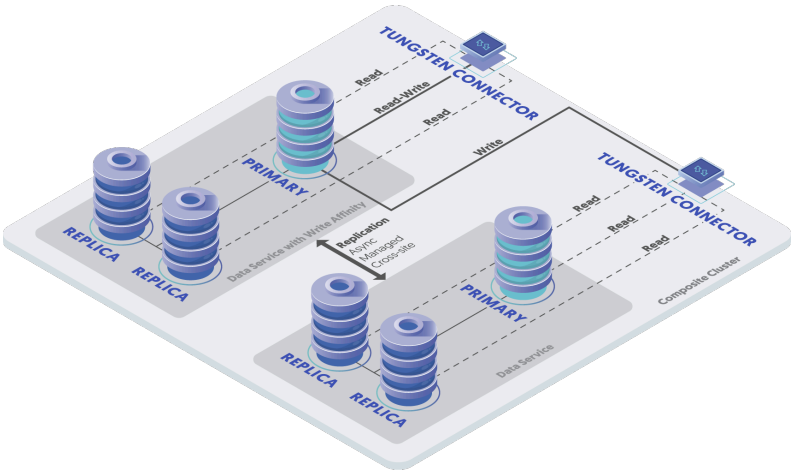
Composite Dynamic Active/Active builds on the foundation of the Composite Active/Active topology and the cluster continues to operate and be configured in the same way.

The difference is, with Composite Dynamic Active/Active, the cluster instructs the Proxy layer to behave like a Composite Active/Passive cluster.

Within your configuration you specify write affinity to a single cluster, meaning that all reads will continue to balance between local replicas, but all writes will be directed to only one cluster.

The diagram below shows how a Composite Dynamic Active/Active would behave in a typical 2-cluster configuration.

Figure 3.4. Topologies: Composite Dynamic Active/Active Clusters



The benefit of a Composite Dynamic Active/Active cluster and being able to direct writes to only one cluster, avoids all the inherent risks of a true Active/Active deployment, such as conflicts when the same row is altered in both clusters.

This is especially useful for deployments that do not have the ability to avoid potential conflicts programmatically.

The additional benefit this topology offers is instant failover of writes in the event of a cluster failure. In Composite Dynamic Active/Active if the cluster with write affinity fails, writes instantly failover to the other cluster, and because that cluster is open for writes, applications will continue uninterrupted. This differs from a Composite Active/Passive where in the event of a cluster failure there needs to be a manual failover process to re-route write operations.

3.4.1. Enabling Composite Dynamic Active/Active

To use Composite Dynamic Active/Active you need to have a Composite Active/Active cluster deployed, then it is simply a case of specifying the required affinity within the connectors.

For the purpose of this example we will assume we have two clusters `alpha` and `beta`. Each cluster will have two connectors and it is desired that the `alpha` cluster be the primary write destination.

Within the configuration for the connectors, add the following:

```
=> On alpha nodes:
connector-write-affinity=alpha
connector-read-affinity=alpha

=> On beta nodes:
connector-write-affinity=alpha
connector-read-affinity=beta
```

This will have the effect of setting the write affinity to the `alpha` cluster primarily on both `alpha` and `beta` clusters as follows:

- `alpha` cluster will get both read and write affinity to `alpha`
- `beta` cluster will get write affinity to `alpha`, but maintain read affinity to `beta`

After recovering a failed site

As outlined above, if the site that has write affinity fails, read-write traffic will failover to another site based on the affinity rules configured. Following recovery of the site that is configured as the primary write site, new connections will follow the write affinity rules, whereas existing connections will remain on the site that was promoted after failover.

To maintain data-integrity and to ensure writes continue to only be directed to a single site, it is therefore essential to also enable the following tpm property:

```
connector-reset-when-affinity-back=true
```

With this enabled, following recovery of the primary write site, all connections (new and old) will revert to the original, intended, cluster configured with primary write affinity.

Testing DAA in Bridge Mode

When using Bridge mode (the default at install), all requests are routed to the Primary by default. To test query routing, run the following query when connected through the Connector:

```
Route to the Primary:
mysql> select @@hostname;
```

In Bridge mode, the only way to verify that reads are being directed to replicas is to establish a read-only port and execute queries through it to force the QoS `RO_RELAXED`.

First, ensure that your INI file has the following option, then run tpm update

```
connector-readonly-listen-port=3307
```

To test, ensure you connect to the specified read-only port:

```
Route to a Replica:
shell> mysql -h... -P3307
mysql> select @@hostname;
```

Testing DAA in Proxy Mode with No R/W Splitting Enabled

To test Connector query routing in Proxy mode, you may use the URL-based R/W splitting to test query routing:

```
Route to the Primary:
shell> mysql -h... -Dtest@qos=RW_STRICT -e "select @@hostname;"

Route to a Replica:
shell> mysql -h... -Dtest@qos=RO_RELAXED -e "select @@hostname;"
```

Testing DAA in Proxy Mode with R/W Splitting Enabled (SmartScale or @direct)

To test Connector query routing in Proxy mode when either SmartScale or @direct read/write splitting has been enabled, you may use the following:

```
Route to the Primary:
mysql> select @@hostname for update;

Route to a Replica:
mysql> select @@hostname;
```

Manual Site-Level Switch

For DAA to work properly, all writes must go to one cluster or another, no exceptions. When you want to move all writes to another site/cluster (like you would in a Composite Active/Passive cluster using the switch command at the composite level), there is no switch command available in Dynamic Active/Active.

As of version 7.0.2, we strongly recommend that you use the cctrl command datasource SERVICE drain [optional timeout in seconds] at the composite level to shun the currently selected Active cluster. This will allow the Connector to finish (drain) all in-flight queries, shun the composite dataservice once fully drained, and then move all writes to another cluster.

Please note that this is different than using the cctrl command datasource SERVICE shun (available prior to version 7.0.2) at the composite level to shun the currently selected Active cluster. Using `shun` instead of `drain` will force the Connector to immediately sever/terminate all in-flight queries, then move all writes to another cluster.

```
shell> cctrl -multi
Tungsten Clustering 7.0.3 build 141
beta: session established, encryption=true, authentication=true
jgroups: encrypted, database: encrypted
[LOGICAL] / > use world
[LOGICAL] /world > datasource alpha drain 30

WARNING: This is an expert-level command:
Incorrect use may cause data corruption
or make the cluster unavailable.

Do you want to continue? (y/n)> y
composite data source 'alpha' is now SHUNNED
[LOGICAL] /world > exit
Exiting...
```

When you are ready to resume writes to the originally-configured site, use the composite-level cctrl command datasource SERVICE welcome. If you have set `connector-reset-when-affinity-back=true` [482], then writes will move back to the original site. If set to `false`, the writes will stay where they are.

```
shell> cctrl -multi
Tungsten Clustering 7.0.3 build 141Tungsten Clustering 7.0.3 build 141
beta: session established, encryption=true, authentication=true
jgroups: encrypted, database: encrypted
```

```
[LOGICAL] / > use world
[LOGICAL] / > datasource alpha welcome

WARNING: This is an expert-level command:
Incorrect use may cause data corruption
or make the cluster unavailable.

Do you want to continue? (y/n)> y
composite data source 'alpha' is now ONLINE
[LOGICAL] /world > exit
Exiting...
```

For more information about the datasource shun command, please visit: Section 9.1.3.5.9, “datasource shun Command”

For more information about the datasource drain command, please visit: Section 9.1.3.5.3, “datasource drain Command”

3.5. Deploying Multi-Site/Active-Active Clustering

> **Note**
>
> The procedures in this section are designed for the Multi-Site/Active-Active topology ONLY. Do NOT use these procedures for Composite Active/Active Clustering using v6 onwards.
>
> For version 6.x onwards, Composite Active/Active Clustering, please refer to Section 3.3, “Deploying Composite Active/Active Clusters”

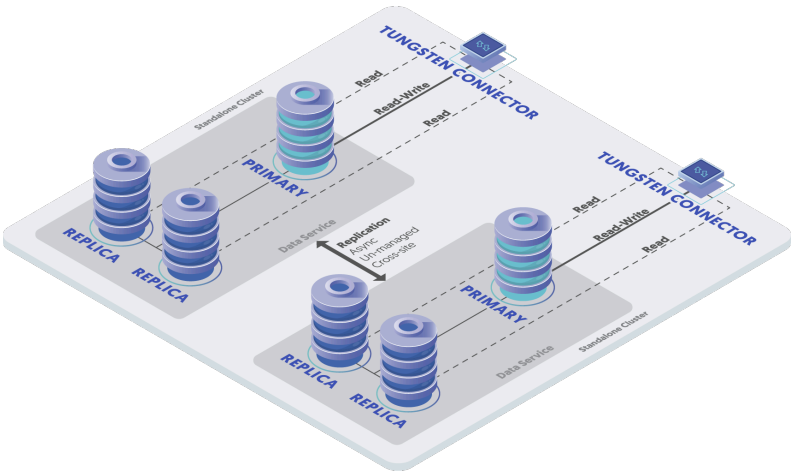
A Multi-Site/Active-Active topology provides all the benefits of a typical dataservice at a single location, but with the benefit of also replicating the information to another site. The underlying configuration within Tungsten Cluster uses the Tungsten Replicator which enables operation between the two sites.

The configuration is in two separate parts:

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

A sample display of how this operates is provided in Figure 3.5, “Topologies: Multi-Site/Active-Active Clusters”.

Figure 3.5. Topologies: Multi-Site/Active-Active Clusters



Using an example of two clusters called `east` and `west`, each containing three database nodes, the service can be described as follows:

- Tungsten Cluster Service: east

  Replicates data between `east1`, `east2`, and `east3`.

- Tungsten Cluster Service: west

  Replicates data between `westdb1`, `westdb2`, and `westdb3`.

- Tungsten Replicator Service: east

Defines the replication of data within east as a replicator service using Tungsten Replicator. This service reads from all the hosts within the Tungsten Cluster service `east` and writes to `westdb1`, `westdb2`, and `westdb3`. The service name is the same to ensure that we do not duplicate writes from the clustered service already running.

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

- Tungsten Replicator Service: west

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

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

- Tungsten Replicator Service: east_west

  Replicates data from `east` to `west`, using Tungsten Replicator. This is a service alias that defines the reading from the dataservice (as an Extractor) to other servers within the destination cluster.

- Tungsten Replicator Service: west_east

  Replicates data from `west` to `east`, using Tungsten Replicator. This is a service alias that defines the reading from the dataservice (as an Extractor) to other servers within the destination cluster.

> **Note**
>
> Although this topology is valid and can be installed with any version of tungsten, it is strongly advised that Composite Active/Active be used in favour in version 6.x or later as this provide additional benefits, especially at the connector layer.

3.5.1. Prepare: Multi-Site/Active-Active Clusters

Some considerations must be taken into account for any active/active scenario:

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

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

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

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

  ```
  binlog-format = row
  ```

- Beware of triggers. Triggers can cause problems during replication because if they are applied on the Replica as well as the Primary you can get data corruption and invalid data. Tungsten Cluster cannot prevent triggers from executing on a Replica, and in an active/active topology there is no sensible way to disable triggers. Instead, check at the trigger level whether you are executing on a Primary or Replica. For more information, see Section C.4.1, “Triggers”.

3.5.2. Install: Multi-Site/Active-Active Clusters

> **Note**
>
> The procedures in this section are designed for the Multi-Site/Active-Active topology ONLY. Do NOT use these procedures for Composite Active/Active Clustering using v6 onwards.
>
> For version 6.x onwards, Composite Active/Active Clustering, please refer to Section 3.3, “Deploying Composite Active/Active Clusters”

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

1. Download and extract the Tungsten Cluster and Tungsten Replicator :

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

```
shell> tar zxf tungsten-clustering-7.0.3-141.tar.gz
shell> tar zxf tungsten-replicator-7.0.3-141.tar.gz
```

2. Change to the Tungsten Cluster directory:

```
shell> cd tungsten-clustering-7.0.3-141
```

3. Create the /etc/tungsten/tungsten.ini> using the example below as a template. Once created to suit your needs, run the tpm to perform the installation.

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

```
[defaults]
user=tungsten
install-directory=/opt/continuent
replication-user=tungsten
replication-password=secret
replication-port=13306
profile-script=~/.bashrc
application-user=app_user
application-password=secret
application-port=3306
skip-validation-check=MySQLPermissionsCheck
rest-api-admin-user=apiuser
rest-api-admin-password=secret
connector-rest-api-address=0.0.0.0
manager-rest-api-address=0.0.0.0
replicator-rest-api-address=0.0.0.0

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

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

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

[east_west]
topology=cluster-slave
master-dataservice=east
slave-dataservice=west
thl-port=2113

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

Configuration group **defaults**

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

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

- user=tungsten [521]

  System User

- install-directory=/opt/continuent [494]

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

- replication-user=tungsten [510]

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

- replication-password=secret [510]

The password to be used when connecting to the database using the corresponding `--replication-user` [510].

- `replication-port=13306` [510]

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

- `profile-script=~/.bashrc` [507]

  Append commands to include env.sh in this profile script

- `application-user=app_user` [474]

  Database username for the connector

- `application-password=secret` [473]

  Database password for the connector

- `application-port=3306` [473]

  Port for the connector to listen on

- `skip-validation-check=MySQLPermissionsCheck` [444]

  The `skip-validation-check` [444] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

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

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

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

  The check in this case is `MySQLDefaultTableTypeCheck` [455], and could be ignored using `skip-validation-check=MySQLDefaultTableTypeCheck` [444].

  Values can be passed as a comma-separated list, or single `skip-validation-check` [444] entries for each check to be skipped.

  Setting both `skip-validation-check` [444] and `enable-validation-check` [442] is equivalent to explicitly disabling the specified check.

- `rest-api-admin-user=apiuser` [512]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `rest-api-admin-password=secret` [512]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

- `connector-rest-api-address=0.0.0.0` [482]

  Address for the API to bind too.

- `manager-rest-api-address=0.0.0.0` [498]

  Address for the API to bind too.

- `replicator-rest-api-address=0.0.0.0` [511]

  Address for the API to bind too.

Configuration group **`defaults.replicator`**

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

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

- `home-directory=/opt/replicator` [494]

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

- `rmi-port=10002` [512]

  Replication RMI listen port

- `executable-prefix=mm` [493]

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

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

Configuration group **`east`**

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

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

- `topology=clustered` [520]

  Replication topology for the dataservice.

- `connectors=east1,east2,east3` [484]

  Hostnames for the dataservice connectors

- `master=east1` [499]

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

- `members=east1,east2,east3` [499]

  Hostnames for the dataservice members

Configuration group **`west`**

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

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

- `topology=clustered` [520]

  Replication topology for the dataservice.

- `connectors=west1,west2,west3` [484]

  Hostnames for the dataservice connectors

- `master=west1` [499]

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

- `members=west1,west2,west3` [499]

  Hostnames for the dataservice members

Configuration group **`east_west`**

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

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

- `topology=cluster-slave` [520]

  Replication topology for the dataservice.

- `master-dataservice=east` [508]

  Dataservice name to use as a relay source

- `slave-dataservice=west` [518]

  Dataservice to use to determine the value of host configuration

- `thl-port=2113` [520]

  Port to use for THL Operations

Configuration group **`west_east`**

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

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

- `topology=cluster-slave` [520]

  Replication topology for the dataservice.

- `master-dataservice=west` [508]

  Dataservice name to use as a relay source

- `slave-dataservice=east` [518]

  Dataservice to use to determine the value of host configuration

- `thl-port=2115` [520]

  Port to use for THL Operations

> **Note**
>
> If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API Access. This must be done by specifying the following options in your configuration:
>
> ```
> rest-api-admin-user=tungsten
> rest-api-admin-pass=secret
> ```

Run tpm to validate the installation. Providing no errors are return, you can then continue to install the software.

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

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

4. Change to the Tungsten Replicator directory:

   ```
   shell> cd tungsten-replicator-7.0.3-141
   ```

5. Run tpm to validate the replicator installation. Providing no errors are return, you can then continue to install the software.

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

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

6. Initialize your `PATH` and environment.

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

The Multi-Site/Active-Active clustering should be installed and ready to use.

3.5.3. Best Practices: Multi-Site/Active-Active Clusters

> **Note**
>
> The procedures in this section are designed for the Multi-Site/Active-Active topology ONLY. Do NOT use these procedures for Composite Active/Active Clustering uing v6 onwards.

For version 6.x onwards, Composite Active/Active Clustering, please refer to Section 3.3, "Deploying Composite Active/Active Clusters"

Note

In addition to this information, follow the guidelines in Section 2.4, "Best Practices".

- Running a Multi-Site/Active-Active service uses many different components to keep data updated on all servers. Monitoring the dataservice is divided into monitoring the two different clusters. Be mindful when using commands that you have the correct path. You should either use the full path to the command under `/opt/continuent` and `/opt/replicator`, or use the aliases created by setting the `--executable-prefix=mm` [493] option.

  For example, calling trepctl status will execute the command under `/opt/continuent` and return the status of the replicator associated with the local cluster service.

  Calling mm_trepctl status will execut ethe command under `/opt/replicator` and return the status of the replictor handling the replication between the local and remote cluster often referred to as the cross-site replicator.

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

Using cctrl gives you access to the cluster for the `east` and `west` dataservices depending on the host that the command is executed from. For example, issuing cctrl on any node within the `east` dataservice followed by the ls, would present the status output similar to the example below:

```
Continuent Tungsten 7.0.3 build 141
east: session established
[LOGICAL] /east > ls

COORDINATOR[east3:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@east1[6117](ONLINE, created=1, active=0)                          |
|connector@east2[6105](ONLINE, created=0, active=0)                          |
|connector@east3[6109](ONLINE, created=0, active=0)                          |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|east1(master:ONLINE, progress=213, THL latency=0.921)                       |
|STATUS [OK] [2025/01/02 03:31:31 PM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=1, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|east2(slave:ONLINE, progress=213, latency=0.950)                            |
|STATUS [OK] [2025/01/02 03:31:31 PM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=east1, state=ONLINE)                        |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|east3(slave:ONLINE, progress=213, latency=0.942)                            |
|STATUS [OK] [2025/01/02 03:31:29 PM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=east1, state=ONLINE)                        |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
```

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

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

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

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

	Tungsten Cluster Service Seqno		Tungsten Replicator Service Seqno	
Operation	**east**	**west**	**east**	**west**
Insert/update data on `east`	Seqno Increment		Seqno Increment	
Insert/update data on `west`		Seqno Increment		Seqno Increment

Within each cluster, cctrl can be used to monitor the current status. For more information on checking the status and controlling operations, see Section 6.3, "Checking Dataservice Status".

> **Note**
>
> For convenience, the shell PATH can be updated with the tools and configuration. With two separate services, both environments must be updated. To update the shell with the Tungsten Cluster service and tools:
>
> ```
> shell> source /opt/continuent/share/env.sh
> ```
>
> To update the shell with the Tungsten Replicator service and tools:
>
> ```
> shell> source /opt/replicator/share/env.sh
> ```

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

```
shell> multi_trepctl --by-service
| host  | servicename | role   | state  | appliedlastseqno | appliedlatency |
| east1 | east        | master | ONLINE |                5 |          0.322 |
| east2 | east        | slave  | ONLINE |                5 |          0.415 |
| east3 | east        | slave  | ONLINE |                5 |          0.421 |
| west1 | east        | slave  | ONLINE |                5 |          0.422 |
| west2 | east        | slave  | ONLINE |                5 |          0.435 |
| west3 | east        | slave  | ONLINE |                5 |          0.460 |
| west1 | west        | master | ONLINE |              213 |          0.035 |
| east1 | west        | slave  | ONLINE |              213 |          0.104 |
| east2 | west        | slave  | ONLINE |              213 |          0.097 |
| east3 | west        | slave  | ONLINE |              213 |          0.132 |
| west2 | west        | slave  | ONLINE |              213 |          0.096 |
| west3 | west        | slave  | ONLINE |              213 |          0.116 |
```

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

> **Note**
>
> For the multi_trepctl command to work, you will need to have enabled passwordless ssh between all nodes within the cluster.

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

```
shell> trepctl status
```

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

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

3.5.4. Resetting a single dataservice

> **Note**
>
> The procedures in this section are designed for the Multi-Site/Active-Active topology ONLY. Do NOT use these procedures for Composite Active/Active Clustering uing v6 onwards.
>
> For version 6.x onwards, Composite Active/Active Clustering, please refer to Section 3.3, "Deploying Composite Active/Active Clusters"

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

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

The following is a guide to the steps that should be followed. In the example procedure it is the *east* service that has failed. It is assumed that the value of executable-prefix [493] has been set to mm and the env.sh script has been executed to configure the environment.

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

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

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

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

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

```
shell west> mm_trepctl offline
shell west> mm_trepctl -service east reset -all -y
```

4. Place all Tungsten Replicator services on all servers in the failed Tungsten Cluster service to offline:

```
shell east> mm_trepctl offline
```

5. Next we reprovision the primary node in the failed cluster (east1 in our example) with a manual backup taken from a replica node within the west cluster (west3 in this example).

   Shun the east1 datasource to be restored, and put the replicator service offline, if not already in a failed state, using cctrl :

```
cctrl [east]> set force true
cctrl [east]> datasource east1 shun
cctrl [east]> replicator east1 offline
```

6. Shun the west3 datasource to be backed up, and put the replicator service offline using cctrl :

```
cctrl [west]> datasource west3 shun
cctrl [west]> replicator west3 offline
```

7. Stop the mysqld service on both hosts:

```
shell> sudo systemctl stop mysqld
```

8. Delete the mysqld data directory on east1 :

```
east1> sudo rm -rf /var/lib/mysql/*
```

9. If necessary, ensure the tungsten user can write to the MySQL directory on east1:

```
east1> sudo chmod 777 /var/lib/mysql
```

10. Use rsync on west3 to send the data files for MySQL to east1 :

```
west3> rsync -aze ssh /var/lib/mysql/* east1:/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.

11. Recover west3 back to the dataservice (This process will automatically restart MySQL):

```
cctrl [west]> datasource west3 recover
```

12. Update the ownership and permissions on the data files on east1:

```
east1> sudo chown -R mysql:mysql /var/lib/mysql
east1> sudo chmod 770 /var/lib/mysql
```

13. Restart MySQL on east1 :

```
east1> sudo systemctl start mysqld
```

14. Reset the local replication services on east1 :

```
east1> trepctl offline
east1> trepctl -service east reset -all -y
east1> trepctl online
```

15. Recover `east1` witin cctrl :

```
cctrl [east]> set force true
cctrl [east]> datasource east1 welcome
cctrl [east]> datasource east1 online
```

16. Using tprovision, restore the remaining nodes (`east{2,3}`) in the failed `east` service from a host in the newly recovered `east1` host:

```
shell east{2,3}> tprovision -s east1 -m xtrabackup
```

> **Note**
>
> For a full explanation of using tprovison see Section 9.23, "The tprovision Script"

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

```
shell east> mm_trepctl online
```

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

```
shell west> mm_trepctl online
```

19. On the `east`, failed, Tungsten Cluster service, put each Tungsten Connector online:

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

20. Set the policy back to *`AUTOMATIC`*:

```
cctrl> set policy automatic
```

3.5.5. Resetting all dataservices

> **Note**
>
> The procedures in this section are designed for the Multi-Site/Active-Active topology ONLY. Do NOT use these procedures for Composite Active/Active Clustering uing v6 onwards.
>
> For version 6.x onwards, Composite Active/Active Clustering, please refer to Section 3.3, "Deploying Composite Active/Active Clusters"

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

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

Place both clusters into *`MAINTENANCE`* mode:

```
shell east> cctrl
cctrl east> set policy maintenance
```

```
shell west> cctrl
cctrl west> set policy maintenance
```

On all nodes in the east cluster:

```
shell> trepctl offline
shell> mm_trepctl offline

shell> trepctl -service east reset -all -y
shell> mm_trepctl -service west reset -all -y

shell> trepctl online
shell> mm_trepctl online
```

On all nodes in the west cluster:

```
shell> trepctl offline
shell> mm_trepctl offline

shell> trepctl -service west reset -all -y
shell> mm_trepctl -service east reset -all -y

shell> trepctl online
shell> mm_trepctl online
```

Return both clusters to *`AUTOMATIC`* mode:

```
shell east> cctrl
```

```
cctrl east> set policy automatic
```

```
shell west> cctrl
cctrl west> set policy automatic
```

3.5.6. Adding a new Cluster/Dataservice

> **Note**
>
> The procedures in this section are designed for the Multi-Site/Active-Active topology ONLY. Do NOT use these procedures for Composite Active/Active Clustering uing v6 onwards.
>
> For version 6.x onwards, Composite Active/Active Clustering, please refer to Section 3.3, "Deploying Composite Active/Active Clusters"

To add an entirely new cluster (dataservice) to the mesh, follow the below simple procedure.

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

1. Choose a cluster to take a node backup from:
   - Choose a cluster and Replica node to take a backup from.
   - Enable *MAINTENANCE* mode for the cluster:

     ```
     shell> cctrl
     cctrl> set policy maintenance
     ```

   - Shun the selected Replica node and stop both local and cross-site replicator services:

     ```
     shell> cctrl
     cctrl> datasource {replica_hostname_here} shun
     replica shell> trepctl offline
     replica shell> replicator stop
     replica shell> mm_trepctl offline
     replica; shell> mm_replicator stop
     ```

   - Take a backup of the shunned node, then copy to/restore on all nodes in the new cluster.
   - Recover the Replica node and put cluster back into *AUTOMATIC* mode:

     ```
     replica shell> replicator start
     replica shell> trepctl online
     replica shell> mm_replicator start
     replica shell> mm_trepctl online
     shell> cctrl
     cctrl> datasource {replica_hostname_here} online
     cctrl> set policy automatic
     ```

2. On ALL nodes in all three (3) clusters, ensure the `/etc/tungsten/tungsten.ini` has all three clusters defined and all the correct cross-site combinations.
3. Install the Tungsten Clustering software on new cluster nodes to create a single standalone cluster and check the cctrl command to be sure the new cluster is fully online.
4. Install the Tungsten Replicator software on all new cluster nodes and start it.

   Replication will now be flowing INTO the new cluster from the original two.
5. On the original two clusters, run tools/tpm update from the cross-site replicator staging software path:

   ```
   shell> mm_tpm query staging
   shell> cd {replicator_staging_directory}
   shell> tools/tpm update --replace-release
   shell> mm_trepctl online
   shell> mm_trepctl services
   ```

   Check the output from the mm_trepctl services command output above to confirm the new service appears and is online.

   > **Note**
   >
   > There is no need to set the cross-site replicators at a starting position because:
   >
   > - Replicator feeds from the new cluster to the old clusters start at seqno 0.

- The tungsten_olda and tungsten_oldb database schemas will contain the correct starting points for the INBOUND feed into the new cluster, so when the cross-site replicators are started and brought online they will read from the tracking table and carry on correctly from the stored position.

3.5.7. Enabling SSL for Replicators Only

Note

The procedures in this section are designed for the Multi-Site/Active-Active topology ONLY. Do NOT use these procedures for Composite Active/Active Clustering uing v6 onwards.

For version 6.x onwards, Composite Active/Active Clustering, please refer to Section 3.3, “Deploying Composite Active/Active Clusters”

It is possible to enable secure communications for just the Replicator layer in a Multi-Site/Active-Active topology. This would include both the Cluster Replicators and the Cross-Site Replicators because they cannot be SSL-enabled independently.

1. Create a certificate and load it into a java keystore, and then load it into a truststore and place all files into the `/etc/tungsten/` directory. For detailed instructions, see Chapter 5, *Deployment: Security*

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

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

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

3. Put all clusters into *`MAINTENANCE`* mode.

```
shell> cctrl
cctrl> set policy maintenance
```

4. On all hosts, update the cluster configuration:

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

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

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

   #### Important

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

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

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

```
shell> cctrl
cctrl> ls

DATASOURCES:
+---------------------------------------------------------------------------+
|db1(master:ONLINE, progress=208950063, THL latency=0.895)                  |
|STATUS [OK] [2018/04/10 11:47:57 AM UTC][SSL]                              |
+---------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                    |
```

```
|  REPLICATOR(role=master, state=ONLINE)                                  |
|  DATASERVER(state=ONLINE)                                               |
|  CONNECTIONS(created=15307, active=2)                                   |
+-------------------------------------------------------------------------+

+-------------------------------------------------------------------------+
|db2(slave:ONLINE, progress=208950061, latency=0.920)                     |
|STATUS [OK] [2018/04/19 11:18:21 PM UTC][SSL]                            |
+-------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                  |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                       |
|  DATASERVER(state=ONLINE)                                               |
|  CONNECTIONS(created=0, active=0)                                       |
+-------------------------------------------------------------------------+

+-------------------------------------------------------------------------+
|db3(slave:ONLINE, progress=208950063, latency=0.939)                     |
|STATUS [OK] [2018/04/25 12:17:20 PM UTC][SSL]                            |
+-------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                  |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                       |
|  DATASERVER(state=ONLINE)                                               |
|  CONNECTIONS(created=0, active=0)                                       |
+-------------------------------------------------------------------------+
```

Both the local cluster replicator status command trepctl status and the cross-site replicator status command mm_trepctl status will show `thls` instead of `thl` in the values for *masterConnectUri*, *masterListenUri* and *pipelineSource*.

```
shell> trepctl status | grep thl

masterConnectUri       : thls://db1:2112/
masterListenUri        : thls://db5:2112/
pipelineSource         : thls://db1:2112/
```

3.5.8. Dataserver maintenance

Note

The procedures in this section are designed for the Multi-Site/Active-Active topology ONLY. Do NOT use these procedures for Composite Active/Active Clustering uing v6 onwards.

For version 6.x onwards, Composite Active/Active Clustering, please refer to Section 3.3, “Deploying Composite Active/Active Clusters”

To perform maintenance on the dataservice, for example to update the MySQL configuration file, can be achieved in a similar sequence to that shown in Section 6.16, “Performing Database or OS Maintenance”, except that you must also restart the corresponding Tungsten Replicator service after the main Tungsten Cluster service has been placed back online.

For example, to perform maintenance on the `east` service:

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

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

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

   ```
   cctrl [east]> datasource east1 shun
   ```

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

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

   ```
   cctrl [east]> service host/mysql restart
   ```

5. Bring the host back into the dataservice:

   ```
   cctrl [east]> datasource host recover
   ```

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

   ```
   cctrl [east]> switch
   ```

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

8. Set the policy back to *AUTOMATIC*:

   ```
   cctrl> set policy automatic
   ```

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

```
shell> mm_trepctl online
```

3.5.8.1. Fixing Replication Errors

In the event of a replication fault, the standard cctrl, trepctl and other utility commands in Chapter 9, *Command-line Tools* can be used to bring the dataservice back into operation. All the tools are safe to use.

If you have to perform any updates or modifications to the stored MySQL data, ensure binary logging has been disabled using:

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

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

3.6. Deploying Tungsten Connector Only

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

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

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

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

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-clustering-7.0.3-141.tar.gz
```

2. Change to the Tungsten Cluster directory:

```
shell> cd tungsten-clustering-7.0.3-141
```

3. Create the `/etc/tungsten/tungsten.ini` using the example below as a template, however, ideally use a copy of the file from your existing cluster to ensure consistency of hostnames and configuration, then simply add the new connector hostname into the `connectors` [484] entry associated with the service that the connector is to be associated with. Once created to suit your needs, run the tpm to perform the installation.

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

```
[defaults]
user=tungsten
install-directory=/opt/continuent
profile-script=~/.bashrc
application-user=app-user
application-password=secret
application-port=3306
replication-port=13306
rest-api-admin-user=apiuser
rest-api-admin-password=secret

[alpha]
connectors=connectorhost1
master=host1
members=host1,host2,host3
```

Configuration group **`defaults`**

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

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

- `user=tungsten` [521]

  System User

- `install-directory=/opt/continuent` [494]

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

- `profile-script=~/.bashrc` [507]

  Append commands to include env.sh in this profile script

- `application-user=app-user` [474]

  Database username for the connector

- `application-password=secret` [473]

  Database password for the connector

- `application-port=3306` [473]

  Port for the connector to listen on

- `replication-port=13306` [510]

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

- `rest-api-admin-user=apiuser` [512]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `rest-api-admin-password=secret` [512]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **`alpha`**

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

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

- `connectors=connectorhost1` [484]

  Hostnames for the dataservice connectors

- `master=host1` [499]

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

- `members=host1,host2,host3` [499]

  Hostnames for the dataservice members

> **Note**
>
> If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API Access. This must be done by specifying the following options in your configuration:
>
> ```
> rest-api-admin-user=tungsten
> rest-api-admin-pass=secret
> ```
>
> For more information on using and configuring the REST API, see Section 11.1, “Getting Started with Tungsten REST API”

The above creates a configuration specifying the datasources, `host{1,2,3}`, and a single connector host based on the hostname of the installation host. Note that the application and datasource port configuration are the same as required by a typical Tungsten Cluster configuration. The values above are identical to those used in Section 3.1, “Deploying Standalone HA Clusters” deployment.

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

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

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

5. Initialize your `PATH` and environment.

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

6. Start the connector service:

```
shell> connector start
```

Once started:

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

```
[LOGICAL] /dsone > ls

COORDINATOR[host1:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@connector2[16019](ONLINE, created=0, active=0)                    |
|connector@host1[18450](ONLINE, created=19638, active=0)                     |
|connector@host2[1995](ONLINE, created=0, active=0)                          |
|connector@host3[8895](ONLINE, created=0, active=0)                          |
+----------------------------------------------------------------------------+
...
```

- The active status of the connector can be monitored using cctrl as normal.
- Updates to the main cluster will not update the Tungsten Cluster of the standalone connector. The standalone must be updated independently of the remainder of the Tungsten Cluster dataservice.
- Connector can be accessed using the connector host and specified port:

```
shell> mysql -utungsten -p -hconnector -P3306
```

- If using proxy mode, the `user.map` authorization file must be created and managed separately on standalone connectors. For more information, see Section 7.6, “User Authentication”

3.7. Deploying Additional Datasources, Managers, or Connectors

3.7.1. Adding Datasources to an Existing Deployment

1. Ensure the new host that is being added has been configured following the Appendix B, *Prerequisites*.
2. Update the `/etc/tungsten/tungsten.ini` configuration file adding the new host to the list of `members` [499], `hosts` [494], and/or `connectors` [484], if applicable.

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

```
[alpha]
...
members=host1,host2,host3,host4
hosts=host1,host2,host3,host4
connectors=host1,host2,host3,host4
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-7.0.3-141

shell> cd {STAGING_DIRECTORY}

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

For information about making updates when using an INI file, please see Section 10.2.4, “Configuration Changes with an INI file”.

3. Using the `--no-connectors` [504] option updates the current deployment without restarting the existing connectors.
4. Initially, the newly added host will attempt to read the information from the existing THL. If the full THL is not available from the Primary, the new Replica will need to be reprovisioned:
   a. Log into the new host.
   b. Execute tprovision to read the information from an existing Replica and overwrite the data within the new host:

```
shell> tprovision -s host2 -m xtrabackup
```

> **Note**
>
> For a full explanation of using tprovison see Section 9.23, “The tprovision Script”

Once the new host has been added and rprovisioned, check the status in cctrl:

```
[LOGICAL] /alpha > ls

COORDINATOR[host1:AUTOMATIC:ONLINE]

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

DATASOURCES:
+----------------------------------------------------------------------------+
|host1(master:ONLINE, progress=219, THL latency=1.047)                       |
|STATUS [OK] [2024/12/13 04:16:17 PM GMT]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+

+----------------------------------------------------------------------------+
|host2(slave:ONLINE, progress=219, latency=1.588)                            |
|STATUS [OK] [2024/12/13 04:16:17 PM GMT]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=host1, state=ONLINE)                        |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+

+----------------------------------------------------------------------------+
|host3(slave:ONLINE, progress=219, latency=2.021)                            |
|STATUS [OK] [2024/12/13 04:16:18 PM GMT]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=host1, state=ONLINE)                        |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+

+----------------------------------------------------------------------------+
|host4(slave:ONLINE, progress=219, latency=1.000)                            |
|STATUS [OK] [2024/01/17 05:28:54 PM GMT]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=host1, state=ONLINE)                        |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
```

If the host has not come up, or the progress does not match the Primary, check Section 6.6, “Datasource Recovery Steps” for more information on determining the exact status and what steps to take to enable the host operation.

3.7.2. Adding Active Witnesses to an Existing Deployment

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

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

Update the `/etc/tungsten/tungsten.ini`> configuration file adding the new host to the list of `members` [499], adding the host against the `witness` option and enabling the `enable-active-witnesses` [490] option.

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

```
[alpha]
...
members=host1,host2,host3,host4
witnesses=host4
```

```
enable-active-witnesses=true
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-7.0.3-141

shell> cd {STAGING_DIRECTORY}

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

For information about making updates when using an INI file, please see Section 10.2.4, “Configuration Changes with an INI file”.

Using the `--no-connectors` [504] option updates the current deployment without restarting the existing connectors.

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

```
[LOGICAL] /alpha > ls

COORDINATOR[host1:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@host1[20446](ONLINE, created=0, active=0)                         |
|connector@host2[21698](ONLINE, created=0, active=0)                         |
|connector@host3[30354](ONLINE, created=0, active=0)                         |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|host1(slave:ONLINE, progress=8946, latency=0.000)                           |
|STATUS [OK] [2018/12/05 04:27:47 PM GMT]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=host3, state=ONLINE)                        |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+

+----------------------------------------------------------------------------+
|host2(slave:ONLINE, progress=8946, latency=0.334)                           |
|STATUS [OK] [2018/12/05 04:06:59 PM GMT]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=host3, state=ONLINE)                        |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+

+----------------------------------------------------------------------------+
|host3(master:ONLINE, progress=8946, THL latency=0.331)                      |
|STATUS [OK] [2018/11/20 05:39:14 PM GMT]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+


WITNESSES:
+----------------------------------------------------------------------------+
|host4(witness:ONLINE)                                                       |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
+----------------------------------------------------------------------------+
```

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

3.7.3. Replacing an Active Witness as a Full Cluster Node

This section explains the simple process for converting an Active Witness into a full cluster node. This process can be used to either convert the existig node or replace the witness with a new node.

1. First, place the cluster into *MAINTENANCE* mode.

```
shell> cctrl
cctrl> set policy maintenance
```

2. Stop the software on the existing Witness node

```
shell> stopall
```

3. Whether you are converting this host, or adding a new host, ensure any additional prerequisities that are needed for a full cluster node are in place, for example MySQL has been installed.

4. Update the ini on all nodes (including connectors) removing the witness properties and placing the new host as part of the cluster configuration, example below.

   Before:

```
[defaults]
user=tungsten
home-directory=/opt/continuent
application-user=app_user
application-password=secret
application-port=3306
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=secret
replication-port=13306
rest-api-admin-user=apiuser
rest-api-admin-password=secret
mysql-allow-intensive-checks=true

[nyc]
enable-active-witnesses=true
topology=clustered
master=db1
members=db1,db2,db3
witnesses=db3
connectors=db1,db2,db3
```

   After:

```
[defaults]
user=tungsten
home-directory=/opt/continuent
application-user=app_user
application-password=secret
application-port=3306
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=secret
replication-port=13306
rest-api-admin-user=apiuser
rest-api-admin-password=secret
mysql-allow-intensive-checks=true

[nyc]
topology=clustered
master=db1
members=db1,db2,db3
connectors=db1,db2,db3
```

5. Update the software on the existing cluster nodes and connector nodes (If separate). Include `--no-connectors` if you want to manually restart them when convenient.

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> tools/tpm update --replace-release
```

6. Either install on the new host or update on the previous Witness host:

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> tools/tpm install
```

   or:

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> tools/tpm update --replace-release -f
```

7. Once the software has been installed you now need to restore a backup of the database onto the node, or provision the database using the provided scripts. Either restore a backup, create and restore a new backup or use tprovision to restore the database on the host.

8. Start the software on the new node/old witness node

```
shell> startall
```

9. If you issued `--no-connectors` during the update, restart the connectors when convenient

```
shell> connector restart
```

10. Check within cctrl from one of the existing database nodes to check that the status returns the exptected output, if it does, return the cluster to *AUTOMATIC* and the process is complete. If the output is not correct, this is usually due to metadata files not updating, therefore on every node, issue the following:

```
shell> tungsten_reset_manager
```

This will clean the metadata files and stop the manager process. Once the script has completed on all nodes, restart the manager process on each node, one-by-one, starting with the Primary node first, followed by the Replicas

```
shell> manager start
```

Finally, return the cluster to *AUTOMATIC*. If the reset process above was performed, it may take a minute or two for the ls output of cctrl to update whilst the metadata files are refreshed.

3.7.4. Replacing a Full Cluster Node as an Active Witness

This section explains the simple process for converting a full cluster node into an Active Witness.

1. First, place the cluster into *MAINTENANCE* mode.

```
shell> cctrl
cctrl> set policy maintenance
```

2. Stop the software on the existing cluster node

```
shell> stopall
```

3. Stop MySQL on the existing cluster node (Syntax is an example and may differ in your environment)

```
shell> systemctl stop mysqld
```

4. Update the ini on all nodes (including connectors) changing the reference to the node to be a witness node, example below.

Before:

```
[defaults]
user=tungsten
home-directory=/opt/continuent
application-user=app_user
application-password=secret
application-port=3306
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=secret
replication-port=13306
rest-api-admin-user=apiuser
rest-api-admin-password=secret
mysql-allow-intensive-checks=true

[nyc]
topology=clustered
master=db1
members=db1,db2,db3
connectors=db1,db2,db3
```

After:

```
[defaults]
user=tungsten
home-directory=/opt/continuent
application-user=app_user
application-password=secret
application-port=3306
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=secret
replication-port=13306
rest-api-admin-user=apiuser
rest-api-admin-password=secret
mysql-allow-intensive-checks=true

[nyc]
enable-active-witnesses=true
topology=clustered
master=db1
members=db1,db2,db3
witnesses=db3
```

```
connectors=db1,db2,db3
```

5. Update the software on the existing cluster nodes and connector nodes (If separate). Include `--no-connectors` if you want to manually restart them when convenient.

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> tools/tpm update --replace-release
```

6. Update on the host you are converting:

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> tools/tpm update --replace-release -f
```

7. Once the updates have been complete, you should then run the tungsten_reset_manager command on each node in the entire cluster. This will ensure the metadata is clean and reference to the node is reflected to be a witness, rather than a full cluter node. On each node, simply execute the command and follow the on screen prompts:

```
shell> tungsten_reset_manager
```

8. Restart the managers on the nodes you have not converted:

```
shell> manager start
```

9. Start the software on the node that you converted:

```
shell> startall
```

10. If you issued `--no-connectors` during the update, restart the connectors when convenient

```
shell> connector restart
```

11. Check within cctrl from one of the existing database nodes to check that the status returns the exptected output, and then return the cluster to *AUTOMATIC* and the process is complete.

3.7.5. Adding Connectors to an Existing Deployment

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

To add more connectors to an existing deployment:

1. On the new host, ensure the Appendix B, *Prerequisites* have been followed.
2. On all existing nodes, update the `/etc/tungsten/tungsten.ini` configuration file adding the new host to the list of `connectors` [484].

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

```
[alpha]
...
connectors=host4
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-7.0.3-141

shell> cd {STAGING_DIRECTORY}

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

For information about making updates when using an INI file, please see Section 10.2.4, “Configuration Changes with an INI file”.

3. Using the `--no-connectors` [504] option updates the current deployment without restarting the existing connectors.
4. During a period when it is safe to restart the connectors:

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

5. Copy the `/etc/tungsten/tungsten.ini` from an existing node to the new connector being configured.

> **Note**
>
> If SSL is enabled, you should also copy the various SSL keys and certificated from /opt/continuent/share of an existing connector to the new connector

6. Install the Tungsten Cluster package or download the Tungsten Cluster tarball, and unpack it on the new Connector host:

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-clustering-7.0.3-141.tar.gz
```

7. Change to the Tungsten Cluster directory:

```
shell> cd tungsten-clustering-7.0.3-141
```

8. Run tpm to validate the installation. Providing no errors are return, you can then continue to install the software.

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

9. If your connectors are using Proxy, you will need to copy the `user.map` file from an existing connector to ensure aiuthentication of clients succeeds

10. If you supplied `start-and-report=true` [514] the software will be running, if this wasn't included, you can now start the software as follows:

```
shell > connector start
```

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

```
[LOGICAL] /alpha > ls

COORDINATOR[host1:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@host1[8616](ONLINE, created=0, active=0)                          |
|connector@host2[12381](ONLINE, created=0, active=0)                         |
|connector@host3[19708](ONLINE, created=0, active=0)                         |
|connector@host4[5085](ONLINE, created=0, active=0)                          |
+----------------------------------------------------------------------------+
```

3.7.6. Converting from a single cluster to a composite cluster

There are two possible scenarios for converting from a single standalone cluster to a composite cluster. The two following sections will guide you through examples of each of these.

3.7.6.1. Convert and add new nodes as a new service

The following steps guide you through updating the configuration to include the new hosts as a new service and convert to a Composite Cluster.

For the purpose of this worked example, we have a single cluster dataservice called `east` with three nodes, defined as `db1`, `db2` and `db3` with `db1` as the Primary.

Our goal is to create a new cluster dataservice called `west` with three nodes, defined as `db4`,`db5` and `db6` with `db4` as the `relay`.

We will configure a new composite dataservice called `global`

The steps show two alternative approaches, to create the `west` as a Passive cluster (Composite Active/Passive) or to create the `west` cluster as a second active cluster (Composite Active/Active)

1. On the new host(s), ensure the Appendix B, *Prerequisites* have been followed.

2. On the new host(s), ensure the `tungsten.ini` contains the correct service blocks for both the existing cluster and the new cluster.

3. On the new host(s), install the proper version of clustering software, ensuring that the version being installed matches the version currently installed on the existing hosts.

```
shell> cd /opt/continuent/sofware
shell> tar zxvf tungsten-clustering-7.0.3-141.tar.gz
shell> cd tungsten-clustering-7.0.3-141
shell> ./tools/tpm install
```

> **Important**
>
> Ensure `--start-and-report` [514] is set to `false` in the configuration for the new hosts.

4. Set the existing cluster to *`MAINTENANCE`* mode using cctrl:

```
shell> cctrl
[LOGICAL] / > set policy maintenance
```

5. Add the definition for the new cluster service `west` and composite service `global` to the existing configuration on the existing host(s):

   For Composite Active/Passive

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

```
[west]
...
connectors=db4,db5,db6
relay-source=east
relay=db4
slaves=db5,db6
topology=clustered

[global]
...
composite-datasources=east,west
```

   Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-7.0.3-141

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update --no-connectors --replace-release
```

   For information about making updates when using an INI file, please see Section 10.2.4, “Configuration Changes with an INI file”.

   For Composite Active/Active

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

```
[west]
topology=clustered
connectors=db4,db5,db6
master=db4
members=db4,db5,db6

[global]
topology=composite-multi-master
composite-datasources=east,west
```

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-7.0.3-141

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update --no-connectors --replace-release
```

   Note

   Using the optional `--no-connectors` [504] option updates the current deployment without restarting the existing connectors.

   Note

   Using the `--replace-release` [443] option ensures the metadata files for the cluster are correctly rebuilt. This parameter MUST be supplied.

6. On every node in the original EAST cluster, make sure all replicators are online:

```
shell> trepctl services
shell> trepctl -all-services online
```

7. If SSL is enabled, you will need to copy the relevant security keys and files to all hosts in the new cluster. tpm cert can be used to simplify this process. For more information see tpm cert Command

8. On all the new hosts in the new cluster, start the manager processes ONLY

```
shell> manager start
```

9. From the original cluster, use cctrl to check that the new dataservice and composite dataservice have been created, and place the new dataservice into *MAINTENANCE* mode

```
shell> cctrl
cctrl> cd /
cctrl> ls
cctrl> use global
cctrl> ls
cctrl> datasource east online
cctrl> set policy maintenance
```

Example from a Composite Active/Passive Cluster

```
tungsten@db1:~  $ cctrl
Tungsten Clustering 7.0.3 build 141
east: session established, encryption=false, authentication=false

[LOGICAL] /east > cd /
[LOGICAL] / > ls

[LOGICAL] / > ls
global
  east
  west

[LOGICAL] / > use global
[LOGICAL] /global > ls
COORDINATOR[db3:MIXED:ONLINE]
   east:COORDINATOR[db3:MAINTENANCE:ONLINE]
   west:COORDINATOR[db5:AUTOMATIC:ONLINE]

ROUTERS:
+---------------------------------------------------------------------------+
|connector@db1[9493](ONLINE, created=0, active=0)                           |
|connector@db2[9341](ONLINE, created=0, active=0)                           |
|connector@db3[10675](ONLINE, created=0, active=0)                          |
+---------------------------------------------------------------------------+

DATASOURCES:
+---------------------------------------------------------------------------+
|east(composite master:OFFLINE)                                              |
|STATUS [OK] [2024/12/09 11:04:17 AM UTC]                                   |
+---------------------------------------------------------------------------+
+---------------------------------------------------------------------------+
|west(composite slave:OFFLINE)                                              |
|STATUS [OK] [2024/12/09 11:04:17 AM UTC]                                   |
+---------------------------------------------------------------------------+

REASON FOR MAINTENANCE MODE: MANUAL OPERATION

[LOGICAL] /global > datasource east online
composite data source 'east@global' is now ONLINE

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

Example from a Composite Active/Active Cluster

```
tungsten@db1:~  $ cctrl
Tungsten Clustering 7.0.3 build 141
east: session established, encryption=false, authentication=false

[LOGICAL] /east > cd /
[LOGICAL] / > ls

[LOGICAL] / > ls
global
  east
    east_from_west
  west
    west_from_east

[LOGICAL] / > use global
[LOGICAL] /global > ls
COORDINATOR[db3:MIXED:ONLINE]
   east:COORDINATOR[db3:MAINTENANCE:ONLINE]
   west:COORDINATOR[db4:AUTOMATIC:ONLINE]

ROUTERS:
+---------------------------------------------------------------------------+
```

```
|connector@db1[23431](ONLINE, created=0, active=0)                              |
|connector@db2[25535](ONLINE, created=0, active=0)                              |
|connector@db3[15353](ONLINE, created=0, active=0)                              |
+-------------------------------------------------------------------------------+

DATASOURCES:
+-------------------------------------------------------------------------------+
|east(composite master:OFFLINE, global progress=10, max latency=1.043)          |
|STATUS [OK] [2024/08/13 11:05:01 AM UTC]                                       |
+-------------------------------------------------------------------------------+
|  east(master:ONLINE, progress=10, max latency=1.043)                          |
|  east_from_west(UNKNOWN:UNKNOWN, progress=-1, max latency=-1.000)             |
+-------------------------------------------------------------------------------+
+-------------------------------------------------------------------------------+
|west(composite master:ONLINE, global progress=-1, max latency=-1.000)          |
|STATUS [OK] [2024/08/13 11:07:56 AM UTC]                                       |
+-------------------------------------------------------------------------------+
|  west(UNKNOWN:UNKNOWN, progress=-1, max latency=-1.000)                       |
|  west_from_east(UNKNOWN:UNKNOWN, progress=-1, max latency=-1.000)             |
+-------------------------------------------------------------------------------+

REASON FOR MAINTENANCE MODE: MANUAL OPERATION

[LOGICAL] /global > datasource east online
composite data source 'east@global' is now ONLINE

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

10. Start the replicators in the new cluster ensuring they start as OFFLINE:

```
shell> replicator start offline
```

11. Go to the relay (or Primary) node of the new cluster (i.e. db4) and provision it from a Replica of the original cluster (i.e. db2):

    Provision the new relay in a Composite Active/Passive Cluster

```
db4-shell> tprovision -s db2 -m xtrabackup
```

    Provision the new primary in a Composite Active/Active Cluster

```
db4-shell> tprovision -s db2 -m xtrabackup -c
```

    > **Note**
    >
    > For a full explanation of using tprovison see Section 9.23, "The tprovision Script"

12. Go to each Replica node of the new cluster and provision from the relay node of the new cluster (i.e. db4):

```
db5-shell> tprovision -s db4 -m xtrabackup
```

13. Bring the replicators in the new cluster online, if not already:

```
shell> trepctl -all-services online
```

14. From a node in the original cluster (e.g. `db1`), using cctrl, set the composite cluster online, if not already, and return to *`AUTOMATIC`*:

```
shell> cctrl
[LOGICAL] / > use global
[LOGICAL] / > datasource west online
[LOGICAL] / > set policy automatic
```

15. Start the connectors associated with the new cluster hosts in `west`:

```
shell> connector start
```

    > **Note**
    >
    > Depending on the mode in which the connectors are running, you may need to configure the `user.map`. If this is in use on the old cluster, then we recommend that you take a copy of this file and place this on the new connectors associated with the new cluster, and then adjust any affinity settings that are required. Additionally, the `user.map` may need adjustments on the original cluster. For more details on the `user.map` file, it is advised to review the relevant sections in the Connector documentation related to the mode your connectors are operating in. These can be found at Section 7.6.1, "`user.map` File Format"

16. If `--no-connectors` [504] was issued during the update, then during a period when it is safe, restart the connectors associated with the original cluster:

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

3.7.6.2. Convert and move nodes to a new service

This method of conversion is a little more complicated and the only safe way to accomplish this would require downtime for the replication on all nodes.

To achieve this without downtime to your applications, it is recommended that all application activity be isolated to the Primary host only. Following the conversion, all activity will then be replicated to the Replica nodes

Our example starting cluster has 5 nodes (1 Primary and 4 Replicas) and uses service name `alpha`. Our target cluster will have 6 nodes (3 per cluster) in 2 member clusters `alpha_east` and `alpha_west` in composite service `alpha`.

This means that we will reuse the existing service name `alpha` as the name of the new composite service, and create two new service names, one for each cluster (`alpha_east` and `alpha_west`).

To convert the above configuration, follow the steps below:

1. On the new host, ensure the Appendix B, *Prerequisites* have been followed.
2. Ensure the cluster is in *`MAINTENANCE`* mode. This will prevent the managers from performing any unexpected recovery or failovers during the process.

   ```
   cctrl> set policy maintenance
   ```
3. Next, you must stop all services on all existing nodes.

   ```
   shell> stopall
   ```
4. Update tungsten.ini on all original 5 nodes, then copy the file to the new node.

   You will need to create two new services for each cluster, and change the original service stanza to represent the composite service. An example of how the complete configuration would look is below.

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

   ```
   [defaults]
   user=tungsten
   install-directory=/opt/continuent
   profile-script=~/.bash_profile
   replication-user=tungsten
   replication-password=secret
   replication-port=13306
   application-user=app_user
   application-password=secret
   application-port=3306
   rest-api-admin-user=apiuser
   rest-api-admin-password=secret

   [alpha_east]
   topology=clustered
   master=db1
   members=db1,db2,db3
   connectors=db1,db2,db3

   [alpha_west]
   topology=clustered
   relay=db4
   members=db4,db5,db6
   connectors=db4,db5,db6
   relay-source=alpha_east

   [alpha]
   composite-datasources=alpha_east,alpha_west
   ```
5. Using you preferred backup/restore method, take a backup of the MySQL database on one of the original nodes and restore this to the new node

   If preferred, this step can be skipped, and the provision of the new node completed via the use of the supplied provisioning scripts, explained in Step 10 below.
6. Invoke the conversion using the tpm command from the software extraction directory.

   Update the software on the existing hosts. This command should be run on all 5 original nodes.

   ```
   shell> tpm query staging
   ```

```
shell> cd {software_staging_dir_from_tpm_query}
shell> ./tools/tpm update --replace-release --force
shell> rm /opt/continuent/tungsten/cluster-home/conf/cluster/*/datasource/*
```

> **Note**
>
> The use of the `--force` [422] option is required to force the override of the old properties

7. Proceed to install the software using the tpm command from the software extraction directory on the new node:

```
shell> cd {software_staging_dir}
shell> ./tools/tpm install
```

> **Note**
>
> Ensure you install the same version of software on the new node that matches exactly, the version on the existing 5 nodes

8. Start all services on all existing nodes.

```
shell> startall
```

9. Bring the clusters back into *AUTOMATIC* mode:

```
shell> cctrl -multi
cctrl> use alpha
cctrl> set policy automatic
cctrl> exit
```

10. If you skipped the backup/restore step above, you now need to provision the database on the new node. To do this, use the tprovision script to provision the database from one of the existing nodes, for example `db5`

```
shell> tprovision -s db5 -m xtrabackup
```

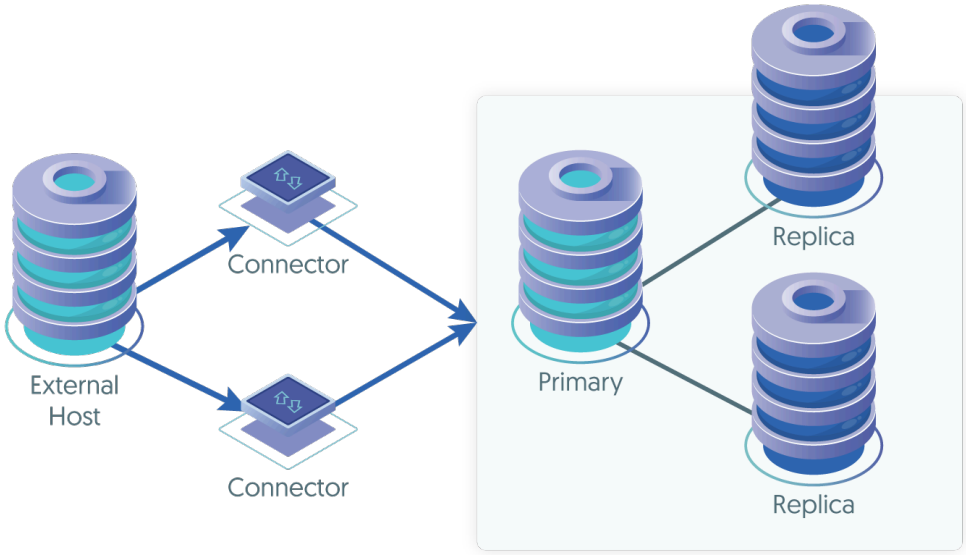
3.8. Replicating Data Into an Existing Dataservice

If you have an existing dataservice, data can be replicated from a standalone MySQL server into the service. The replication is configured by creating a service that reads from the standalone MySQL server and writes into the cluster through a connector attached to your dataservice. By writing through the connector, changes to the underlying dataservice topology can be handled.

Using a replicator that writes data into an existing data service can be used when migrating from an existing service into a new Tungsten Cluster service.

For more information on initially provisioning the data for this type of operation, see Section 6.12.2, “Migrating from MySQL Native Replication Using a New Service”.

Figure 3.6. Topologies: Replicating into a Dataservice



In order to configure this deployment, there is just on simpe step:

1. Create a new replicator on the source server that extracts the data from the binary logs, and applies to the target cluster using a special `direct` topology.

There are also the following requirements:

- If Tungsten Cluster™ is in use on the target, then you must use the same version of Tungsten Replicator™ that matches the version runnning in the Cluster.
- 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` privileges.
- Replicator must be able to connect as the `tungsten` user to the databases within the cluster.
- When writing into the Primary through the connector, the user must be given the correct privileges to write and update the MySQL server. For this reason, the easiest method is to use the `tungsten` user, and ensure that that user has been added to the `user.map`:

```
tungsten secret alpha
```

Install the Tungsten Replicator™ package (see Section 2.2.2, “Using the RPM package files”), or download the compressed tarball and unpack it on `host1`:

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-replicator-7.0.3-141.tar.gz
```

Change to the Tungsten Replicator staging directory:

```
shell> cd tungsten-replicator-7.0.3-141
```

Configure the replicator on `host1`

Create the `/etc/tungsten/tungsten.ini` using the example below as a template. Once created to suit your needs, run tpm to perform the installation.

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

```
[defaults]
install-directory=/opt/continuent
user=tungsten
skip-validation-check=MySQLNoMySQLReplicationCheck
rest-api-admin-user=apiuser
rest-api-admin-password=secret
replicator-rest-api-address=0.0.0.0
log-slave-updates=true

[beta]
topology=direct
master=host1
direct-datasource-host=host1
direct-replication-port=3306
replication-host=connectorhost1
replication-user=tungsten
replication-password=secret
replication-port=3306
```

Configuration group **`defaults`**

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

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

- `install-directory=/opt/continuent` [494]

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

- `user=tungsten` [521]

  System User

- `skip-validation-check=MySQLNoMySQLReplicationCheck` [444]

The skip-validation-check [444] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

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

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

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

The check in this case is MySQLDefaultTableTypeCheck [455], and could be ignored using skip-validation-check=MySQLDefaultTableTypeCheck [444].

Values can be passed as a comma-separated list, or single skip-validation-check [444] entries for each check to be skipped.

Setting both skip-validation-check [444] and enable-validation-check [442] is equivalent to explicitly disabling the specified check.

- rest-api-admin-user=apiuser [512]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.
- rest-api-admin-password=secret [512]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.
- replicator-rest-api-address=0.0.0.0 [511]

  Address for the API to bind too.
- log-slave-updates=true [497]

  Should replicas log updates to binlog

Configuration group **beta**

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

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

- topology=direct [520]

  Replication topology for the dataservice.
- master=host1 [499]

  The hostname of the primary (extractor) within the current service.
- direct-datasource-host=host1 [489]

  Database server hostname
- direct-replication-port=3306 [489]

  Database server port
- replication-host=connectorhost1 [510]

  Hostname of the datasource where the database is located. If the specified hostname matches the current host or member name, the database is assumed to be local. If the hostnames do not match, extraction is assumed to be via remote access. For MySQL hosts, this configures a remote replication Replica (relay) connection.
- replication-user=tungsten [510]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).
- replication-password=secret [510]

  The password to be used when connecting to the database using the corresponding --replication-user [510].

- `replication-port=3306` [510]

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

This creates a configuration that specifies that the topology should read directly from the source host, `host1`, writing directly to `connectorhost1`.

This assumes the replicator is being installed on the source host, `host1`.

Now install the service using tpm:

```
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 can start the replicator as follows:

```
shell> replicator start
```

By default the replicator will start from the current binlog position. If you need to start from a different position, you can do this as follows:

```
shell> replicator start offline
```

Next, using the binary log and position you want to start from, you can position the replicators starting position. For example, you have determined that the starting point is binary log file `mysql-bin.000011` and position `758`, then using the index of the filename and position, execute the following command:

```
shell> trepctl online -from-event 000011:758
```

You can now monitor replication using the status command:

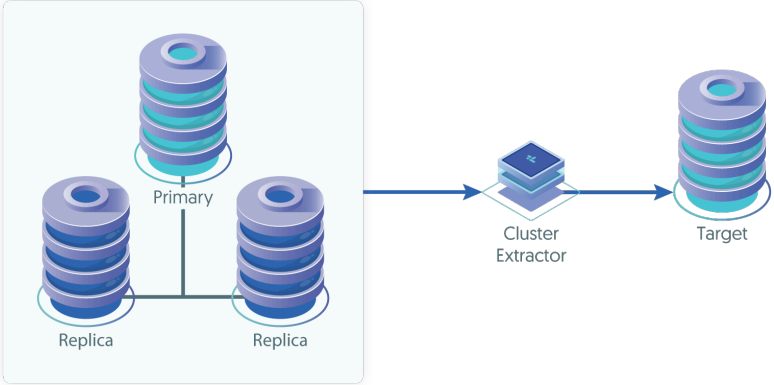
```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000011:0000000000003988;156
appliedLastGoodLatency : 171.866
appliedLastGoodSeqno   : 8
appliedLastSeqno       : 8
appliedLatency         : 171.866
autoRecoveryEnabled    : false
autoRecoveryTotal      : 0
channels               : 1
clusterName            : beta
currentEventId         : mysql-bin.000012:0000000000000157
currentTimeMillis      : 1737546857700
dataServerHost         : connectorhost1
extensions             :
host                   : host1
latestEpochNumber      : 0
masterConnectUri       : null
masterListenUri        : null
maximumStoredSeqNo     : 8
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : jdbc:mysql:thin://host1:13306/
relativeLatency        : 378.7
resourceJdbcDriver     : org.drizzle.jdbc.DrizzleDriver
resourceJdbcUrl        : jdbc:mysql:thin://connectorhost1:3306/${DBNAME}?jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull&tinyInt1isBit=fa
resourceVendor         : mysql
rmiPort                : 10002
role                   : direct
seqnoType              : java.lang.Long
serviceName            : beta
serviceType            : local
simpleServiceName      : beta
siteName               : default
sourceId               : host1
state                  : ONLINE
timeInStateSeconds     : 208.04
timezone               : GMT
transitioningTo        :
uptimeSeconds          : 333.742
useSSLConnection       : false
version                : Tungsten Replicator 7.0.3 build 141
```

```
Finished status command...
```

3.9. Replicating Data Out of a Cluster

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

Figure 3.7. Topologies: Replicating Data Out of a Cluster



During the installation process a `topology=cluster-alias` [520] and `cluster-slave` are declared. The `cluster-alias` describes all of the servers in the cluster and how they may be reached. The `cluster-slave` defines one or more servers that will replicate from the cluster.

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

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

3.9.1. Prepare: Replicating Data Out of a Cluster

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

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

3.9.2. Deploy: Replicating Data Out of a Cluster

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

   ```
   shell> cd /opt/continuent/software
   shell> tar zxf tungsten-replicator-7.0.3-141.tar.gz
   ```
2. Change to the unpackaged directory:

```
shell> cd tungsten-replicator-7.0.3-141
```

3. Configure the replicator

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

```
[defaults]
install-directory=/opt/continuent
profile-script=~/.bash_profile
replication-password=secret
replication-port=13306
replication-user=tungsten
user=tungsten
rest-api-admin-user=apiuser
rest-api-admin-password=secret
replicator-rest-api-address=0.0.0.0

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

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

Configuration group **defaults**

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

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

- `install-directory=/opt/continuent` [494]

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

- `profile-script=~/.bash_profile` [507]

  Append commands to include env.sh in this profile script

- `replication-password=secret` [510]

  The password to be used when connecting to the database using the corresponding `--replication-user` [510].

- `replication-port=13306` [510]

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

- `replication-user=tungsten` [510]

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

- `user=tungsten` [521]

  System User

- `rest-api-admin-user=apiuser` [512]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `rest-api-admin-password=secret` [512]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

- `replicator-rest-api-address=0.0.0.0` [511]

  Address for the API to bind too.

Configuration group **alpha**

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

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

- `master=host1` [499]

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

- `members=host1,host2,host3` [499]

  Hostnames for the dataservice members

- `thl-port=2112` [520]

  Port to use for THL Operations

- `topology=cluster-alias` [520]

  Replication topology for the dataservice.

Configuration group **`beta`**

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

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

- `relay=host6` [499]

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

- `relay-source=alpha` [508]

  Dataservice name to use as a relay source

- `topology=cluster-slave` [520]

  Replication topology for the dataservice.

> **Important**
>
> If you are replicating to a non-MySQL server. Include the following steps in your configuration.
>
> ```
> shell> mkdir -p /opt/continuent/share/
> shell> cp tungsten-replicator/support/filters-config/convertstringfrommysql.json »
>    /opt/continuent/share/
> ```
>
> Then, include the following parameters in the configuration
>
> ```
> property=replicator.stage.remote-to-thl.filters=convertstringfrommysql
> property=replicator.filter.convertstringfrommysql.definitionsFile= »
>    /opt/continuent/share/convertstringfrommysql.json
> ```

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

> **Note**
>
> Do not include `start-and-report=true` [514] if you are taking over for MySQL native replication. See Section 6.12.1, “Migrating from MySQL Native Replication 'In-Place'” for next steps after completing installation.

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

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

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

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

The replicator should be installed and ready to use.

3.10. Migrating and Seeding Data

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

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

1. Deploy Tungsten Cluster using the model or system appropriate according to Chapter 2, *Deployment*. Ensure that the Tungsten Cluster is not started automatically by excluding the `--start` [514] or `--start-and-report` [514] options from the tpm commands.

2. On each Replica

   Confirm that native replication is working on all Replica nodes :

   MySQL 8.4+

   ```
   shell> echo 'SHOW REPLICA STATUS\G' | tpm mysql | \
   egrep 'Source_Host| Last_Error| Replica_SQL_Running'
                   Source_Host: tr-ssl1
              Replica_SQL_Running: Yes
                     Last_Error:
   ```

   Upto MySQL 8.0

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

3. On the Primary and each Replica

   Reset the Tungsten Replicator position on all servers :

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

4. On the Primary

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

   ```
   shell> startall
   shell> trepctl online
   ```

5. On the Primary

   Put the cluster into *MAINTENANCE* mode using cctrl to prevent Tungsten Cluster automatically reconfiguring services:

   ```
   cctrl > set policy maintenance
   ```

6. On each Replica

   MySQL 8.4+

   Record the current Replica log position (as reported by the `Relay_Source_Log_File` and `Exec_Source_Log_Pos` output from `SHOW REPLICA STATUS`. Ideally, each Replica should be stopped at the same position:

   ```
   shell> echo 'SHOW REPLICA STATUS\G' | tpm mysql | \
   egrep 'Source_Host| Last_Error| Relay_Source_Log_File| Exec_Source_Log_Pos'
                   Source_Host: tr-ssl1
          Relay_Source_Log_File: mysql-bin.000025
                     Last_Error: Error executing row event: 'Table 'tungsten_alpha.heartbeat' doesn't exist'
            Exec_Source_Log_Pos: 181268
   ```

   Upto MySQL 8.0

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

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

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

7. On the Primary

   Take the replicator offline and clear the THL:

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

8. On the Primary

   Start replication, using the *lowest* binary log file and log position from the Replica information determined in step 6.

   ```
   shell> trepctl online -from-event 000025:181268
   ```

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

9. On each Replica

   a. Disable native replication to prevent native replication being accidentally started on the Replica.

      MySQL 8.4+

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

      Upto MySQL 8.0

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

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

      ```
      shell> trepctl online
      shell> startall
      ```

      The Replica will start reading from the binary log position configured on the Primary.

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

      ```
      shell> trepctl online -from-event 000025:188249
      shell> startall
      ```

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

11. Put the cluster back into *AUTOMATIC* mode:

    ```
    cctrl> set policy automatic
    ```

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

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

Once these steps have been completed, Tungsten Cluster should be operating as the replication service for your MySQL servers. Use the information in Chapter 6, *Operations Guide* to monitor and administer the service.

3.10.2. Migrating from MySQL Native Replication Using a New Service

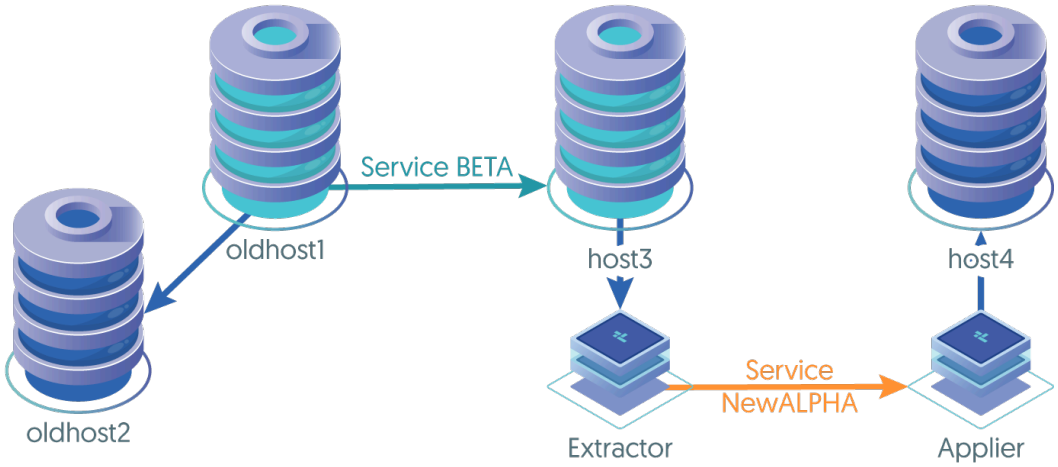
When running an existing MySQL native replication service that needs to be migrated to a Tungsten Cluster service, one solution is to create the new Tungsten Cluster service, synchronize the content, and then install a service that migrates data from the existing native service to the new service while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

The basic structure is shown in Figure 3.8, “Migration: Migrating Native Replication using a New Service”. The migration consists of two steps:

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

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

Figure 3.8. Migration: Migrating Native Replication using a New Service



To configure the service:

1. Stop replication on a Replica for the existing native replication installation :

   MySQL 8.4+

   ```
   mysql> STOP REPLICA;
   ```

   Obtain the current Replica position within the Primary binary log :

   ```
   mysql> SHOW REPLICA STATUS\G
   ...
                   Source_Host: host3
         Relay_Source_Log_File: mysql-bin.000002
           Exec_Source_Log_Pos: 559
   ...
   ```

   Upto MySQL 8.0

   ```
   mysql> STOP SLAVE;
   ```

   Obtain the current Replica position within the Primary binary log :

   ```
   mysql> SHOW SLAVE STATUS\G
   ...
                   Master_Host: host3
         Relay_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 Primary may be used if you do not have a Replica 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 Replica using native replication :

   MySQL 8.4+

   ```
   mysql> START REPLICA;
   ```

   Upto MySQL 8.0

   ```
   mysql> START SLAVE;
   ```

4. On the Primary and each Replica within the new service, restore the backup data and start the database service

5. Setup the new Tungsten Cluster 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 Primary, and the Primary of `newalpha`. The information provided in Section 3.8, “Replicating Data Into an Existing Dataservice” will help.

7. Start the replicator in an offline state:

```
shell> replicator start offline
```

8. Set the replication position for `beta` to the point within the binary logs where the backup was taken:

```
shell> trepctl online -from-event 000002:559
```

Once replication has been started, use trepctl to check the status and ensure that replication is operating correctly.

The original native MySQL replication Primary can continue to be used for reading and writing from within your application, and changes will be replicated into the new service on the new hardware. Once the applications have been updated to use the new service, the old servers can be decommissioned and replicator service `beta` stopped and removed.

3.10.3. Seeding Data through MySQL

Once the Tungsten Replicator is installed, it can be used to provision all Replicas with the Primary data. The Replicas 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 Primary and reloading it back into the Primary server. This will create a full set of THL entries for the Replica replicators to apply. There may be no other applications accessing the Primary 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 Primary is a MySQL Replica, then the Replica process may be stopped and started to prevent any changes without affecting other servers.

1. If you are using a MySQL Replica as the Primary, stop the replication thread :

   MySQL 8.4+

   ```
   mysql> STOP REPLICA;
   ```

   Upto MySQL 8.0

   ```
   mysql> STOP SLAVE;
   ```

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

   ```
   shell> trepctl status
   ```

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

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 Replica 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 Replica or use ddlscan to recreate them.

5. Load the dump file back into the Primary 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 Replicas will automatically apply these statements through normal replication.

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

   MySQL 8.4+

   ```
   mysql> START REPLICA;
   ```

   Upto MySQL 8.0

   ```
   mysql> START SLAVE;
   ```

7. Monitor replication status on the Primary and Replicas :

   ```
   shell> trepctl status
   ```

Chapter 4. Deployment: Advanced

The following sections provide guidance and instructions for creating advanced deployments, including configuration automatic startup and shutdown during boot procedures, upgrades, downgrades, and removal of Tungsten Cluster.

4.1. Deploying Parallel Replication

Parallel apply is an important technique for achieving high speed replication and curing Replica lag. It works by spreading updates to Replicas over multiple threads that split transactions on each schema into separate processing streams. This in turn spreads I/O activity across many threads, which results in faster overall updates on the Replica. In ideal cases throughput on Replicas may improve by up to 5 times over single-threaded MySQL native replication.

> **Note**
>
> It is worth noting that the only thing Tungsten parallelizes is applying transactions to Replicas. All other operations in each replication service are single-threaded.

4.1.1. Application Prerequisites for Parallel Replication

Parallel replication works best on workloads that meet the following criteria:

- ROW based binary logging must be enabled in the MySQL database.
- Data are stored in independent schemas. If you have 100 customers per server with a separate schema for each customer, your application is a good candidate.
- Transactions do not span schemas. Tungsten serializes such transactions, which is to say it stops parallel apply and runs them by themselves. If more than 2-3% of transactions are serialized in this way, most of the benefits of parallelization are lost.
- Workload is well-balanced across schemas.
- The Replica host(s) are capable and have free memory in the OS page cache.
- The host on which the Replica runs has a sufficient number of cores to operate a large number of Java threads.
- Not all workloads meet these requirements. If your transactions are within a single schema only, you may need to consider different approaches, such as Replica prefetch. Contact Continuent for other suggestions.

Parallel replication does not work well on underpowered hosts, such as Amazon m1.small instances. In fact, any host that is already I/O bound under single-threaded replication will typical will not show much improvement with parallel apply.

> **Note**
>
> Currently, it is not recommended to use the SMARTSCALE connector configuration in conjunction with Parallel Apply. This is due to progress only being measured against the slowest channel.

4.1.2. Enabling Parallel Apply During Install

Parallel apply is enabled using the `svc-parallelization-type` [516] and `channels` [477] options of tpm. The parallelization type defaults to *`none`* which is to say that parallel apply is disabled. You should set it to *`disk`* [104]. The `channels` [477] option sets the the number of channels (i.e., threads) you propose to use for applying data. Here is a code example of a MySQL Applier installation with parallel apply enabled. The Replica will apply transactions using 30 channels.

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

```
[defaults]
install-directory=/opt/continuent
user=tungsten
mysql-allow-intensive-checks=true
profile-script=~/.bash_profile
application-port=3306
application-user=app_user
application-password=secret
replication-port=13306
replication-user=tungsten
replication-password=secret
connector-smartscale=false # parallel apply and smartscale are not compatible
svc-parallelization-type=disk
channels=10
rest-api-admin-user=apiuser
```

```
rest-api-admin-password=secret
connector-rest-api-address=0.0.0.0
manager-rest-api-address=0.0.0.0
replicator-rest-api-address=0.0.0.0

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

Configuration group **defaults**

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

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

- `install-directory=/opt/continuent` [494]

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

- `user=tungsten` [521]

  System User

- `mysql-allow-intensive-checks=true` [502]

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

- `profile-script=~/.bash_profile` [507]

  Append commands to include env.sh in this profile script

- `application-port=3306` [473]

  Port for the connector to listen on

- `application-user=app_user` [474]

  Database username for the connector

- `application-password=secret` [473]

  Database password for the connector

- `replication-port=13306` [510]

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

- `replication-user=tungsten` [510]

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

- `replication-password=secret` [510]

  The password to be used when connecting to the database using the corresponding `--replication-user` [510].

- `connector-smartscale=false # parallel apply and smartscale are not compatible` [483]

  Enable SmartScale R/W splitting in the connector

- `svc-parallelization-type=disk` [516]

  Method for implementing parallel apply

- `channels=10` [477]

  Number of replication channels to use for parallel apply.

- `rest-api-admin-user=apiuser` [512]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `rest-api-admin-password=secret` [512]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.
- `connector-rest-api-address=0.0.0.0` [482]

  Address for the API to bind too.
- `manager-rest-api-address=0.0.0.0` [498]

  Address for the API to bind too.
- `replicator-rest-api-address=0.0.0.0` [511]

  Address for the API to bind too.

Configuration group **`alpha`**

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

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

- `master=host1` [499]

  The hostname of the primary (extractor) within the current service.
- `members=host1,host2,host3` [499]

  Hostnames for the dataservice members
- `connectors=host1,host2,host3` [484]

  Hostnames for the dataservice connectors
- `topology=clustered` [520]

  Replication topology for the dataservice.

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` [477] — Sets the replicator block commit size, which is the number of transactions to commit at once on Replicas. Values up to 100 are normally fine.
- `native-slave-takeover` [504] — Used to allow Tungsten to take over from native MySQL replication and parallelize it.

You can check the number of active channels on a Replica by looking at the "channels" property once the replicator restarts.

```
Replica shell> trepctl -service alpha status| grep channels
channels               : 10
```

Important

The channel count for a Primary will ALWAYS be 1 because extraction is single-threaded:

```
Primary shell> trepctl -service alpha status| grep channels
channels               : 1
```

Warning

Enabling parallel apply will dramatically increase the number of connections to the database server.

Typically the calculation on a Replica would be: Connections = Channel_Count x Sevice_Count x 2, so for a 4-way Composite Composite Active/Active topology with 30 channels there would be 30 x 4 x 2 = 240 connections required for the replicator alone, not counting application traffic.

You may display the currently used number of connections in MySQL:

```
mysql> SHOW STATUS LIKE 'max_used_connections';
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| Max_used_connections | 190   |
+----------------------+-------+
```

```
1 row in set (0.00 sec)
```

Below are suggestions for how to change the maximum connections setting in MySQL both for the running instance as well as at startup:

```
mysql> SET GLOBAL max_connections = 512;

mysql> SHOW VARIABLES LIKE 'max_connections';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_connections | 512   |
+-----------------+-------+
1 row in set (0.00 sec)

shell> vi /etc/my.cnf
#max_connections = 151
max_connections = 512
```

4.1.3. Channels

Channels and Parallel Apply

Parallel apply works by using multiple threads for the final stage of the replication pipeline. These threads are known as channels. Restart points for each channel are stored as individual rows in table `trep_commit_seqno` [693] if you are applying to a relational DBMS server, including MySQL, Oracle, and data warehouse products like Vertica.

When you set the `channels` [477] argument, the tpm program configures the replication service to enable the requested number of channels. A value of 1 results in single-threaded operation.

Do not change the number of channels without setting the replicator offline cleanly. See the procedure later in this page for more information.

How Many Channels Are Enough?

Pick the smallest number of channels that loads the Replica fully. For evenly distributed workloads this means that you should increase channels so that more threads are simultaneously applying updates and soaking up I/O capacity. As long as each shard receives roughly the same number of updates, this is a good approach.

For unevenly distributed workloads, you may want to decrease channels to spread the workload more evenly across them. This ensures that each channel has productive work and minimizes the overhead of updating the channel position in the DBMS.

Once you have maximized I/O on the DBMS server leave the number of channels alone. Note that adding more channels than you have shards does not help performance as it will lead to idle channels that must update their positions in the DBMS even though they are not doing useful work. This actually slows down performance a little bit.

Effect of Channels on Backups

If you back up a Replica that operates with more than one channel, say 30, you can only restore that backup on another Replica that operates with the same number of channels. Otherwise, reloading the backup is the same as changing the number of channels without a clean offline.

When operating Tungsten Replicator in a Tungsten cluster, you should always set the number of channels to be the same for all replicators. Otherwise you may run into problems if you try to restore backups across MySQL instances that load with different locations.

If the replicator has only a single channel enabled, you can restore the backup anywhere. The same applies if you run the backup after the replicator has been taken offline cleanly.

4.1.4. Parallel Replication and Offline Operation

4.1.4.1. Clean Offline Operation

When you issue a trepctl offline command, Tungsten Replicator will bring all channels to the same point in the log and then go offline. This is known as going offline cleanly. When a Replica has been taken offline cleanly the following are true:

- The `trep_commit_seqno` [693] table contains a single row
- The `trep_shard_channel` [694] table is empty

When parallel replication is not enabled, you can take the replicator offline by stopping the replicator process. There is no need to issue a trepctl offline command first.

4.1.4.2. Tuning the Time to Go Offline Cleanly

Putting a replicator offline may take a while if the slowest and fastest channels are far apart, i.e., if one channel gets far ahead of another. The separation between channels is controlled by the *maxOfflineInterval* parameter, which defaults to 5 seconds. This sets the allowable distance between commit timestamps processed on different channels. You can adjust this value at installation or later. The following example shows how to change it after installation. This can be done at any time and does not require the replicator to go offline cleanly.

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

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

```
[alpha]
...
property=replicator.store.parallel-queue.maxOfflineInterval=30
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-7.0.3-141

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 10.2.4, “Configuration Changes with an INI file”.

The offline interval is only the the approximate time that Tungsten Replicator will take to go offline. Up to a point, larger values (say 60 or 120 seconds) allow the replicator to parallelize in spite of a few operations that are relatively slow. However, the down side is that going offline cleanly can become quite slow.

4.1.4.3. Unclean Offline

If you need to take a replicator offline quickly, you can either stop the replicator process or issue the following command:

```
shell> trepctl offline -immediate
```

Both of these result in an unclean shutdown. However, parallel replication is completely crash-safe provided you use transactional table types like InnoDB, so you will be able to restart without causing Replica consistency problems.

> **Warning**
>
> You must take the replicator offline cleanly to change the number of channels or when reverting to MySQL native replication. Failing to do so can result in errors when you restart replication.

4.1.5. Adjusting Parallel Replication After Installation

4.1.5.1. How to Enable Parallel Apply After Installation

> **Warning**
>
> Be sure to place the cluster into *MAINTENANCE* mode first so the Manager does not attempt to automatically bring the replicator online.
>
> ```
> cctrl> set policy maintenance
> ```

To enable parallel replication after installation, take the replicator offline cleanly using the following command:

```
shell> trepctl offline
```

Modify the configuration to add two parameters:

```
[defaults]
...
svc-parallelization-type=disk
channels=10
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
```

```
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-7.0.3-141

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 10.2.4, “Configuration Changes with an INI file”.

> **Note**
>
> You make use an actual data service name in place of the keyword `defaults`.

Signal the changes by a complete restart of the Replicator process:

```
shell> replicator restart
```

> **Warning**
>
> Be sure to place the cluster into *`AUTOMATIC`* mode as soon as all replicators are updated and back online.
>
> ```
> cctrl> set policy automatic
> ```

You can check the number of active channels on a Replica by looking at the "channels" property once the replicator restarts.

```
Replica shell> trepctl -service alpha status| grep channels
channels               : 10
```

> **Important**
>
> The channel count for a Primary will ALWAYS be 1 because extraction is single-threaded:
>
> ```
> Primary shell> trepctl -service alpha status| grep channels
> channels               : 1
> ```

> **Warning**
>
> Enabling parallel apply will dramatically increase the number of connections to the database server.
>
> Typically the calculation on a Replica would be: Connections = Channel_Count x Sevice_Count x 2, so for a 4-way Composite Composite Active/Active topology with 30 channels there would be 30 x 4 x 2 = 240 connections required for the replicator alone, not counting application traffic.
>
> You may display the currently used number of connections in MySQL:
>
> ```
> mysql> SHOW STATUS LIKE 'max_used_connections';
> +----------------------+-------+
> | Variable_name        | Value |
> +----------------------+-------+
> | Max_used_connections | 190   |
> +----------------------+-------+
> 1 row in set (0.00 sec)
> ```
>
> Below are suggestions for how to change the maximum connections setting in MySQL both for the running instance as well as at startup:
>
> ```
> mysql> SET GLOBAL max_connections = 512;
>
> mysql> SHOW VARIABLES LIKE 'max_connections';
> +-----------------+-------+
> | Variable_name   | Value |
> +-----------------+-------+
> | max_connections | 512   |
> +-----------------+-------+
> 1 row in set (0.00 sec)
>
> shell> vi /etc/my.cnf
> #max_connections = 151
> max_connections = 512
> ```

4.1.5.2. How to Change Channels Safely

To change the number of channels you must take the replicator offline cleanly using the following command:

```
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` [693] table, you will notice only a single row, which shows that updates to the Replica have been completely serialized to a single point. At this point you may safely reconfigure the number of channels on the replicator, for example using the following command:

```
[alpha]
...
channels=5
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-7.0.3-141

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 10.2.4, “Configuration Changes with an INI file”.

You can check the number of active channels on a Replica by looking at the "channels" property once the replicator restarts.

If you attempt to reconfigure channels without going offline cleanly, Tungsten Replicator will signal an error when you attempt to go online with the new channel configuration. The cure is to revert to the previous number of channels, go online, and then go offline cleanly. Note that attempting to clean up the `trep_commit_seqno` [693] and `trep_shard_channel` [694] tables manually can result in your Replicas becoming inconsistent and requiring full resynchronization. You should only do such cleanup under direction from Continuent support.

Warning

Failing to follow the channel reconfiguration procedure carefully may result in your Replicas becoming inconsistent or failing. The cure is usually full resynchronization, so it is best to avoid this if possible.

4.1.5.3. How to Disable Parallel Replication Safely

The following steps describe how to gracefully disable parallel apply replication.

Replication Graceful Offline (critical first step)

To disable parallel apply, you must first take the replicator offline cleanly using the following command:

```
shell> trepctl offline
```

This command brings all channels up the same transaction in the log, then goes offline. If you look in the `trep_commit_seqno` [693] table, you will notice only a single row, which shows that updates to the Replica have been completely serialized to a single point. At this point you may safely disable parallel apply on the replicator, for example using the following command:

```
[alpha]
...
svc-parallelization-type=none
channels=1
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-7.0.3-141

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 10.2.4, “Configuration Changes with an INI file”.

Verification

You can check the number of active channels on a Replica by looking at the "channels" property once the replicator restarts.

```
shell> trepctl -service alpha status| grep channels
channels               : 1
```

Notes and Warnings

If you attempt to reconfigure channels without going offline cleanly, Tungsten Replicator will signal an error when you attempt to go online with the new channel configuration. The cure is to revert to the previous number of channels, go online, and then go offline cleanly. Note that attempting to clean up the `trep_commit_seqno` [693] and `trep_shard_channel` [694] tables manually can result in your Replicas becoming inconsistent and requiring full resynchronization. You should only do such cleanup under direction from Continuent support.

Warning

Failing to follow the channel reconfiguration procedure carefully may result in your Replicas becoming inconsistent or failing. The cure is usually full resynchronization, so it is best to avoid this if possible.

4.1.5.4. How to Switch Parallel Queue Types Safely

As with channels you should only change the parallel queue type after the replicator has gone offline cleanly. The following example shows how to update the parallel queue type after installation:

```
[alpha]
...
svc-parallelization-type=disk
channels=5
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-7.0.3-141

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 10.2.4, “Configuration Changes with an INI file”.

4.1.6. Monitoring Parallel Replication

Basic monitoring of a parallel deployment can be performed using the techniques in Chapter 6, *Operations Guide*. Specific operations for parallel replication are provided in the following sections.

4.1.6.1. Useful Commands for Parallel Monitoring Replication

The replicator has several helpful commands for tracking replication performance:

Command	Description
trepctl status	Shows basic variables including overall latency of Replica and number of apply channels
trepctl status -name shards [388]	Shows the number of transactions for each shard
trepctl status -name stores [389]	Shows the configuration and internal counters for stores between tasks
trepctl status -name tasks [390]	Shows the number of transactions (events) and latency for each independent task in the replicator pipeline

4.1.6.2. Parallel Replication and Applied Latency On Replicas

The trepctl status appliedLastSeqno parameter shows the sequence number of the last transaction committed. Here is an example from a Replica with 5 channels enabled.

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

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

- Busy channels mark their position in table trep_commit_seqno [693] as they commit. These are up-to-date with the traffic on that channel, but channels have latency between those that have a lot of big transactions and those that are more lightly loaded.

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

For systems with few transactions it is useful to lower the synchronization interval to a smaller number of transactions, for example 500. The following command shows how to adjust the synchronization interval after installation:

```
[alpha]
...
property=replicator.store.parallel-queue.syncInterval=500
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-7.0.3-141

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 10.2.4, “Configuration Changes with an INI file”.

Note that there is a trade-off between the synchronization interval value and writes on the DBMS server. With the foregoing setting, all channels will write to the `trep_commit_seqno` [693] 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 [389] 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.

4.1.6.3. Relative Latency

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

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

In this example the last transaction had a latency of .571 seconds from the time it committed on the Primary and committed 8.944 seconds ago. If relative latency increases significantly in a busy system, it may be a sign that replication is stalled. This is a good parameter to check in monitoring scripts.

4.1.6.4. Serialization Count

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

You can detect the number of transactions that have been serialized by looking at the *`serializationCount`* parameter using the trepctl status -name stores [389] 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.

4.1.6.5. Maximum Offline Interval

The maximum offline interval (*maxOfflineInterval*) parameter controls the "distance" between the fastest and slowest channels when parallel apply is enabled. The replicator measures distance using the seconds between commit times of the last transaction processed on each channel. This time is roughly equivalent to the amount of time a replicator will require to go offline cleanly.

You can change the *maxOfflineInterval* as shown in the following example, the value is defined in seconds.

```
[alpha]
...
property=replicator.store.parallel-queue.maxOfflineInterval=30
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-7.0.3-141

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 10.2.4, “Configuration Changes with an INI file”.

You can view the configured value as well as the estimate current value using the trepctl status -name stores [389] command, as shown in yet another example:

```
shell> trepctl status -name stores
Processing status command (stores)...
NAME                      VALUE
----                      -----
...
estimatedOfflineInterval: 1.3
...
maxOfflineInterval      : 30
...
Finished status command (stores)...
```

4.1.6.6. Workload Distribution

Parallel apply works best when transactions are distributed evenly across shards and those shards are distributed evenly across available channels. You can monitor the distribution of transactions over shards using the trepctl status -name shards [388] command. This command lists transaction counts for all shards, as shown in the following example.

```
shell> trepctl status -name shards
Processing status command (shards)...
...
NAME                VALUE
----                -----
appliedLastEventId: mysql-bin.000211:0000000020095076;0
appliedLastSeqno  : 78023
appliedLatency    : 0.255
eventCount        : 3523
shardId           : cust1
stage             : q-to-dbms
...
```

```
Finished status command (shards)...
```

If one or more shards have a very large *eventCount* value compared to the others, this is a sign that your transaction workload is poorly distributed across shards.

The listing of shards also offers a useful trick for finding serialized transactions. Shards that Tungsten Replicator cannot safely parallelize are assigned the dummy shard ID #UNKNOWN. Look for this shard to find the count of serialized transactions. The *appliedLastSeqno* for this shard gives the sequence number of the most recent serialized transaction. As the following example shows, you can then list the contents of the transaction to see why it serialized. In this case, the transaction affected tables in different schemas.

```
shell> trepctl status -name shards
Processing status command (shards)...
NAME                VALUE
----                -----
appliedLastEventId: mysql-bin.000211:0000000020095529;0
appliedLastSeqno  : 78026
appliedLatency    : 0.558
eventCount        : 26
shardId           : #UNKNOWN
stage             : q-to-dbms
...
Finished status command (shards)...
shell> thl list -seqno 78026
SEQ# = 78026 / FRAG# = 0 (last frag)
- TIME = 2025-01-17 22:29:42.0
- EPOCH# = 1
- EVENTID = mysql-bin.000211:0000000020095529;0
- SOURCEID = logos1
- METADATA = [mysql_server_id=1;service=percona;shard=#UNKNOWN]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [##charset = ISO8859_1, autocommit = 1, sql_auto_is_null = 0, »
    foreign_key_checks = 1, unique_checks = 1, sql_mode = '', character_set_client = 8, »
    collation_connection = 8, collation_server = 33]
- SCHEMA =
- SQL(0) = insert into mats_0.foo values(1) /* ___SERVICE___ = [percona] */
- OPTIONS = [##charset = ISO8859_1, autocommit = 1, sql_auto_is_null = 0, »
    foreign_key_checks = 1, unique_checks = 1, sql_mode = '', character_set_client = 8, »
    collation_connection = 8, collation_server = 33]
- SQL(1) = insert into mats_1.foo values(1)
```

The replicator normally distributes shards evenly across channels. As each new shard appears, it is assigned to the next channel number, which then rotates back to 0 once the maximum number has been assigned. If the shards have uneven transaction distributions, this may lead to an uneven number of transactions on the channels. To check, use the trepctl status -name tasks [390] and look for tasks belonging to the q-to-dbms stage.

```
shell> trepctl status -name tasks
Processing status command (tasks)...
...
NAME                VALUE
----                -----
appliedLastEventId: mysql-bin.000211:0000000020095076;0
appliedLastSeqno  : 78023
appliedLatency    : 0.248
applyTime         : 0.003
averageBlockSize  : 2.520
cancelled         : false
currentLastEventId: mysql-bin.000211:0000000020095076;0
currentLastFragno : 0
currentLastSeqno  : 78023
eventCount        : 5302
extractTime       : 274.907
filterTime        : 0.0
otherTime         : 0.0
stage             : q-to-dbms
state             : extract
taskId            : 0
...
Finished status command (tasks)...
```

If you see one or more channels that have a very high *eventCount*, consider either assigning shards explicitly to channels or redistributing the workload in your application to get better performance.

4.1.7. Controlling Assignment of Shards to Channels

Tungsten Replicator by default assigns channels using a round robin algorithm that assigns each new shard to the next available channel. The current shard assignments are tracked in table trep_shard_channel [694] 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` [694] table persist if either the DBMS or the replicator fails.

It is possible to override the default assignment by updating the `shard.list` file found in the `tungsten-replicator/conf` directory. This file normally looks like the following:

```
# SHARD MAP FILE.
# This file contains shard handling rules used in the ShardListPartitioner
# class for parallel replication.  If unchanged shards will be hashed across
# available partitions.

# You can assign shards explicitly using a shard name match, where the form
# is <db>=<partition>.
#common1=0
#common2=0
#db1=1
#db2=2
#db3=3

# Default partition for shards that do not match explicit name.
# Permissible values are either a partition number or -1, in which
# case values are hashed across available partitions.  (-1 is the
# default.
#(*)=-1

# Comma-separated list of shards that require critical section to run.
# A "critical section" means that these events are single-threaded to
# ensure that all dependencies are met.
#(critical)=common1,common2

# Method for channel hash assignments.  Allowed values are round-robin and
# string-hash.
(hash-method)=round-robin
```

You can update the shard.list file to do three types of custom overrides.

1. Change the hashing method for channel assignments. Round-robin uses the `trep_shard_channel` [694] table. The string-hash method just hashes the shard name.
2. Assign shards to explicit channels. Add lines of the form *`shard=channel`* to the file as shown by the commented-out entries.
3. Define critical shards. These are shards that must be processed in serial fashion. For example if you have a sharded application that has a single global shard with reference information, you can declare the global shard to be critical. This helps avoid applications seeing out of order information.

Changes to shard.list must be made with care. The same cautions apply here as for changing the number of channels or the parallelization type. For subscription customers we strongly recommend conferring with Continuent Support before making changes.

4.1.8. Disk vs. Memory Parallel Queues

Channels receive transactions through a special type of queue, known as a parallel queue. Tungsten offers two implementations of parallel queues, which vary in their performance as well as the requirements they may place on hosts that operate parallel apply. You choose the type of queue to enable using the `svc-parallelization-type` [516] 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:

```
[alpha]
...
property=replicator.store.parallel-queue.maxSize=10000
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-7.0.3-141

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 10.2.4, “Configuration Changes with an INI file”.

You may need to increase the Java VM heap size when you increase the parallel queue maximum size. Use the `java-mem-size` [496] option on the tpm command for this purpose or edit the Replicator `wrapper.conf` file directly.

Warning

Memory queues are not recommended for production use at this time. Use disk queues.

4.2. Starting and Stopping Tungsten Components

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

```
shell> stopall
Stopping Tungsten Connector...
Stopped Tungsten Connector.
Stopping Tungsten Replicator Service...
Stopped Tungsten Replicator Service.
Stopping Tungsten Manager Service...
Stopped Tungsten Manager Service.
```

To start all services, use the startall script:

```
shell> startall
Starting Tungsten Manager Service...

Starting Tungsten Replicator Service...

Starting Tungsten Connector...
```

4.2.1. Stopping/Starting Tungsten Replicator

Warning

Stopping, or taking offline, a running replicator temporarily stops and restarts replication. Either set *MAINTENANCE* mode within cctrl [see Section 6.16, “Performing Database or OS Maintenance” or shun the datasource before stopping the replicator if you need to ensure it does not restart [Section 6.3.7.1, “Shunning a Datasource”.

Warning

Before stopping a replicator it is good practice to take it offline cleanly first. Doing so will ensure a clean stop of any replication in flight and ensure replication positions are correctly recorded.

If Parallel Apply is in use, this is essential, see Section 4.1.4, "Parallel Replication and Offline Operation" for more information

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

```
shell> trepctl -all-services offline
shell> replicator stop
Stopping Tungsten Replicator Service...
Stopped Tungsten Replicator Service.
```

To start the Tungsten Replicator service if it is not already running:

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

To restart the Tungsten Replicator service if it is running:

```
shell> trepctl -all-services offline
shell> replicator restart
Stopping Tungsten Replicator Service...
Stopped Tungsten Replicator Service.
Starting Tungsten Replicator Service...
```

To start the Tungsten Replicator service if it is not already running, but ensure replication is offline:

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

4.2.2. Stopping/Starting Tungsten Connectors

Warning

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

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

```
shell> connector stop
Stopping Tungsten Connector Service...
Stopped Tungsten Connector Service.
```

To start the Tungsten Connector service if it is not already running:

```
shell>  connector start
Starting Tungsten Connector Service...
Waiting for Tungsten Connector Service.....
running: PID:12338
```

To restart the Tungsten Connector service if it is running:

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

If the cluster was configured with `auto-enable=false` [474] then you will need to put each node online individually.

4.2.3. Stopping/Starting Tungsten Managers

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

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

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

1. To set the *MAINTENANCE* policy mode:

   ```
   [LOGICAL:EXPERT] /dsone > set policy maintenance
   ```

2. On each datasource in the dataservice:

a. Stop the service:

```
shell> manager stop
```

b. Then start the manager service:

```
shell> manager start
```

c. Or:

```
shell> manager restart
```

3. Once all the managers have been restarted, set the policy mode back to *AUTOMATIC*:

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

4.2.4. Restarting the Multi-Site/Active-Active Replicator Service

Warning

Restarting a running replicator temporarily stops and restarts replication. When using Multi-Site/Active-Active, restarting the additional replicator will stop replication between sites.

These instructions assume you have installed the additional replicator with the `executable-prefix=mm` [493] option. If not, you should go to `/opt/replicator/tungsten/tungsten-replicator/bin` and run the replicator and trepctl commands directly.

Warning

Before stopping a replicator it is good practice to take it offline cleanly first. Doing so will ensure a clean stop of any replication in flight and ensure replication positions are correctly recorded.

If Parallel Apply is in use, this is essential, see Section 4.1.4, “Parallel Replication and Offline Operation” for more information

Warning

The Tungsten Replicator services that connect clusters in a Multi-Site/Active-Active topology are not managed by the individual cluster services, and therefore the cluster will not automatically restart the cross-site replicator services.

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

```
shell> mm_trepctl -all-services offline
shell> mm_replicator stop
Stopping Tungsten Replicator Service...
Stopped Tungsten Replicator Service.
```

To start the Tungsten Replicator service if it is not already running:

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

To restart the Tungsten Replicator service if it is running:

```
shell> mm_trepctl -all-services offline
shell> mm_replicator restart
Stopping Tungsten Replicator Service...
Stopped Tungsten Replicator Service.
Starting Tungsten Replicator Service...
```

To start the Tungsten Replicator service if it is not already running, but ensure replication is offline:

```
shell> mm_replicator start offline
Starting Tungsten Replicator Service...
```

4.3. Configuring Startup on Boot

By default, Tungsten Clustering does not start automatically on boot. To enable Tungsten Clustering to start at boot time, use the deployall script provided in the installation directory to create the necessary boot scripts, or include `install=true` [494] within your configuration prior to installation.

> **Note**
>
> If you are using `systemctl` to manage your startup scripts and you have selinux enabled, you will also need to execute the following commands to ensure components start correctly:
>
> ```
> sudo semanage fcontext -a -t bin_t "/opt/continuent(/.*)?"
>
> sudo restorecon -r -v /opt/continuent
> ```
>
> If you have installed the software in an alternative directory, adjust the paths in the exmaple above accordingly

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

```
shell> undeployall
```

4.3.1. Configuring Multi-Site/Active-Active Replicator Startup on Boot

Because there is an additional Tungsten Replicator running, each must be individually configured to startup on boot:

- For the Tungsten Cluster service, use Section 4.3, “Configuring Startup on Boot”.
- For the Tungsten Replicator service, a custom startup script must be created, otherwise the replicator will be unable to start as it has been configured in a different directory.

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

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

  2. Stop the Tungsten Replicator process. Failure to do this will cause issues because the service will no longer recognize the existing PID file and report it is not running.

     ```
     shell> /etc/init.d/mm_replicator stop
     ```

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

     ```
     APP_NAME="mmreplicator"
     ```

  4. Start the Tungsten Replicator process.

     ```
     shell> /etc/init.d/mm_replicator start
     ```

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

     On Debian/Ubuntu:

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

     On RedHat/CentOS:

     ```
     shell> sudo chkconfig --add mm_replicator
     ```

4.4. Removing Datasources, Managers or Connectors

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

In this section:

- Section 4.4.1, “Removing a Datasource from an Existing Deployment”
- Section 4.4.2, “Removing a Connector from an Existing Deployment”

4.4.1. Removing a Datasource from an Existing Deployment

There are really three different ways to remove a service from a composite cluster, (CAP, DAA or CAA):

- Using command tools/tpm update --replace-release

  See Section 4.4.1.1, “Removing a Composite Datasource/Cluster from an Existing Deployment Using tpm update”

- Using command tpm delete-service

See Section 4.4.1.2, “Removing a Composite Datasource/Cluster from an Existing Deployment Using tpm delete-service”

- Manually using individual commands for each required step

  See Section 4.4.1.3, “Removing a Composite Datasource/Cluster from an Existing Deployment Manually”

Internal Operations Reference

> **Important**
>
> For Reference Only
>
> Below are the internal operations taken during the script processing, presented for clarity.
>
> Do NOT perform any step below by hand.

For CAP, tpm update and tpm delete-service both run the Manager Cleanup Steps on each node

For CAA, tpm update and tpm delete-service both run the Manager Cleanup Steps and the Replicator Cleanup Steps on each node

- Manager Cleanup Steps
  - Manager Cleanup Steps performed on a node in a service not being deleted
    - Delete the dataservice from the Manager layer using cctrl
    - Remove the matching service line from `cluster-home/conf/dataservices.properties`
  - Manager Cleanup Steps performed on a node in service to be removed
    - Stop all Tungsten services
    - Run the tpm uninstall command
- Replicator Cleanup Steps
  - Replicator Cleanup Steps performed on a node in a service not being deleted
    - Offline all Tungsten Replicator services
    - Remove the two `tungsten/tungsten-replicator/conf/static-*_from_{service}.properties*` files
    - Restart the Tungsten Replicator process
    - Online all Tungsten Replicator services
    - Delete the service-specific thl and relay subdirs
  - Replicator Cleanup Steps performed on a node in service to be removed
    - Offline all Tungsten Replicator services
    - Remove the two `tungsten/tungsten-replicator/conf/static-*_from_{service}.properties*` files

4.4.1.1. Removing a Composite Datasource/Cluster from an Existing Deployment Using tpm update

To remove an entire composite datasource (cluster) from an existing deployment using the tpm update command, you must perform the following steps on every node in the composite cluster, including the ones to be removed.

1. Edit the `tungsten.ini` file on every node in the composite cluster you are keeping
2. Delete the entire stanza for the cluster service you wish to remove
3. Delete the cluster name from the composite service definition
4. Save and exit the `tungsten.ini` file
5. Execute the ./tools/tpm update --replace-release command from the staging directory.

   For more information about running tpm update, please see Section 10.2.4, “Configuration Changes with an INI file”.

Once the procedure is complete on all nodes, the service will no longer be visible in cctrl.

4.4.1.2. Removing a Composite Datasource/Cluster from an Existing Deployment Using tpm delete-service

To remove an entire composite datasource (cluster) from an existing deployment using the tpm delete-service command, you must perform the following steps on every node in the composite cluster, including the ones to be removed.

1. Execute the tpm delete-service {service_name_here} command
2. During the execution of the tpm command on every node in the composite cluster that you are keeping, you will be prompted to edit the `tungsten.ini` file

   This step will not be called if the node is due to be de-commissioned (i.e. the service has been deleted).
3. Delete the entire stanza for the cluster service you wish to remove
4. Delete the cluster name from the composite service definition
5. Save and exit the `tungsten.ini` file
6. tpm will complete the service deletion based on your edits. Once this is done on all of the nodes, the service will be gone from cctrl.

4.4.1.3. Removing a Composite Datasource/Cluster from an Existing Deployment Manually

While it is possible to manually remove a service from a Composite Cluster, there are many steps involved. Continuent recommends using either tpm update or tpm delete-service to do so.

For tpm update, see Section 4.4.1.1, “Removing a Composite Datasource/Cluster from an Existing Deployment Using tpm update”

For tpm delete-service, see Section 4.4.1.2, “Removing a Composite Datasource/Cluster from an Existing Deployment Using tpm delete-service”

To manually remove an entire composite datasource (cluster) from an existing deployment there are two primary stages, removing the child service from the composite parent cluster inside cctrl, and then removing the Tungsten software from the nodes to be decommissioned in the cluster service being deleted.

For CAA clusters, additional manual steps at the replicator layer will also be required.

For example, to remove cluster `west` from a composite dataservice:

1. Check the current service state:

```
shell> cctrl -multi
    [LOGICAL] / > ls

    +----------------------------------------------------------------------------+
    |DATA SERVICES:                                                              |
    +----------------------------------------------------------------------------+
    east
    global
    west

    [LOGICAL] / > use global
    [LOGICAL] /global > ls

    COORDINATOR[db1:AUTOMATIC:ONLINE]

    DATASOURCES:
    +----------------------------------------------------------------------------+
    |east(composite master:ONLINE)                                               |
    |STATUS [OK] [2017/05/16 01:25:31 PM UTC]                                    |
    +----------------------------------------------------------------------------+

    +----------------------------------------------------------------------------+
    |west(composite slave:ONLINE)                                                |
    |STATUS [OK] [2017/05/16 01:25:30 PM UTC]                                    |
    +----------------------------------------------------------------------------+
```

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

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

3. Remove the composite member cluster from the composite service using the drop command.

```
[LOGICAL] /global > drop composite datasource west
    COMPOSITE DATA SOURCE 'west@global' WAS DROPPED
```

```
[LOGICAL] /global > ls

COORDINATOR[db1:AUTOMATIC:ONLINE]

DATASOURCES:
+---------------------------------------------------------------------------+
|east(composite master:ONLINE)                                              |
|STATUS [OK] [2017/05/16 01:25:31 PM UTC]                                   |
+---------------------------------------------------------------------------+

[LOGICAL] /global > cd /
[LOGICAL] / > ls

+---------------------------------------------------------------------------+
|DATA SERVICES:                                                             |
+---------------------------------------------------------------------------+
east
global
```

4. If the removed composite datasource still appears in the top-level listing, then you will need to clean up by hand. For example:

```
[LOGICAL] /global > cd /
    [LOGICAL] / > ls

    +---------------------------------------------------------------------------+
    |DATA SERVICES:                                                             |
    +---------------------------------------------------------------------------+
    east
    global
    west
```

   Stop all managers on all nodes at the same time

```
shell > manager stop
```

```
shell > vim $CONTINUENT_HOME/cluster-home/conf/dataservices.properties

    Before:
    east=db1,db2,db3
    west=db4,db5,db6

    After:
    east=db1,db2,db3
```

   Start all managers one-by-one, starting with the current Primary

```
shell > manager start
```

   Once all managers are running, check the list again:

```
shell> cctrl -multi
    [LOGICAL] / > ls

    +---------------------------------------------------------------------------+
    |DATA SERVICES:                                                             |
    +---------------------------------------------------------------------------+
    east
    global
```

5. Switch to *AUTOMATIC* policy mode:

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

Now the cluster has been removed from the composite dataservice, the services on the old nodes must be stopped and then removed from the configuration.

1. Stop the running services on all nodes in the removed cluster:

```
shell> stopall
```

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

   - If you are using staging directory method with tpm:

     a. Change to the staging directory. The current staging directory can be located using tpm query staging:

```
shell> tpm query staging
```

```
    tungsten@host1:/home/tungsten/tungsten-clustering-7.0.3-141
    shell> cd /home/tungsten/tungsten-clustering-7.0.3-141
```

b. Update the configuration, omitting the cluster datasource name from the list of members of the dataservice:

```
shell> tpm update global --composite-datasources=east
```

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

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

3. Stop the replicator/manager from being started again.
   - If this all the services on the this node, replicator, manager and connector are being removed, remove the Tungsten Cluster installation entirely:
     - Remove the startup scripts from your server:

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

     - Remove the installation directory:

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

   - If the replicator/manager has been installed on a host but the connector is not being removed, remove the start scripts to prevent the services from being automatically started:

```
shell> rm /etc/init.d/tmanager
    shell> rm /etc/init.d/treplicator
```

Replicator Cleanup Steps for CAA Clusters

- Replicator Cleanup Steps performed on a node in a service not being deleted
  - Offline all Tungsten Replicator services
  - Remove the two `tungsten/tungsten-replicator/conf/static-*_from_{service}.properties*` files
  - Restart the Tungsten Replicator process
  - Online all Tungsten Replicator services
  - Delete the service-specific thl and relay subdirs
- Replicator Cleanup Steps performed on a node in service to be removed
  - Offline all Tungsten Replicator services
  - Remove the two `tungsten/tungsten-replicator/conf/static-*_from_{service}.properties*` files

4.4.2. Removing a Connector from an Existing Deployment

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

For example, to remove host4 from the current dataservice:

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

```
shell> connector stop
```

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

```
shell> tpm query staging
tungsten@host1:/home/tungsten/tungsten-clustering-7.0.3-141
```

```
shell> cd /home/tungsten/tungsten-clustering-7.0.3-141
```

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

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

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

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

4. Stop the connector from being started again. If the connector is restarted, it will connect to the previously configured Primary and begin operating again.

- If this is a standalone Connector installation, remove the Tungsten Cluster installation entirely:
  - Remove the startup scripts from your server:

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

  - Remove the installation directory:

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

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

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

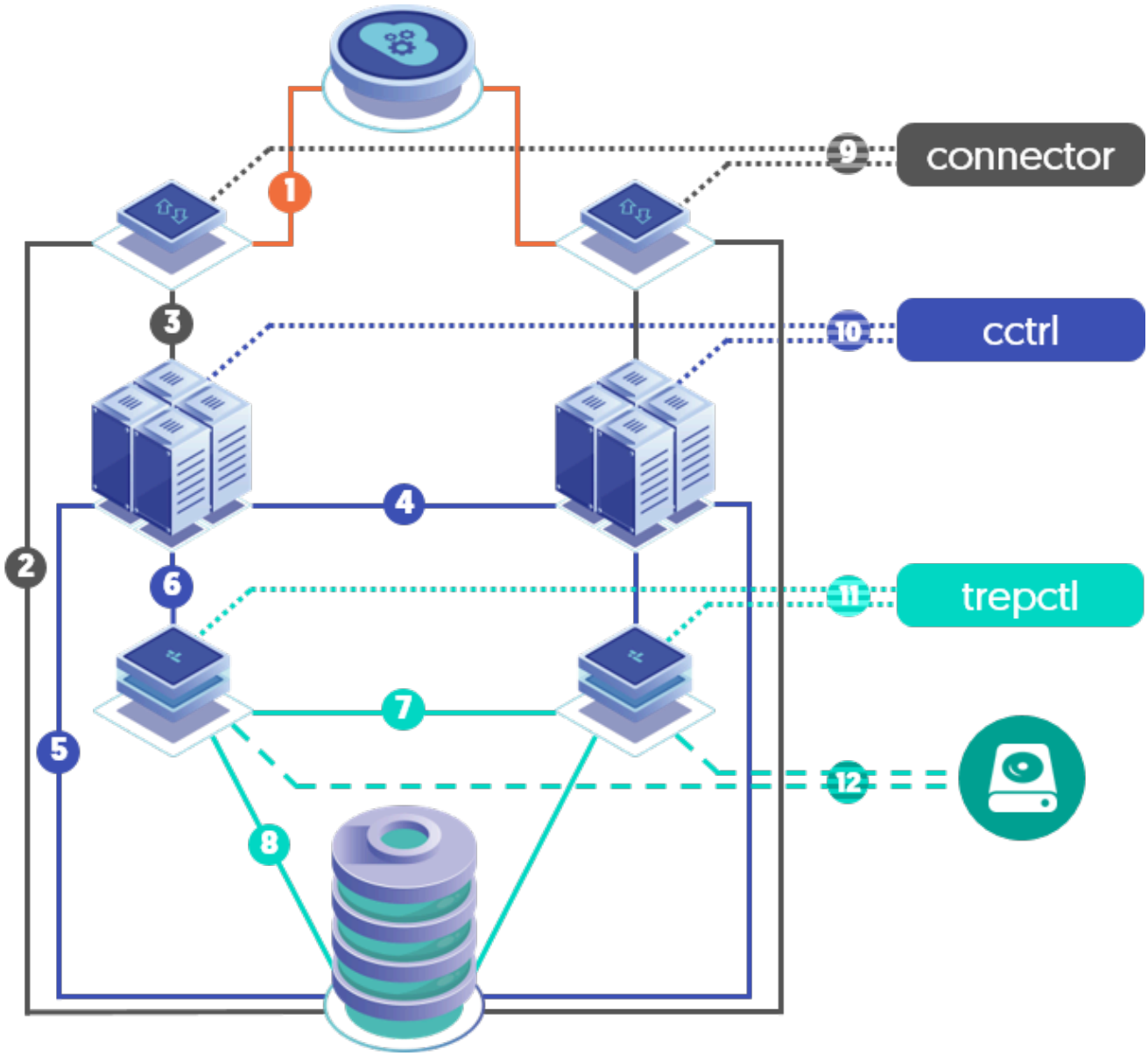
Chapter 5. Deployment: Security

Tungsten Cluster supports SSL, TLS and certificates for both communication and authentication for all components within the system, and to the underlying databases. This security is enabled by default and includes:

- Authentication between command-line tools (cctrl), and between background services.
- SSL/TLS between command-line tools and background services.
- SSL/TLS between Tungsten Replicator and datasources.
- SSL/TLS between Managers and datasources.
- SSL/TLS between Tungsten Connector and datasources.
- SSL for all API calls.
- File permissions and access by all components.

The following graphic provides a visual representation of the various communication channels which may be encrypted.

Figure 5.1. Security Internals: Cluster Communication Channels



For the key to the above diagram, please see Section 10.5.20, “tpm report Command”.

If you are using a single staging directory to handle your complete installation, tpm will automatically create the necessary certificates for you. If you are using an INI based installation, then the installation process will create the certificates for you, however you will need to manually sync them between hosts prior to starting the various components.

It is assumed that your underlying database has SSL enabled and the certificates are available. If you need, and want, this level of security enabling, you can refer to Section 5.13.1, “Enabling Database SSL” for the steps required.

> **Important**
>
> Due to a known issue in earlier Java revisions that may cause performance degradation with client connections, it is strongly advised that you ensure your Java version is one of the following MINIMUM releases before enabling SSL:
>
> - Oracle JRE 8 Build 261
> - Oracle JRE 11 Build 8
> - OpenJDK 8 Build 222

5.1. Enabling Security

By default, security is enabled for new installations.

Security can be enabled/disabled by adding the `disable-security-controls` [490] option to the configuration.

If this property is not supplied, or set to false, then security will be enabled. If set to true, then security will be disabled.

Enabling security through this single option, has the same effect as adding:

- `--file-protection-level=0027` [493]
- `--rmi-ssl=true` [492]
- `--thl-ssl=true` [493]
- `--rmi-authentication=true` [492]
- `--jgroups-ssl=true` [492]
- `--datasource-enable-ssl=true` [486]
- `--connector-ssl=true` [491]
- `--connector-rest-api-ssl=true` [483]
- `--manager-rest-api-ssl=true` [498]
- `--replicator-rest-api-ssl=true` [512]

> **Important**
>
> If you are enabling to-the-database encryption, you must ensure this has been enabled in your database and the relevant certificates are available first. See Section 5.13.1, “Enabling Database SSL” for steps.

> **Important**
>
> Installing from a staging host will automatically generate certificates and configuration for a secured installation. No further changes or actions are required.
>
> For INI-based installations, there are additional steps required to copy the needed certificate files to all of the nodes. Please see Section 5.1.2, “Enabling Security using the INI Method” for details.

5.1.1. Enabling Security using the Staging Method

Security will be enabled during initial install by default, should you choose to disable at install, then these steps will guide you in the process to enable as part of a post-install update

Enabled During Install

As mentioned, security is enabled by default. This is controlled by the `--disable-security-controls=false` [490]. If not supplied, the default is false. You can choose to specify this in your configuration for transparency if you wish.

```
shell> tools/tpm configure defaults --disable-security-controls=false \
[...the rest of the configuration options...]
shell> tools/tpm install
```

The above configuration (and the default) will assume that your database has been configured with SSL enabled. The installation will error and fail if this is not the case. You must manually ensure database SSL has been enabled prior to issuing the install. Steps to enable this can be found in Section 5.13.1, "Enabling Database SSL"

If you DO NOT want to enable database level SSL, then you must also include the following option in the tpm configure command above:

```
--enable-connector-ssl=false
--datasource-enable-ssl=false
```

Important

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

Enabling Post-Installation

If, at install time, you disabled security (by specifying `--disable-security-controls=true` [490]) you can enable it by changing the value to false.

```
shell> tools/tpm configure defaults --disable-security-controls=false
shell> tools/tpm update --replace-jgroups-certificate --replace-tls-certificate --replace-release
```

The above configuration will assume that your database has been configured with SSL enabled. The update will error and fail if this is not the case. You must manually ensure database SSL has been enabled prior to issuing the update. Steps to enable this can be found in Section 5.13.1, "Enabling Database SSL"

If you DO NOT want to enable database level SSL, then you must also include the following options in the tpm configure command above:

```
--enable-connector-ssl=false
--datasource-enable-ssl=false
```

Following the update, you will also need to manually re-sync the certificates and keystores to all other nodes within your configuration. The following example uses scp for the copy and uses db1 as the primary source for the files to be copied. Adjust accordingly for your environment.

1. Place the cluster into *MAINTENANCE* mode

   ```
   cctrl> set policy maintenance
   ```

2. Sync Certificates and Keystores to all nodes

   ```
   db1> for host in db2 db3 db4 db5 db6; do
   scp /opt/continuent/share/[jpt]* ${host}:/opt/continuent/share
   scp /opt/continuent/share/.[jpt]* ${host}:/opt/continuent/share
   done
   ```

3. Restart all components, on all hosts

   ```
   shell> manager restart
   shell> replicator restart
   shell> connector restart
   ```

4. Place the cluster back to *AUTOMATIC* mode

   ```
   cctrl> set policy automatic
   ```

Warning

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

5.1.2. Enabling Security using the INI Method

Security will be enabled during initial install by default, should you choose to disable at install, then these steps will guide you in the process to enable as part of a post-install update

Enabled During Install

As mentioned, security is enabled by default. This is controlled by the `disable-security-controls` [490] property. If not supplied, the default is false. You can choose to specify this in your configuration for transparency if you wish.

```
disable-security-controls=false [490]
```

The above configuration (and the default) will assume that your database has been configured with SSL enabled. The installation will error and fail if this is not the case. You must manually ensure database SSL has been enabled prior to issuing the install. Steps to enable this can be found in Section 5.13.1, “Enabling Database SSL”

If you DO NOT want to enable database level SSL, then you must also include the following options in your `tungsten.ini` file:

```
enable-connector-ssl=false [491]
datasource-enable-ssl=false [486]
```

Following installation there are a few additional steps that will be required before starting the software.

- You must select one of the nodes and copy that node's certificate/keystore/truststore files to all other nodes.

  For example, assuming you choose db1, and have 5 other nodes to copy the files to you could use this syntax:

  ```
  shell> for host in db2 db3 db4 db5 db6; do
  scp /opt/continuent/share/[jpt]* ${host}:/opt/continuent/share/
  scp /opt/continuent/share/.[jpt]* ${host}:/opt/continuent/share/
  done
  ```

  > **Important**
  >
  > The above example assumes ssh has been setup between nodes as the tungsten OS user. If this is not the case you will need to use whichever methods you have available to sync these files.

- Then, on all nodes, you can start the software:

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

Enabling Post-Installation

If, at install time, you disabled security (by specifying `disable-security-controls=true` [490] you can enable it by changing the value to false in your `tungsten.ini` on all nodes.

The above configuration (and the default) will assume that your database has been configured with SSL enabled. The update will error and fail if this is not the case. You must manually ensure database SSL has been enabled prior to issuing the update. Steps to enable this can be found in Section 5.13.1, “Enabling Database SSL”

If you DO NOT want to enable database level SSL, then you must also include the following options in your `tungsten.ini` file:

```
enable-connector-ssl=false [491]
datasource-enable-ssl=false [486]
```

Before issuing the update, there are a number of additional steps required. These are outlined below:

- First, configure the `tungsten.ini` file as follows:

  ```
  disable-security-controls=false [490]
  start-and-report=false [514]
  ```

- Enable *`MAINTENANCE`* mode on the cluster

  ```
  shell> cctrl
  cctrl> set policy maintenance
  ```

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

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

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

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

For example, assuming you choose db1, and have 5 other nodes to copy the files to you could use this syntax:

```
shell> for host in db2 db3 db4 db5 db6; do
scp /opt/continuent/share/[jpt]* ${host}:/opt/continuent/share/
scp /opt/continuent/share/.[jpt]* ${host}:/opt/continuent/share/
done
```

> **Important**
>
> The above example assumes ssh has been setup between nodes as the tungsten OS user. If this is not the case you will need to use whichever methods you have available to sync these files.

- On all nodes:

```
shell> startall
```

5.2. Disabling Security

There may be situations where you wish to disable securityfor the entire installation.

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

```
--disable-security-controls=true [490]
```

Disabling security through this single option, has the same effect as adding:

- `--file-protection-level=none` [493]

  Disables file level protection, including ownership and file mode settings.

- `--rmi-ssl=false` [492]

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

- `--thl-ssl=false` [493]

  Disables the use of SSL/TLS for THL transmission between replicators.

- `--rmi-authentication=false` [492]

  Disables the use of authentication when accessing and controlling services.

- `--jgroups-ssl=false` [492]

  Disables SSL/TLS for group communication within the cluster.

- `--datasource-enable-ssl=false` [486]

- `--connector-ssl=false` [491]

  Disables SSL within the connectors

- `--connector-rest-api-ssl=false` [483]

  Disables SSL for communication with the Connector API. This does not disable the API altogether. To do that, refer to `connector-rest-api` [482]

- `--manager-rest-api-ssl=false` [498]

  Disables SSL for communication with the Manager API. This does not disable the API altogether. To do that, refer to `manager-rest-api` [498]

- `--replicator-rest-api-ssl=false` [512]

  Disables SSL for communication with the Replicator API. This does not disable the API altogether. To do that, refer to `replicator-rest-api` [511]

5.3. Creating Suitable Certificates

By default, tpm can automatically create suitable certificates and configuration for use in your deployment. To create the required certificates by hand, use one of the following procedures.

5.3.1. Creating Tungsten Internal Certificates Manually

To manually generate the security files, use the steps below:

- Generating a JGroups Certificate

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

```
shell> keytool -genseckey -alias jgroups \
-validity 3650 \
-keyalg Blowfish -keysize 56 -keystore /etc/tungsten/secure/tungsten_jgroups_keystore.jceks \
-storepass mykeystorepass -keypass mykeystorepass \
-storetype JCEKS
```

- Generating a TLS Certificate

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

```
shell> keytool -genkey -alias tls \
-validity 3650 \
-keyalg RSA -keystore /etc/tungsten/secure/tungsten_tls_keystore.jks \
-dname "cn=Continuent, ou=IT, o=Continuent, c=US" \
-storepass mykeystorepass -keypass mykeystorepass
```

5.4. Installing from a Staging Host with Custom Certificates

Follow the steps in Section 5.3, “Creating Suitable Certificates” to create JGroups and TLS certificates.

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

5.4.1. Installing from a Staging Host with Manually-Generated Certificates

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

5.5. Installing via INI File with Custom Certificates

Follow the steps in Section 5.3, “Creating Suitable Certificates” to create JGroups and TLS certificates.

5.5.1. Installing via INI File with Manually-Generated Certificates

1. Transfer the generated certificates to the same path on all hosts.
2. Update your configuration to specify these certificates and the keystore password:

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

5.6. Installing via INI File with CA-Signed Certificates

- This procedure will take a signed certificate from a known Certificate Authority and use it as the basis for all SSL operations within the cluster, not including Connector client-server SSL, which is configured separately. Please visit Section 5.13.3, “Configuring Connector SSL” for more information about configuring Connector SSL.
- The below example procedure assumes that you have an existing, installed and running cluster with security enabled by setting `disable-security-controls=false` [490]

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

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

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

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

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

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

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

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

> **Important**
>
> When using OpenSSL 3.0 with Java 1.8, you MUST add the `-legacy` option to the openssl command.

- Create a pkcs12-based keystore (.jks) version of the signed certificate:

```
shell> keytool -importkeystore -deststorepass tungsten -destkeystore /etc/tungsten/secure/tungsten_keystore.jks \
  -srckeystore ~/certs/tungsten_sec.crt.p12 -srcstoretype pkcs12 -deststoretype pkcs12
Importing keystore /home/tungsten/certs/tungsten_sec.crt.p12 to /etc/tungsten/secure/tungsten_keystore.jks...
Enter source keystore password:  tungsten
Entry for alias replserver successfully imported.
Import command completed:  1 entries successfully imported, 0 entries failed or cancelled
```

- Import the Certificate Authority's certificate into the keystore:

```
shell> keytool -import -alias careplserver -file ~/certs/ca.crt.pem -keypass tungsten \
  -keystore /etc/tungsten/secure/tungsten_keystore.jks -storepass tungsten
...
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

- Import the Certificate Authority's intermediate certificate (if supplied) into the keystore:

```
shell> keytool -import -alias interreplserver -file ~/certs/int.crt.pem -keypass tungsten \
  -keystore /etc/tungsten/secure/tungsten_keystore.jks -storepass tungsten
Certificate was added to keystore
```

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

```
shell> keytool -export -alias replserver -file ~/certs/client.cer \
  -keystore /etc/tungsten/secure/tungsten_keystore.jks
Enter keystore password:  tungsten
Certificate stored in file </home/tungsten/certs/client.cer>
```

- Create the truststore:

```
shell> keytool -import -trustcacerts -alias replserver -file ~/certs/client.cer \
  -keystore /etc/tungsten/secure/tungsten_truststore.ts -storepass tungsten -noprompt
Certificate was added to keystore
```

- Create the rmi_jmx password store entry:

```
shell> tpasswd -c tungsten tungsten -t rmi_jmx -p /etc/tungsten/secure/passwords.store -e \
  -ts /etc/tungsten/secure/tungsten_truststore.ts -tsp tungsten
Using parameters:
----------------
security.properties    = /opt/continuent/tungsten/cluster-home/../cluster-home/conf/security.properties
password.file.location    = /etc/tungsten/secure/passwords.store
encrypted.password    = true
truststore.location    = /etc/tungsten/secure/tungsten_truststore.ts
truststore.password    = *********
----------------
Creating non existing file: /etc/tungsten/secure/passwords.store
User created successfuly: tungsten
```

- Create the tls password store entry:

```
shell> tpasswd -c tungsten tungsten -t unknown -p /etc/tungsten/secure/passwords.store -e \
  -ts /etc/tungsten/secure/tungsten_truststore.ts -tsp tungsten
Using parameters:
```

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

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

```
shell> tpasswd -l -p /etc/tungsten/secure/passwords.store -ts /etc/tungsten/secure/tungsten_truststore.ts
Using parameters:
-----------------
security.properties    = /opt/continuent/tungsten/cluster-home/../cluster-home/conf/security.properties
password.file.location    = ./passwords.store
encrypted.password    = true
truststore.location    = ./tungsten_truststore.ts
truststore.password    = ********
-----------------
Listing users by application type:

[unknown]
-----------
tungsten

[rmi_jmx]
-----------
tungsten
```

- On host db1, transfer the generated certificates to the same path on all remaining hosts:

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

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

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

> **Important**
>
> When `java-keystore-path` [496] is passed to tpm, the keystore must contain both tls and mysql certs when appropriate. tpm will NOT add mysql cert nor generate tls cert when this flag is found, so both certs must be manually imported already.

- On one node only, enable *`MAINTENANCE`* mode:

```
cctrl> set policy maintenance
```

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

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

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

- On one node only, enable *`AUTOMATIC`* mode and check cluster status:

```
shell> cctrl
Tungsten Clustering 6.0.5 build 40
alpha: session established, encryption=true, authentication=true
cctrl> set policy automatic
cctrl> ls
COORDINATOR[db1:AUTOMATIC:ONLINE]

ROUTERS:
```

```
+----------------------------------------------------------------------------+
|connector@db1[9871](ONLINE, created=0, active=0)                            |
|connector@db2[27930](ONLINE, created=0, active=0)                           |
|connector@db3[23727](ONLINE, created=0, active=0)                           |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|db1(master:ONLINE, progress=1, THL latency=0.656)                           |
|STATUS [OK] [2019/06/06 12:48:11 PM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db2(slave:ONLINE, progress=1, latency=9.858)                                |
|STATUS [OK] [2019/06/06 12:48:11 PM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db3(slave:ONLINE, progress=1, latency=19.235)                               |
|STATUS [OK] [2019/06/06 12:48:10 PM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
```

5.7. Replacing the JGroups Certificate from a Staging Directory

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

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

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

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

Then perform an update and replace the entire release directory:

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

5.8. Replacing the TLS Certificate from a Staging Directory

If you meet the requirements to use an automatically generated certificate from the staging directory, the tpm update command can handle the certificate replacement. Simply add the `--replace-tls-certificate` [509] option to your command. This will create errors if your staging configuration does not reflect the full list of hosts or if you limit the command to a specific host.

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

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

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

Then perform an update and replace the entire release directory:

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

5.9. Removing JGroups Encryption from a Staging Directory

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

```
shell> tpm configure SERVICE \
```

```
        --jgroups-ssl=false
```

Then perform an update and replace the entire release directory:

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

5.10. Removing JGroups Encryption via INI File

To remove the JGroups encryption from a running cluster:

1. Put the cluster into *MAINTENANCE* mode
2. Update the INI file

   ```
   jgroups-ssl=false
   ```

3. Run tpm update and restart the manager

   ```
   shell> tpm update
   shell> manager restart
   ```

4. Check the cluster through cctrl
5. Put the cluster back into *AUTOMATIC* mode

5.11. 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
```

5.12. Removing TLS Encryption via INI File

To remove the TLS encryption from a running cluster:

1. Put the cluster into *MAINTENANCE* mode
2. Update the INI file

   ```
   thl-ssl=false
   rmi-ssl=false
   rmi-authentication=false
   ```

3. Run tpm update and restart the manager

   ```
   shell> tpm update
   shell> manager restart
   shell> replicator restart
   ```

4. Check the cluster through cctrl
5. Cycle through the connectors and restart them

   ```
   shell> connector restart
   ```

6. Put the cluster back into *AUTOMATIC* mode

5.13. Enabling Tungsten<>Database Security

This section explains how to enable security between the database and various other parts of the topology, including:

- Database server SSL

  This is the first step, and the prerequiste for all the remaining steps. You must have the database server properly configured to support SSL before any of the other procedures will work.

See Section 5.13.1, “Enabling Database SSL”

- Tungsten Replicator to the database server

  This usually happens during the second step, and what allows Tungsten Replicator to communicate securely with the database server.

  See Section 5.13.2, “Configure Tungsten<>Database Secure Communication”

- Tungsten Manager to the database server

  This usually happens during the second step, and what allows Tungsten Manager to communicate securely with the database server.

  See Section 5.13.2, “Configure Tungsten<>Database Secure Communication”

- Connector to the database server

  This is usually the third step, and what allows the Tungsten Connector to communicate securely with the database server.

  See Section 5.13.3.1, “Enable and Test SSL encryption from the Connector to the Database”

- Application to the Database through the Connector

  This is usually the last step, and what allows the Application to communicate securely with the database server through the Tungsten Connector.

  See Section 5.13.3.2, “Test SSL encryption from the Application to the Database”

5.13.1. Enabling Database SSL

The steps outlined below explain how to enable security within MySQL (If it is not already enabled by default in the release your are using). There are different approaches depending on the version/distribution of MySQL you are using. If in any doubt, you should consult the appropriate documentation pages for the MySQL release you are using.

5.13.2. Configure Tungsten<>Database Secure Communication

If you choose to enable database level SSL within your MySQL installation, there are a number of additional steps required to allow the Tungsten Components to be able to communicate to the database layer.

The steps below make the following assumptions:

- You have enabled SSL using the correct procedures for your distribution of MySQL. If not, refer to Section 5.13.1, “Enabling Database SSL”.
- You have generated, and have access to, the client level certificates and keys

1. If SSL has been enabled within the Tungsten installation, then you should either have the following parameter in your configuration, or it will be ommitted altogether since security is enabled by default:

   ```
   disable-security-controls=false
   ```

   As a result, you should have a number of files within `/opt/continuent/share`

   ```
   shell> ls -l
   total 20
   -rw-rw-r-- 1 tungsten tungsten  104 Jul 18 10:15 jmxremote.access
   -rw-rw-r-- 1 tungsten tungsten  729 Jul 18 10:15 passwords.store
   -rw-rw-r-- 1 tungsten tungsten 2268 Jul 18 10:15 tungsten_keystore.jks
   -rw-rw-r-- 1 tungsten tungsten 1079 Jul 18 10:15 tungsten_truststore.ts
   ```

2. If you do not have SSL enabled within the installation and you require this, then follow the steps in Section 5.1, “Enabling Security” first
3. Next, add the following parameters to your installation, but do not run tpm update yet:

   ```
   datasource-enable-ssl=true
   ```

4. You now need to convert the mysql client key to PKCS12 format. Adjust the path and filename in the example to suit your environment

   ```
   shell> openssl pkcs12 -export -in /home/tungsten/client-cert.pem \
   -inkey /home/tungsten/client-key.pem \
   -name mysql -out /home/tungsten/client-key.p12
   ```

   Important

   When prompted for a password, you MUST enter tungsten

> **Important**
>
> When using OpenSSL 3.0 with Java 1.8, you MUST add the `-legacy` option to the openssl command.

5. You now need to import the key, either into the existing keystore if it exists, or into a new one if SSL is not being enabled at the replicator level

   If Tungsten level SSL has been enabled

   ```
   shell> keytool -importkeystore -deststorepass tungsten \
   -destkeystore /opt/continuent/share/tungsten_keystore.jks \
   -srckeystore /home/tungsten/client-key.p12 -srcstoretype PKCS12
   ```

   If ONLY Database SSL is required

   ```
   shell> keytool -importkeystore -deststorepass tungsten \
   -destkeystore /home/tungsten/tungsten_keystore.jks \
   -srckeystore /home/tungsten/client-key.p12 -srcstoretype PKCS12
   ```

   When prompted for a password, enter tungsten

6. Next, import the client certificate into the truststore

   If Tungsten level SSL has been enabled

   ```
   shell> keytool -import -alias mysql -trustcacerts -file /home/tungsten/ca.pem \
   -keystore /opt/continuent/share/tungsten_truststore.ts
   ```

   If ONLY Database SSL is required

   ```
   shell> keytool -import -alias mysql -trustcacerts -file /home/tungsten/ca.pem \
   -keystore /home/tungsten/tungsten_truststore.ts
   ```

   When prompted for a password, enter tungsten

7. Finally, and only if Tungsten level SSL has been enabled, we need to create backups copies of the keystore and truststore as follows:

   ```
   shell> cp /opt/continuent/share/tungsten_truststore.ts /opt/continuent/share/.tungsten_truststore.ts.orig
   shell> cp /opt/continuent/share/tungsten_keystore.jks /opt/continuent/share/.tungsten_keystore.jks.orig
   ```

8. Issue tpm update to apply the configuration

The replicators will be restarted as part of the update process, and should now be using SSL to connect successfully to MySQL

5.13.3. Configuring Connector SSL

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

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

There are three different tpm properties that control SSL for the connectors when using Proxy mode, these are:

- `connector-client-ssl` [490] : This controls SSL between your applications and the connectors.
- `connector-server-ssl` [490] : This controls SSL between the connectors and MySQL.
- `connector-ssl` [491] : This is an alias that will control both of the above properties.

Additionally, `connector-ssl-capable` [484] can be used to control whether the connector advertises that it is SSL capable to clients. When SSL is enabled, this property is also enabled. With some clients, this triggers them to use SSL even if SSL has not been configured. This causes the connections to fail and not operate correctly. In those situations, setting this value to `false` would be appropriate

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

To enable SSL communication with Tungsten Connector you must create suitable certificates keys and keystores, as described in Chapter 5, *Deployment: Security*. The keystores used for Tungsten Connector can be the same, or different, to the keystores used for securing the manager and replication communication.

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

Connector SSL will be enabled by default during installation, the `connector-ssl` [491] option can be used to explicitly enable this if required `disable-security-controls=true` [490]

Before changing the property and enabling Connector SSL, a number of other steps first need to be accomplished.

- Create, activate and test SSL keys for the MySQL server. Refer to Section 5.13.1, "Enabling Database SSL" for steps on accomplishing this,
- Enable and test SSL encrypted traffic between the MySQL server and the Connector. See Section 5.13.3.1, "Enable and Test SSL encryption from the Connector to the Database"
- Enable and test SSL encrypted traffic between the Application/Client and the Connector. See Section 5.13.3.2, "Test SSL encryption from the Application to the Database"

> **Note**
>
> If you are installing a new cluster you only need to ensure database SSL has been configured. Everything else will be handled. If you are configuring connector SSL as a post-installation task, then this document explains tha various steps required.

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

1. Convert MySQL Client Cert to pkcs12 format

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

2. Create `tungsten_connector_keystore.jks`

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

3. Import the CA Cert into the KeyStore

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

4. Import the CA Cert into the TrustStore

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

5. For INI-based deployments only, copy the certs to all Connector nodes (repeat as needed so that every Connector node has the same certificates)

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

6. Set proper ownership and permissions on ALL Connector nodes

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

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

   See Section 7.6.1, "`user.map` File Format" for more information.

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

8. Update the Connector configuration to enable SSL

- Staging Method

  Update all nodes (DB & Connector) in the cluster

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

shell> tools/tpm update
```

- INI Method

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

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

shell> tpm update
```

9. Test SSL connectivity through the connector

   a. Connect as the default application user

```
shell> tpm connector
```

   b. Check the connection status

> **Note**
>
> Expecting "SSL.IN=false SSL.OUT=true"
>
> SSL.IN is false because the the tpm connector command calls the mysql client in non-SSL mode.
>
> SSL.OUT is true because the connection to the database is encrypted, even if the connection from the mysql client is not.
>
> This can be verified with the "sudo tcpdump -X port 13306" command. Without the encryption, queries and responses are sent in plaintext and are visible in the output of tcpdump. When encryption is enabled, the queries and results are no longer visible.

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

   c. Check the SSL status

> **Note**
>
> Expecting "SSL: Not in use"
>
> SSL is not in use because the the tpm connector command calls the mysql client in non-SSL mode.
>
> The connection to the database is encrypted, even if the connection from the mysql client is not.
>
> This can be verified with the "sudo tcpdump -X port 13306" command. Without the encryption, queries and responses are sent in plaintext and are visible in the output of tcpdump. When encryption is enabled, the queries and results are no longer visible.

```
mysql> status
```

```
--------------
mysql  Ver 14.14 Distrib 5.5.42-37.1, for Linux (x86_64) using readline 5.1

Connection id:      70
Current database:
Current user:       app_user@app1
SSL:            Not in use
Current pager:      stdout
Using outfile:      ''
Using delimiter:    ;
Server version:      5.5.42-37.1-log-tungsten Percona Server (GPL), Release 37.1, Revision 39acee0
Protocol version:   10
Connection:      app1 via TCP/IP
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    latin1
Conn.  characterset:    latin1
TCP port:       3306
Uptime:          2 hours 27 min 53 sec

Threads: 4  Questions: 41474  Slow queries: 0  Opens: 47
Flush tables: 2  Open tables: 10  Queries per second avg: 4.674
--------------
```

Important

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

5.13.3.2. Test SSL encryption from the Application to the Database

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

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

2. Check the connection status

Note

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

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

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

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

3. Check the SSL status

Note

Expecting "Cipher in use is *xxx-xxx-xxxxxx-xxx*"

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

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

```
mysql> status
--------------
mysql  Ver 14.14 Distrib 5.5.42-37.1, for Linux (x86_64) using readline 5.1

Connection id:      68
Current database:
Current user:       ssl_user@app1
SSL:            Cipher in use is DHE-RSA-AES256-SHA
Current pager:      stdout
Using outfile:      ''
Using delimiter:    ;
Server version:      5.5.42-37.1-log-tungsten Percona Server (GPL), Release 37.1, Revision 39acee0
Protocol version:   10
Connection:      app1 via TCP/IP
```

```
Server characterset:   latin1
Db     characterset:   latin1
Client characterset:   latin1
Conn.  characterset:   latin1
TCP port:      3306
Uptime:         2 hours 33 min 32 sec

Threads: 4  Questions: 43065  Slow queries: 0  Opens: 47
Flush tables: 2  Open tables: 10  Queries per second avg: 4.674
--------------
```

Important

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

Chapter 6. Operations Guide

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

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

If the script was not added to the login script automatically, or needs to be added to the current session, the script is located within the `share` directory of the installation directory. For example, `/opt/continuent/share/env.sh`. To load into the current session use source. See Section 6.2, "Establishing the Shell Environment" for more information.

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

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

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

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

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

6.1. The Home Directory

After installing Tungsten Clustering the home directory will be filled with a set of new directories. The home directory is specified by `home-directory` [494] or `install-directory` [494]. If you have multiple installations on a single server; each directory will include the same entries.

- tungsten - A symlink to the most recent version of the software. The symlink points into the `releases` directory. You should always use the symlink to ensure the most recent configuration and software is used.
- releases - Storage for the current and previous versions of the software. During an upgrade the new software will be copied into this directory and the `tungsten` symlink will be updated. See Section D.1.2, "The `releases` Directory" for more information.
- service_logs - Includes symlinks to the primary log for the replicator, manager and connector. This directory also includes logs for other tools distributed for Tungsten Cluster.
- backups - Default storage for backup files created through trepctl or cctrl if no overriding property has been specified. See Section D.1.1, "The `backups` Directory" for more information.
- thl - Default storage for THL files created by the replicator. Each replication service gets a dedicated sub-directory for storing THL files. See Section D.1.5, "The `thl` Directory" for more information.
- relay - Temporary storage for downloaded MySQL binary logs before they are converted into THL files.
- share - Storage for files that must persist between different software versions. The `env.sh` script will setup your shell environment to allow easy access to Tungsten Cluster tools. This is also the default location for all necessary SSL certificates and keys.

6.2. Establishing the Shell Environment

The tools required to operate Tungsten Clustering are located in many directories around the home directory. The best way to access them is by setting up your shell environment.

The `env.sh` file will automatically be included if you specify the `profile-script` [507] 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 [493] and profile-script [507] 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 [493] the trepctl command would be accessed as mm_trepctl.

6.3. Checking Dataservice Status

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

```
shell> cctrl
Tungsten Clustering 7.0.3 build 141
alpha: session established, encryption=false, authentication=false
jgroups: unencrypted, database: unencrypted
[LOGICAL] /alpha > ls
COORDINATOR[db1:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@db1[2194](ONLINE, created=0, active=0)                            |
|connector@db2[2187](ONLINE, created=0, active=0)                            |
|connector@db3[2185](ONLINE, created=0, active=0)                            |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|db1(master:ONLINE, progress=4, THL latency=0.415)                           |
|STATUS [OK] [2025/01/23 11:35:54 AM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db2(slave:ONLINE, progress=4, latency=0.457)                                |
|STATUS [OK] [2025/01/23 11:35:54 AM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db3(slave:ONLINE, progress=4, latency=0.459)                                |
|STATUS [OK] [2025/01/23 11:35:54 AM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
```

The output consists of the following major sections:

- COORDINATOR

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

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

  - HOSTNAME

    The hostname of the current coordinator.

  - POLICY

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

    For more information on policy modes, see Section 6.4, “Policy Modes” .

  - STATUS

The current status of the coordinator host.

- **ROUTERS**

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

- **DATASOURCES**

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

```
+----------------------------------------------------------------------------+
|db1(master:ONLINE, progress=4, THL latency=0.415)                           |
|STATUS [OK] [2025/01/23 11:35:54 AM UTC]                                    |
+----------------------------------------------------------------------------+
```

  The first line describes the host and status information:

  - Hostname of the datasource (db1)
  - Current role within the dataservice and status of the datasource. For more information on roles, see Understanding Datasource Roles. For information on datasource states, see Section 6.3.4, “Understanding Datasource States” .
  - The progress indicates the current sequence number from the THL for the datasource.
  - The THL latency shows the current latency of the datasource. For a Primary datasource this is the latency between the data being written to the MySQL binary log and being processed in the THL. For a Replica, it shows the latency between the original commit (from the Primary) and the application on the Replica. For more extensive detail on interpreting latency, see Section 6.3.1, “Latency or Relative Latency Display”

  The second line provides a more detailed current status, and the time since the status was last changed. In the event of a change of status, for example to the SHUNNED or OFFLINE state, the time will indicate how long the node has been in that status.

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

  - `|  MANAGER(state=ONLINE)                                                  |`

    The Manager service, and the current status of the manager. If a configured datasource is down, has recently been restarted, or the manager has been stopped, the status may be offline.

  - `|  REPLICATOR(role=slave, master=db1, state=ONLINE)                       |`

    The Tungsten Replicator service, which replicates data between hosts. The status shows the current role (Replica), the Primary host, and the current status of the replicator.

  - `|  DATASERVER(state=ONLINE)                                              |`

    The status of the dataserver service, which indicates the status of the underlying database service.

  - `|  CONNECTIONS(created=0, active=0)                                      |`

    The Tungsten Connector service, showing the number of connections have been created on this service, and the number that are currently active.

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

6.3.1. Latency or Relative Latency Display

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

- Absolute latency — (default) is the difference between when a transaction was applied to a Replica and when the transaction was originally applied to the Primary.
- Relative latency — is the difference between now and when the last transaction was written to the Replica.

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

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

To enable relative latency, the cluster must have been deployed, or updated, using the `use-relative-latency=true` [485] option to tpm. Once enabled, the following operational activities change:

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

```
mysql> show slave status\G
*************************** 1. row ***************************
...
Seconds_Behind_Master: 0
...
1 row in set (0.01 sec)
mysql> show slave status\G
*************************** 1. row ***************************
...
Seconds_Behind_Master: 7
...
1 row in set (0.01 sec)
mysql> show slave status\G
*************************** 1. row ***************************
...
Seconds_Behind_Master: 38
...
1 row in set (0.01 sec)
```

- cctrl will output an additional field, `relative`, showing the relative latency value against the standard latency value. This can be seen in the example below:

```
[LOGICAL] /alpha > ls
COORDINATOR[db3:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@db1[61355](ONLINE, created=1, active=0)                           |
|connector@db2[61007](ONLINE, created=1, active=0)                           |
|connector@db3[63912](ONLINE, created=1, active=0)                           |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|db1(master:ONLINE, progress=0, THL latency=0.669, relative=6.124)           |
|STATUS [OK] [2025/01/24 11:49:49 AM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+

+----------------------------------------------------------------------------+
|db2(slave:ONLINE, progress=0, latency=1.002, relative=6.158)                |
|STATUS [OK] [2025/01/24 11:49:49 AM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+

+----------------------------------------------------------------------------+
|db3(slave:ONLINE, progress=0, latency=0.767, relative=6.184)                |
|STATUS [OK] [2025/01/24 11:49:49 AM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=3, active=0)                                          |
+----------------------------------------------------------------------------+
```

- The Tungsten Connector will use the value when the *maxAppliedLatency* option is used in the connection string to determine whether to route a connection to a Primary or a Replica.

  For example, when running a scrip that sends a heartbeat, and then connects through a connector, the connection will be routed first to the Replica, and then to the Primary:

```
echo "cluster heartbeat" | cctrl
sleep 1
mysql -utungsten -psecret --port=3306 --host=db2 \
mysql@maxAppliedLatency=20?qos=RO_RELAXED -e"select 1;tungsten connection status;"

sleep 21

mysql -utungsten -psecret --port=3306 --host=db2 \
mysql@maxAppliedLatency=20?qos=RO_RELAXED -e"select 1;tungsten connection status;"
```

  The output of the execution of the script shows the Replica and then Primary connections:

```
[LOGICAL] /alpha > cluster heartbeat
HEARTBEAT 'DEFAULT' INSERTED
[LOGICAL] /alpha >
Exiting...
+---+
| 1 |
+---+
| 1 |
+---+
+-------------------------------------------------------------------------------------------------------------+
| Message                                                                                                     |
+-------------------------------------------------------------------------------------------------------------+
| db3@alpha(slave:ONLINE)                                                                                     |
| STATUS [OK] [Fri Jan 24 13:43:04 UTC 2025], QOS=RO_RELAXED SSL.IN=false SSL.OUT=false                       |
+-------------------------------------------------------------------------------------------------------------+
+---+
| 1 |
+---+
| 1 |
+---+
+-------------------------------------------------------------------------------------------------------------+
| Message                                                                                                     |
+-------------------------------------------------------------------------------------------------------------+
| db1@alpha(master:ONLINE)                                                                                    |
| STATUS [OK] [Fri Jan 24 13:43:04 UTC 2025], QOS=RO_RELAXED SSL.IN=false SSL.OUT=false                       |
+-------------------------------------------------------------------------------------------------------------+
```

6.3.2. Getting Detailed Information

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

```
[LOGICAL] /alpha > ls -l db2
COORDINATOR[db2:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@db1[74034](ONLINE, created=4, active=0)                           |
|connector@db2[75837](ONLINE, created=8, active=0)                           |
|connector@db3[76888](ONLINE, created=4, active=0)                           |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|db2(slave:ONLINE, progress=3, latency=267.323)                              |
|STATUS [OK] [2025/01/24 01:26:24 PM UTC]                                    |
+----------------------------------------------------------------------------+
|         activeConnectionCount:    0                                        |
|                  alertMessage:                                             |
|                   alertStatus:    OK                                       |
|                     alertTime:    1737725184632                            |
|                appliedLatency:    267.323                                  |
|callableStatementsCreatedCount:    0                                        |
|       connectionsCreatedCount:    0                                        |
|                dataServerHost:    db2                                      |
|               dataServiceName:    alpha                                    |
|             datasourceGroupId:    0                                        |
|                        driver:    org.drizzle.jdbc.DrizzleDriver           |
|                     highWater:    0(mysql-bin.000019:0000000000000625;10516)|
|                          host:    db2                                      |
|                     isArchive:    false                                    |
|                   isAvailable:    true                                     |
```

```
|                  isComposite:    false                                       |
|                isSubService:    false                                       |
|                   lastError:                                                |
|              lastShunReason:                                                |
|             masterConnectUri:    thl://db1:2112/                           |
|                         name:    db2                                       |
|preparedStatementsCreatedCount:   0                                         |
|              relativeLatency:    276.085                                   |
|                         role:    slave                                     |
|                     sequence:    Sequence(0:0)                             |
|                        state:    ONLINE                                    |
|       statementsCreatedCount:    0                                         |
|url:                                                                        |
|jdbc:mysql:thin://db2:13306/${DBNAME}?jdbcCompliantTruncation=false&zeroDateT|
|imeBehavior=convertToNull&tinyInt1isBit=false&allowMultiQueries=true&yearIsDa|
|teType=false                                                                |
|                       useSSL:    false                                     |
|                       vendor:    mysql                                     |
|                   vipAddress:                                              |
|                  vipInterface:                                             |
|                    vipIsBound:    false                                    |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db2:REPLICATOR(role=slave, state=ONLINE)                                    |
+----------------------------------------------------------------------------+
|           appliedLastEventId:    mysql-bin.000019:0000000000002023;11970   |
|        appliedLastGoodLatency:    267.323                                  |
|          appliedLastGoodSeqno:    3                                        |
|              appliedLastSeqno:    3                                        |
|                appliedLatency:    267.323                                  |
|           autoRecoveryEnabled:    false                                    |
|             autoRecoveryTotal:    0                                        |
|                      channels:    1                                        |
|                   clusterName:    alpha                                    |
|               currentEventId:    NONE                                     |
|             currentTimeMillis:    1737726516085                            |
|                dataServerHost:    db2                                      |
|                    extensions:                                             |
|                          host:    db2                                      |
|             latestEpochNumber:    2                                        |
|              masterConnectUri:    thl://db1:2112/                          |
|               masterListenUri:    thl://db2:2112/                          |
|            maximumStoredSeqNo:    3                                        |
|            minimumStoredSeqNo:    0                                        |
|            notificationSource:    TungstenReplicatorOSSChecker:connected   |
|               offlineRequests:    NONE                                     |
|                  pendingError:    NONE                                     |
|             pendingErrorCode:    NONE                                      |
|           pendingErrorEventId:    NONE                                     |
|             pendingErrorSeqno:    -1                                       |
|       pendingExceptionMessage:    NONE                                     |
|                pipelineSource:    thl://db1:2112/                          |
|              relativeLatency:    276.085                                  |
|           resourceJdbcDriver:    org.drizzle.jdbc.DrizzleDriver           |
|resourceJdbcUrl:                                                            |
|jdbc:mysql:thin://db2:13306/${DBNAME}?jdbcCompliantTruncation=false&zeroDateT|
|imeBehavior=convertToNull&tinyInt1isBit=false&allowMultiQueries=true&yearIsDa|
|teType=false                                                                |
|                resourceVendor:    mysql                                    |
|                       rmiPort:    10000                                    |
|                          role:    slave                                    |
|                    seqnoType:    java.lang.Long                           |
|                  serviceName:    alpha                                    |
|                  serviceType:    local                                    |
|            simpleServiceName:    alpha                                    |
|                     siteName:    default                                  |
|                     sourceId:    db2                                      |
|                        state:    ONLINE                                   |
|           timeInStateSeconds:    8.814                                    |
|                     timezone:    GMT                                      |
|              transitioningTo:                                              |
|                 uptimeSeconds:    336.638                                  |
|              useSSLConnection:    false                                    |
|                      version:    Tungsten Clustering 7.0.3 build 141      |
|                   vipAddress:                                              |
|                  vipInterface:                                             |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db2:DATASERVER(state=ONLINE)                                                |
+----------------------------------------------------------------------------+
|                  Binlog_Do_DB:                                             |
|              Binlog_Ignore_DB:                                             |
|             Executed_Gtid_Set:                                             |
|                          File:    mysql-bin.000011                         |
```

```
|                     Position:    157                                          |
|               applied_latency:    267                                          |
|                  epoch_number:    2                                            |
|                       eventid:    mysql-bin.000019:0000000000002023;11970      |
|             extract_timestamp:    1737726240000                                |
|                        fragno:    0                                            |
|                    is_read_only:    1                                          |
|                       last_frag:    1                                          |
|              relative_latency:    275                                          |
|                         seqno:    3                                            |
|                      shard_id:    tungsten_alpha                               |
|                     source_id:    db1                                          |
|                       task_id:    0                                            |
|               update_timestamp:    1737726507000                                |
+---------------------------------------------------------------------------------+
```

The information output is very detailed and provides a summary of all the configuration and status information for the given host. The connector information shows connectors made to each configured dataserver by each connector service. The datasource section shows detailed information on the dataserver and replicator services. The output from the replicator service is equivalent to that output by trepctl .

6.3.3. Understanding Datasource Roles

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

Role	Supplies Replication Data	Receives Replication Data	Load Balancing	Failover
master	Yes	No	Yes	Yes
slave	No	Yes	Yes	Yes
relay	Yes	Yes	Yes	Yes
standby	No	Yes	No	Yes
archive	No	Yes	Yes	No

More detailed information for each role:

- *master* (Primary)

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

- *slave* (Replica)

  A *slave* datasource is receiving data from a *master* and having that replicated data applied by Tungsten Cluster. Replicas are used for read-only operations by applications.

- *relay*

  A *relay* datasource receives replication data, from upstream primary nodes in remote clusters, and serves the downstream replicas within the same cluster.

  This role is specific to the passive cluster within an Composite Active/Passive topology, or within the cross-site sub-services within Composite Active/Active and Composite Dynamic Active/Active topologies.

  In the event of a failover between the active and passive cluster within a Composite Active/Passive configuration, the *relay* node will be promoted to the *master* role. During a switch, the old *master* will become the *relay*.

- *standby*

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

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

- *archive*

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

6.3.4. Understanding Datasource States

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

6.3.4.1. ONLINE State

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

6.3.4.2. OFFLINE State

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

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

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

6.3.4.3. FAILED State

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

```
+----------------------------------------------------------------------------+
|db3(slave:FAILED(DATASERVER 'db3@alpha' STOPPED), progress=3, latency=0.194)|
|STATUS [CRITICAL] [2025/01/24 02:59:42 PM UTC]                              |
|REASON[DATASERVER 'db3@alpha' STOPPED]                                      |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
|  DATASERVER(state=STOPPED)                                                 |
|  CONNECTIONS(created=8, active=0)                                          |
+----------------------------------------------------------------------------+
```

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

6.3.4.4. SHUNNED State

A SHUNNED datasource implies that the datasource is OFFLINE. Unlike the OFFLINE state, a SHUNNED datasource is not automatically recovered.

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

Datasources can be manually or automatically shunned. The current reason for the SHUNNED state is indicated in the status output. For example, in the sample below, the node db3 was manually shunned for maintenance reasons:

```
+----------------------------------------------------------------------------+
|db3(slave:SHUNNED(MANUALLY-SHUNNED), progress=3, latency=0.000)             |
|STATUS [SHUNNED] [2025/01/24 03:00:42 PM UTC]                               |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=8, active=0)                                          |
+----------------------------------------------------------------------------+
```

6.3.4.4.1. Various SHUNNED States

A SHUNNED node can have a number of different sub-states depending on certain actions or events that have happened within the cluster. These are as folllows:

- SHUNNED(DRAIN-CONNECTIONS)
- SHUNNED(FAILSAFE_SHUN)
- SHUNNED(MANUALLY-SHUNNED)
- SHUNNED(CONFLICTS-WITH-COMPOSITE-MASTER)
- SHUNNED(FAILSAFE AFTER Shunned by fail-safe procedure)
- SHUNNED(SUBSERVICE-SWITCH-FAILED)
- SHUNNED(FAILED-OVER-TO-node)
- SHUNNED(SET-RELAY)

- SHUNNED(FAILOVER-ABORTED AFTER UNABLE TO COMPLETE FAILOVER)
- SHUNNED(CANNOT-SYNC-WITH-HOME-SITE)

Below are various examples and possible troubleshooting steps and soultions, where applicable.

> **Warning**
>
> Please THINK before you issue ANY commands. These are examples ONLY, and are not to be followed blindly because every situation is different

6.3.4.4.1.1. SHUNNED(DRAIN-CONNECTIONS)

The DRAIN-CONNECTIONS state means that the datasource [NODE|CLUSTER] drain [timeout] command has been successfully completed and the node or cluster is now SHUNNED as requested.

The datasource drain command will prevent new connections to the specified data source, while ongoing connections remain untouched. If a timeout (in seconds) is given, ongoing connections will be severed after the timeout expires. This command returns immediately, no matter whether a timeout is given or not. Under the hood, this command will put the data source into SHUNNED state, with lastShunReason set to DRAIN-CONNECTIONS. This feature is available as of version 7.0.2

```
[LOGICAL] / > use gloabl
[LOGICAL] /global > datasource beta drain
[LOGICAL] /global > ls
DATASOURCES:
+---------------------------------------------------------------------------+
|alpha(composite master:ONLINE, global progress=215, max latency=0.937)    |
|STATUS [OK] [2025/01/24 03:08:54 PM UTC]                                   |
+---------------------------------------------------------------------------+
|  alpha(master:ONLINE, progress=8, max latency=0.937)                      |
|  alpha_from_beta(relay:ONLINE, progress=207, max latency=0.633)           |
+---------------------------------------------------------------------------+
+---------------------------------------------------------------------------+
|beta(composite master:SHUNNED(DRAIN-CONNECTIONS), global progress=8, max   |
|latency=0.955)                                                             |
|STATUS [SHUNNED] [2025/01/24 03:11:50 PM UTC]                              |
+---------------------------------------------------------------------------+
|  beta(master:SHUNNED, progress=207, max latency=0.611)                    |
|  beta_from_alpha(relay:ONLINE, progress=8, max latency=0.955)             |
+---------------------------------------------------------------------------+

[LOGICAL] /global > recover
```

6.3.4.4.1.2. SHUNNED(FAILSAFE_SHUN)

The FAILSAFE_SHUN state means that there was a complete network partition so that none of the nodes were able to communicate with each other. The database writes are blocked to prevent a split-brain from happening.

```
+---------------------------------------------------------------------------+
|db1(master:SHUNNED(FAILSAFE_SHUN), progress=56747909871, THL               |
|latency=12.157)                                                            |
|STATUS [OK] [2025/01/24 02:08:54 PM UTC]                                   |
+---------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                     |
| REPLICATOR(role=master, state=ONLINE)                                     |
| DATASERVER(state=ONLINE)                                                  |
| CONNECTIONS(created=374639937, active=0)                                  |
+---------------------------------------------------------------------------+
+---------------------------------------------------------------------------+
|db2(slave:SHUNNED(FAILSAFE_SHUN), progress=-1, latency=-1.000)             |
|STATUS [SHUNNED] [2025/01/24 03:11:50 PM UTC]                              |
+---------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                     |
| REPLICATOR(role=slave, master=db1, state=OFFLINE)                         |
| DATASERVER(state=STOPPED)                                                 |
| CONNECTIONS(created=70697946, active=0)                                   |
+---------------------------------------------------------------------------+
+---------------------------------------------------------------------------+
|db3(slave:SHUNNED(FAILSAFE_SHUN), progress=56747909871, latency=12.267)    |
|STATUS [SHUNNED] [2025/01/24 02:32:50 PM UTC]                              |
+---------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                     |
| REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
| DATASERVER(state=ONLINE)                                                  |
| CONNECTIONS(created=168416988, active=0)                                  |
+---------------------------------------------------------------------------+

cctrl> set force true
cctrl> datasource db1 welcome
cctrl> datasource db1 online (if needed)
```

```
cctrl> recover
```

6.3.4.4.1.3. SHUNNED(MANUALLY-SHUNNED)

The MANUALLY-SHUNNED state means that an administrator has issued the datasource {NODE|CLUSTER} shun command using cctrl or the REST API, resulting in the specified node or cluster being SHUNNED.

> **Warning**
>
> Unless unavoidable, and directed by Continuent Support, you should never manually shun the primary node

```
+----------------------------------------------------------------------------+
|db2(slave:SHUNNED(MANUALLY-SHUNNED), progress=8, latency=0.937)             |
|STATUS [SHUNNED] [2025/01/24 03:16:56 PM UTC]                               |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=9, active=0)                                          |
+----------------------------------------------------------------------------+

cctrl> datasource db2 welcome
cctrl> datasource db2 online (if needed)
cctrl> recover
```

6.3.4.4.1.4. SHUNNED(CONFLICTS-WITH-COMPOSITE-MASTER)

The CONFLICTS-WITH-COMPOSITE-MASTER state means that we already have an active primary in the cluster and we can't bring this primary online because of this.

```
+----------------------------------------------------------------------------+
|db1(master:SHUNNED(CONFLICTS-WITH-COMPOSITE-MASTER),                        |
|progress=25475128064, THL latency=0.010)                                    |
|STATUS [SHUNNED] [2025/01/24 03:16:56 PM UTC]                               |
+----------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                      |
| REPLICATOR(role=master, state=ONLINE)                                      |
| DATASERVER(state=ONLINE)                                                   |
| CONNECTIONS(created=2568, active=0)                                        |
+----------------------------------------------------------------------------+
```

6.3.4.4.1.5. SHUNNED(FAILSAFE AFTER Shunned by fail-safe procedure)

The FAILSAFE AFTER Shunned by fail-safe procedure state means that the Manager voting Quorum encountered an unrecoverable problem and shut down database writes to prevent a Split-brain situation.

```
+----------------------------------------------------------------------------+
|db1(master:SHUNNED(FAILSAFE AFTER Shunned by fail-safe                      |
|procedure), progress=96723577, THL latency=0.779)                           |
|STATUS [OK] [2025/01/24 03:08:57 PM UTC]                                    |
+----------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                      |
| REPLICATOR(role=master, state=ONLINE)                                      |
| DATASERVER(state=ONLINE)                                                   |
| CONNECTIONS(created=135, active=0)                                         |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db2(slave:SHUNNED(FAILSAFE AFTER Shunned by fail-safe                       |
|procedure), progress=96723575, latency=0.788)                               |
|STATUS [SHUNNED] [2025/01/24 03:16:56 PM UTC]                               |
+----------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                      |
| REPLICATOR(role=slave, master=db1, state=ONLINE)                           |
| DATASERVER(state=ONLINE)                                                   |
| CONNECTIONS(created=28, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db5(slave:SHUNNED:ARCHIVE (FAILSAFE AFTER Shunned by                        |
|fail-safe procedure), progress=96723581, latency=0.905)                     |
|STATUS [OK] [2025/01/24 03:08:57 PM UTC]                                    |
+----------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                      |
| REPLICATOR(role=slave, master=db1, state=ONLINE)                           |
| DATASERVER(state=ONLINE)                                                   |
| CONNECTIONS(created=23, active=0)                                          |
+----------------------------------------------------------------------------+

cctrl> set force true
cctrl> datasource db1 welcome
cctrl> datasource db1 online (if needed)
cctrl> recover
```

6.3.4.4.1.6. SHUNNED(SUBSERVICE-SWITCH-FAILED)

The SUBSERVICE-SWITCH-FAILED state means that the cluster tried to switch the Primary role to another node in response to an admin request, but was unable to do so due to a failure at the sub-service level in a Composite Active/Active cluster.

```
+----------------------------------------------------------------------------+
|db1(relay:SHUNNED(SUBSERVICE-SWITCH-FAILED), progress=6668586,              |
|latency=1.197)                                                              |
|STATUS [OK] [2025/01/24 03:08:57 PM UTC]                                    |
+----------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                      |
| REPLICATOR(role=relay, master=db4, state=ONLINE)                           |
| DATASERVER(state=ONLINE)                                                   |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db2(slave:SHUNNED(SUBSERVICE-SWITCH-FAILED), progress=6668586,              |
|latency=1.239)                                                              |
|STATUS [SHUNNED] [2025/01/24 03:16:56 PM UTC]                               ||
+----------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                      |
| REPLICATOR(role=slave, master=db1, state=ONLINE)                           |
| DATASERVER(state=ONLINE)                                                   |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db3(slave:SHUNNED(SUBSERVICE-SWITCH-FAILED), progress=6668591,              |
|latency=0.501)                                                              |
|STATUS [OK] [2025/01/24 03:08:57 PM UTC]                                    |
+----------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                      |
| REPLICATOR(role=slave, master=pip-db1, state=ONLINE)                       |
| DATASERVER(state=ONLINE)                                                   |
+----------------------------------------------------------------------------+

cctrl> use {SUBSERVICE-NAME-HERE}
cctrl> set force true
cctrl> datasource db1 welcome
cctrl> datasource db1 online (if needed)
cctrl> recover
```

6.3.4.4.1.7. SHUNNED(FAILED-OVER-TO-node)

The FAILED-OVER-TO-node state means that the cluster automatically and successfully invoked a failover from one node to another. The fact that there appear to be two masters is completely normal after a failover, and indicates the cluster should be manually recovered once the node which failed is fixed.

> **Note**
>
> After recovery, the failed master will be recovered as a slave. The custer does not automatically switch back to the original master. Should this be required, you should issue the switch to *nodename* command.

```
+----------------------------------------------------------------------------+
|db1(master:SHUNNED(FAILED-OVER-TO-db3), progress=8, THL latency=0.825)      |
|STATUS [SHUNNED] [2025/01/24 03:22:57 PM UTC]                               |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=DEGRADED)                                   |
|  DATASERVER(state=STOPPED)                                                 |
|  CONNECTIONS(created=1, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db2(slave:ONLINE, progress=8, latency=0.937)                                |
|STATUS [OK] [2025/01/24 03:23:17 PM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=9, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db3(master:ONLINE, progress=10, THL latency=0.165)                          |
|STATUS [OK] [2025/01/24 03:23:02 PM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=1, active=0)                                          |
+----------------------------------------------------------------------------+

cctrl> recover
cctrl> switch to db (if required)
```

6.3.4.4.1.8. SHUNNED(SET-RELAY)

The SET-RELAY state means that the cluster was in the middle of a switch which failed to complete for either a Composite Active/Passive Passive cluster, or in a Composite Active/Active sub-service.

```
+---------------------------------------------------------------------------+
|db1(relay:SHUNNED(SET-RELAY), progress=-1, latency=-1.000)                 |
|STATUS [SHUNNED] [2025/01/24 03:22:57 PM UTC]                              |
+---------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                     |
| REPLICATOR(role=relay, master=db4, state=SUSPECT)                         |
| DATASERVER(state=ONLINE)                                                  |
+---------------------------------------------------------------------------+
+---------------------------------------------------------------------------+
|db2(slave:SHUNNED(SUBSERVICE-SWITCH-FAILED), progress=14932,               |
|latency=0.000)                                                             |
|STATUS [SHUNNED] [2025/01/24 03:22:57 PM UTC]                              |
+---------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                     |
| REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
| DATASERVER(state=ONLINE)                                                  |
+---------------------------------------------------------------------------+
+---------------------------------------------------------------------------+
|db3(slave:SHUNNED(SUBSERVICE-SWITCH-FAILED), progress=14932,               |
|latency=0.000)                                                             |
|STATUS [SHUNNED] [2025/01/24 03:22:57 PM UTC]                              |
+---------------------------------------------------------------------------+
| MANAGER(state=ONLINE) |
| REPLICATOR(role=slave, master=db1, state=ONLINE) |
| DATASERVER(state=ONLINE) |
+---------------------------------------------------------------------------+

cctrl> use {PASSIVE-SERVICE-NAME-HERE}
cctrl> set force true
cctrl> datasource db1 welcome
cctrl> datasource db1 online (if needed)
cctrl> recover
```

6.3.4.4.1.9. SHUNNED(FAILOVER-ABORTED AFTER UNABLE TO COMPLETE FAILOVER)

The FAILOVER-ABORTED AFTER UNABLE TO COMPLETE FAILOVER state means that the cluster tried to automatically fail over the Primary role to another node but was unable to do so.

```
+---------------------------------------------------------------------------+
|db1(master:SHUNNED(FAILOVER-ABORTED AFTER UNABLE TO COMPLETE FAILOVER      |
|       FOR DATASOURCE 'db1'. CHECK COORDINATOR MANAGER LOG),                |
|       progress=21179013, THL latency=4.580)                               |
|STATUS [SHUNNED] [2025/01/24 03:22:57 PM UTC]                              |
+---------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                    |
|  REPLICATOR(role=master, state=ONLINE)                                    |
|  DATASERVER(state=ONLINE)                                                 |
|  CONNECTIONS(created=294474815, active=0)                                 |
+---------------------------------------------------------------------------+
+---------------------------------------------------------------------------+
|db2(slave:ONLINE, progress=21179013, latency=67.535)                       |
|STATUS [OK] [2025/01/24 03:23:02 PM UTC]                                   |
+---------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                    |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                         |
|  DATASERVER(state=ONLINE)                                                 |
|  CONNECTIONS(created=22139851, active=1)                                  |
+---------------------------------------------------------------------------+
+---------------------------------------------------------------------------+
|db3(slave:ONLINE, progress=21179013, latency=69.099)                       |
|STATUS [OK] [2025/01/24 03:23:02 PM UTC]                                   |
+---------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                    |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                         |
|  DATASERVER(state=ONLINE)                                                 |
|  CONNECTIONS(created=66651718, active=7)                                  |
+---------------------------------------------------------------------------+
```

6.3.4.4.1.10. SHUNNED(CANNOT-SYNC-WITH-HOME-SITE)

The CANNOT-SYNC-WITH-HOME-SITE state is a composite-level state which means that the sites were unable to see each other at some point in time. This scenario may need a manual recovery at the composite level for the cluster to heal.

```
From alpha side:
beta(composite master:SHUNNED(CANNOT-SYNC-WITH-HOME-SITE)
```

```
From beta side:
alpha(composite master:SHUNNED(CANNOT-SYNC-WITH-HOME-SITE)

cctrl compositeSvc> recover
```

6.3.5. Understanding Replicator Roles

Replicators can have one of the following roles, *master*, *slave*, *thl-server*, *thl-client* or *thl-applier*.

Replicators within Composite Active/Passive Composite Active/Active and Composite Dynamic Active/Active clusters can also have the *relay* role.

- *master*

  A replicator in a *master* role extracts data from a source database (for example, by reading the binary log from a MySQL server), and generates THL. As a *master* the replicator also provides the THL to other replicators over the network connection.

- *slave*

  A *slave* replicator pulls THL data from a *master* and then applies that data to a target database.

- *relay*

  A *relay* replicator is associated with the *relay* node within Composite Active/Passive, Composite Active/Active and Composite Dynamic Active/Active clusters only.

  It receives and applies locally, THL from the remote active cluster and is then made available for downstream replica nodes within the same custer.

- *thl-server*

  A *thl-server* replicator is a special role that Extractor replicators can be changed to temporarily when a Primary is taken offline. This will allow downstream Applier replicators to download and apply any THL that hasn't yet been processed by the Applier.

  To enable this role, first you must place the cluster into *MAINTENANCE* mode and then issue the following statements

  ```
  shell> trepctl offline
  shell> trepctl setrole -role thl-server
  shell> trepctl online
  ```

  To revert back to the original Extractor role, issue the following

  ```
  shell> trepctl offline
  shell> trepctl setrole -role master
  shell> trepctl online
  ```

- *thl-client*

  A *thl-client* replicator is a special role that Applier replicators can be changed to. This will allow the Applier replicator to download any THL available from the upstream Extractor, but does NOT apply the THL to the target database.

  To enable this role, first you must place the cluster into *MAINTENANCE* mode and then issue the following statements

  ```
  shell> trepctl offline
  shell> trepctl setrole -role thl-client
  shell> trepctl online
  ```

  To revert back to the original Applier role, issue the following

  ```
  shell> trepctl offline
  shell> trepctl setrole -role slave
  shell> trepctl online
  ```

- *thl-applier*

  A *thl-applier* replicator is a special role that applier replicators can be changed to temporarily when a Primary is taken offline. This will allow downstream applier replicators to apply any locally available THL that hasn't yet been processed by the applier.

  To enable this role, first you must place the cluster into *MAINTENANCE* mode and then issue the following statements

  ```
  shell> trepctl offline
  shell> trepctl setrole -role thl-applier
  shell> trepctl online
  ```

  To revert back to the original role, issue the following

```
shell> trepctl offline
shell> trepctl setrole -role slave
shell> trepctl online
```

6.3.6. Checking Replication Status

To check the replication status you can use the trepctl command. This accepts a number of command-specific verbs that provide status and control information for your configured cluster. The basic format of the command is:

```
shell> trepctl [-host hostname] command
```

The `-host` option is not required, and enables you to check the status of a different host than the current node.

To get the basic information about the currently configured services on a node and current status, use the services verb command:

```
shell> trepctl services
Processing services command...
NAME              VALUE
----              -----
appliedLastSeqno: 211
appliedLatency  : 17.66
role            : slave
serviceName     : firstrep
serviceType     : local
started         : true
state           : ONLINE
Finished services command...
```

In the above example, the output shows the last sequence number and latency of the host, in this case an Applier, compared to the Extractor from which it is processing information. In this example, the last sequence number and the latency between that sequence being processed on the Extractor and applied to the Target is 17.66 seconds. You can compare this information to that provided by the Extractor, either by logging into the Extractor and running the same command, or by using the host command-line option:

```
shell> trepctl -host host1 services
Processing services command...
NAME              VALUE
----              -----
appliedLastSeqno: 365
appliedLatency  : 0.614
role            : master
serviceName     : firstrep
serviceType     : local
started         : true
state           : ONLINE
Finished services command...
```

By comparing the *appliedLastSeqno* for the Extractor against the value on the Applier, it is possible to determine that the Applier and the Extractor are not yet synchronized.

For a more detailed output of the current status, use the status command, which provides much more detailed output of the current replication status:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000064:0000000002757461;0
appliedLastSeqno       : 212
appliedLatency         : 263.43
channels               : 1
clusterName            : default
currentEventId         : NONE
currentTimeMillis      : 1365082088916
dataServerHost         : host2
extensions             :
latestEpochNumber      : 0
masterConnectUri       : thl://host1:2112/
masterListenUri        : thl://host2:2112/
maximumStoredSeqNo     : 724
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : thl://host1:2112/
relativeLatency        : 655.915
resourcePrecedence     : 99
```

```
rmiPort                : 10000
role                   : slave
seqnoType              : java.lang.Long
serviceName            : firstrep
serviceType            : local
simpleServiceName      : firstrep
siteName               : default
sourceId               : host2
state                  : ONLINE
timeInStateSeconds     : 893.32
uptimeSeconds          : 9370.031
version                : Tungsten Clustering (for MySQL) 7.0.3 build 141
Finished status command...
```

Similar to the host specification, trepctl provides information for the default service. If you have installed multiple services, you must specify the service explicitly:

```
shell> trepctrl -service servicename status
```

If the service has been configured to operate on an alternative management port, this can be specified using the `-port` option. The default is to use port 10000.

The above command was executed on the Target host, `host2`. Some key parameter values from the generated output:

- *appliedLastEventId*

  This shows the last event from the source event stream that was applied to the database. In this case, the output shows that source of the data was a MySQL binary log. The portion before the colon, `mysql-bin.000064` is the filename of the binary log on the Source. The portion after the colon is the physical location, in bytes, within the binary log file.

- *appliedLastSeqno*

  The last sequence number for the transaction from the Tungsten stage that has been applied to the database. This indicates the last actual transaction information written into the Target database.

  When using parallel replication, this parameter returns the minimum applied sequence number among all the channels applying data.

- *appliedLatency*

  The *appliedLatency* is the latency between the commit time and the time the last committed transaction reached the end of the corresponding pipeline within the replicator.

  In replicators that are operating with parallel apply, *appliedLatency* indicates the latency of the trailing channel. Because the parallel apply mechanism does not update all channels simultaneously, the figure shown may trail significantly from the actual latency.

- *masterConnectUri*

  On an Extractor, the value will be empty.

  On an Applier, the URI of the Extractor Tungsten Clustering (for MySQL) from which the transaction data is being read from. The value supports multiple URIs (separated by comma) for topologies with multiple Sources.

- *maximumStoredSeqNo*

  The maximum transaction ID that has been stored locally on the machine in the THL. Because Tungsten Clustering (for MySQL) operates in stages, it is sometimes important to compare the sequence and latency between information being ready from the source into the THL, and then from the THL into the database. You can compare this value to the *appliedLastSeqno*, which indicates the last sequence committed to the database. The information is provided at a resolution of milliseconds.

- *pipelineSource*

  Indicates the source of the information that is written into the THL. For an Extractor, *pipelineSource* is the MySQL binary log. For an Applier, *pipelineSource* is the THL of the Extractor.

- *relativeLatency*

  The relativeLatency is the latency between now and timestamp of the last event written into the local THL. An increasing *relativeLatency* indicates that the replicator may have stalled and stopped applying changes to the dataserver.

- *state*

  Shows the current status for this node. In the event of a failure, the status will indicate that the node is in a state other than `ONLINE`. The *timeInStateSeconds* will indicate how long the node has been in that state, and therefore how long the node may have been down or unavailable.

The easiest method to check the health of your replication is to compare the current sequence numbers and latencies for each Applier compared to the Extractor. For example:

```
shell> trepctl -host host2 status|grep applied
appliedLastEventId     : mysql-bin.000076:0000000087725114;0
appliedLastSeqno       : 2445
appliedLatency         : 252.0
...
shell> trepctl -host host1 status|grep applied
appliedLastEventId     : mysql-bin.000076:0000000087725114;0
appliedLastSeqno       : 2445
appliedLatency         : 2.515
```

Note

For parallel replication and complex multi-service replication structures, there are additional parameters and information to consider when checking and confirming the health of the cluster.

The above indicates that the two hosts are up to date, but that there is a significant latency on the Applier for performing updates.

Tungsten Replicator Schema

Tungsten Replicator creates and updates information in a special schema created within the database which contains more specific information about the replication information transferred. The schema is named according to the servicename of the replication configuration, for example if the server is `firstrep`, the schema will be `tungsten_firstrep`.

The sequence number of the last transferred and applied transaction is recorded in the `trep_commit_seqno` [693] table.

6.3.6.1. 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` [145]

  The replicator service is starting up and reading the replicator properties configuration file.

- `OFFLINE:NORMAL` [145]

  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` [145]

  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.

- `SUSPECT` [145]

  This replicator state is only seen within cctrl. The underlying replicator status should be checked to see the full error state by reviewing the output of trepctl status.

- `GOING-ONLINE:RESTORING` [145]

  The replicator is preparing to go online and is currently restoring data from a backup.

- `GOING-ONLINE:SYNCHRONIZING` [145]

  The replicator is preparing to go online and is currently preparing to process any outstanding events from the incoming event stream. This mode occurs when an Applier has been switched online after maintenance, or in the event of a temporary network error where the Applier has reconnected to the Extractor.

- `ONLINE`

  The node is currently online and processing events, reading incoming data and applying those changes to the database as required. In this mode the current status and position within the replication stream is recorded and can be monitored. Replication will continue until an error or administrative condition switches the node into the `OFFLINE` state.

- `GOING-OFFLINE` [145]

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 [145] state. This state may also be seen in a node where auto-enable is disabled after a start or restart operation.

- ONLINE:DEGRADED [146]

  This status will be seen on an Extractor replicator and is indicative of the replicator loosing connectivity to the Source Database that it is extracting from. The replicator will still continue to extract entries from the binary log that have not yet been processed. After extracting all log entries, the replicator will proceed to the ONLINE:DEGRADED-BINLOG-FULLY-READ [146] state.

- ONLINE:DEGRADED-BINLOG-FULLY-READ [146]

  This status will be seen on an Extractor replicator following the ONLINE:DEGRADED [146] state and indicates that the replicator has completed reading all binglog entries. In a clustering environment, it indicates to the cluster that failover can now proceed.

In general, the state of a node during operation will go through a natural progression within certain situations. In normal operation, assuming no failures or problems, and no management requested offline, a node will remain in the ONLINE state indefinitely.

Maintenance on Tungsten Replicator or the dataserver must be performed while in the OFFLINE state. In the OFFLINE state, write locks on the THL and other files are released, and reads or writes from the dataserver are stopped until the replicator is ONLINE again.

6.3.6.2. 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
Administrator puts node into offline state	GOING-OFFLINE [145]
Node is offline	OFFLINE:NORMAL [145]
Administrator puts node into online state	GOING-ONLINE:SYNCHRONIZING [145]
Node catches up with Extractor	ONLINE

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
Failure causes the node to go offline	OFFLINE:ERROR [145]
Administrator fixes error and puts node into online state	GOING-ONLINE:SYNCHRONIZING [145]
Node catches up with Extractor	ONLINE

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
Failure causes the node to go offline	OFFLINE:ERROR [145]
Administrator restores node from backup data	GOING-ONLINE:RESTORING [145]
Once restore is complete, node synchronizes with the Extractor	GOING-ONLINE:SYNCHRONIZING [145]
Node catches up with Extractor	ONLINE

6.3.6.3. Changing Replicator States

You can manually change the replicator states on any node by using the trepctl command.

To switch to the OFFLINE state if you are currently ONLINE:

```
shell> trepctl offline
```

Unless there is an error, no information is reported. The current state can be verified using the trepctl status:

```
shell> trepctl status
Processing status command...
...
```

```
state                : OFFLINE:NORMAL
timeInStateSeconds   : 21.409
uptimeSeconds        : 935.072
```

To switch back to the ONLINE state:

```
shell> trepctl online
```

When using replicator states in this manner, the replication between hosts is effectively paused. Any outstanding events from the Extractor will be replicated to the Applier with the replication continuing from the point where the node was switched to the OFFLINE state. The sequence number and latency will be reported accordingly, as seen in the example below where the node is significantly behind the Primary:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000004:0000000005162941;0
appliedLastSeqno       : 21
appliedLatency         : 179.366
```

6.3.7. Changing Datasource States

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

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

```
datasource DATASOURCENAME SUBCOMMAND
```

For example, to shun the node host1:

```
[LOGICAL:EXPERT] /alpha > datasource db1 shun
```

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

6.3.7.1. Shunning a Datasource

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

Datasources can be automatically or manually shunned:

- *Automatic* shunning occurs when the dataservice is in *AUTOMATIC* policy mode, and the datasource has become unresponsive or fails. For example, when a Primary fails, an automatic switch to a new Primary is performed, and the old Primary is shunned.
- *Manual* shunning occurs when the shun command is given to a datasource. Manual shunning can be used to set a datasource into a state that allows for maintenance and management operations to be performed on the datasource.

To manually shun the datasource:

```
[LOGICAL:EXPERT] /alpha > datasource db2 shun
DataSource 'db2' set to SHUNNED
```

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

```
+----------------------------------------------------------------------------+
|db2(slave:SHUNNED(MANUALLY-SHUNNED), progress=10, latency=42.292)           |
|STATUS [SHUNNED] [2025/01/24 03:55:28 PM UTC]                               |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=9, active=0)                                          |
+----------------------------------------------------------------------------+
```

Important

Shunning a datasource does not stop the replicator; replication will continue on a shunned datasource until the replication service is explicitly placed into the offline state.

Warning

Unless unavoidable, and directed by Continuent Support, you should never manually shun the primary node

The level of the shunning is reported in the status as a manual operation. A manually shunned datasource can be enabled using the datasource recover command, see Section 6.3.7.2, “Recover a Datasource” .

6.3.7.2. Recover a Datasource

The datasource recover command is a deeper operation that performs a number of actions to get the datasource back into the operational state. When used, the datasource recover command performs the following operations:

- Restarts failed or stopped services
- Changes the datasource configuration so that it is configured as a Primary or Replica. For example, an automatically failed Primary will be reconfigured to operate as a Replica to the current Primary.
- Restarts the replicator service in the Replica or Primary role as appropriate

In all cases, the datasource recover command should be used if a datasource is offline or shunned, and it can be used at all times to get a datasource back in to operational state within the cluster. In essence, recover performs the same operations automatically as would be performed manually to get the node into the right state.

```
[LOGICAL] /alpha > datasource host3 recover
RECOVERING DATASOURCE 'db2@alpha'
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'db2'
Verified that DB server notification 'db2' is in state 'ONLINE'
DATA SERVER 'db2' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'db2@alpha' TO A SLAVE USING 'db3@alpha' AS THE MASTER
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'db2'
Verified that DB server notification 'db2' is in state 'ONLINE'
DATA SERVER 'db2' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERY OF 'db2@alpha' WAS SUCCESSFUL
```

During the recovery process, the node will be checked, replication reconfigured, and the node brought back in to active service. If this process fails because the databases and replication states are out of sync and cannot be recovered, Tungsten Cluster may advise that a backup of another datasource and recovery to this datasource is performed. For more information on restoring from backups, see Section 6.11, “Restoring a Backup” .

6.3.7.3. Offline a Datasource

A datasource can be explicitly placed into offline mode. In offline mode, client applications connections to datasources are paused. When switching to `OFFLINE` mode existing connections are given a five-second grace period to complete their operations before being forced to disconnect. Replicator operation is not affected.

To set a datasource offline:

```
[LOGICAL] /alpha > datasource db2 offline
DataSource 'db2@alpha' is now OFFLINE
```

If the dataservice is in *`AUTOMATIC`* policy mode, and there are no other faults in the datasource, it will automatically be placed into `ONLINE` mode. To set a datasource offline the dataservice must be in *`MAINTENANCE`* or *`MANUAL`* policy modes.

6.3.7.4. Mark a Datasource as Standby

`standby` datasources receive replication data, but are not part of the load-balancing provided by Tungsten Connector. In the event of a failover situation, a *`standby`* datasource will be enabled within the cluster as a Replica. Because the *`standby`* datasource is up to date with respect to the replication of data, this process is instantaneous. The connector will be updated, and the new Replica will operate as a read-only datasource.

To configure a datasource as a *`standby`*:

```
[LOGICAL] /alpha > datasource db2 standby
Datasource 'db2' now has role 'standby'
```

To clear the *`standby`* state:

```
[LOGICAL] /alpha > datasource db2 clear standby
Datasource 'db2' now has role 'slave'
```

Note

When a Replica goes into standby mode, it will finish running any SQL queries that were started before it went into standby mode. New queries, even on the same connection, will not be directed to a Replica that has just gone into standby

6.3.7.5. Mark a Datasource as Archive

An *`archive`* datasource receives replication data and is included as part of the load-balancing provided by Tungsten Connector. It is excluded from failover switches and will not be used as a Primary in the event of a failure. To mark a datasource as an archive datasource:

```
[LOGICAL] /alpha > datasource db2 set archive
```

To remove the archive role:

```
[LOGICAL] /alpha > datasource db2 clear archive
```

The archive role is a temporary requirement, and will not survive a re-install or upgrade.

6.3.8. Datasource Statuses

In addition to the overall state, all datasources have a specific status that indicates the current health and operation, rather than the configured state for that datasource. For example, a datasource can be in the online state, but have a DIMINISHED [149] status if there is a recoverable problem with one of the datasource components.

The purpose of the STATUS field is to provide standard, datasource-state-specific values for ease of parsing and backwards-compatibility with older versions of the cctrl command.

The STATUS field is effectively the same information as the datasource state that appears on the first line after the colon (:), just presented slightly differently.

Here are the possible values for STATUS, showing the Datasource State first, and the matching STATUS second:

Datasource State	Alert STATUS
ONLINE	OK
OFFLINE	WARN (for non-composite datasources)
OFFLINE	DIMINISHED (for composite passive replica)
OFFLINE	CRITICAL (for composite active primary)
FAILED	CRITICAL
SHUNNED	SHUNNED

Any other datasource status sets the STATUS to UNKNOWN.

- **OK** [149]

  The OK [149] status indicates that the datasource is currently operating correctly.

- **DIMINISHED** [149]

  A DIMINISHED [149] status indicates that there is a problem with one of the dataservice services which is causing a reduced level of expected service. For example, in the sample output below, the reason is indicated as a stopped replicator service.

  ```
  +---------------------------------------------------------------------------+
  |db1(master:ONLINE)                                                         |
  |STATUS [DIMINISHED] [2025/01/24 03:23:02 PM UTC]                           |
  |REASON[REPLICATOR STOPPED]                                                 |
  +---------------------------------------------------------------------------+
  |  MANAGER(state=ONLINE)                                                    |
  |  REPLICATOR(state=STOPPED)                                                |
  |  DATASERVER(state=ONLINE)                                                 |
  |  CONNECTIONS(created=195, active=0)                                       |
  +---------------------------------------------------------------------------+
  ```

  The underlying service fault should be fixed and the status rechecked.

If all the services are ONLINE, but one node is in the DIMINISHED [149] state, you should let the auto-recovery process complete. To do this:

1. Place the cluster into *AUTOMATIC* mode:

   ```
   [LOGICAL] /alpha > set policy automatic
   ```

2. Set the status of the node in the DIMINISHED [149] state to OFFLINE:

   ```
   [LOGICAL] /alpha > datasource db1 offline
   ```

Automatic recovery will then recover the node for you.

6.3.9. Datasource States and Policy Mode Interactions

States can be explicitly set through cctrl, however, depending on the current policy mode, the actual status set may be different from that initially set. For example, when shunning a datasource, the datasource will immediately go into SHUNNED state.

```
[LOGICAL] /alpha > datasource db2 shun
DataSource 'db2' set to SHUNNED
```

To bring the datasource back into operation, it must be brought back using the recover command:

```
[LOGICAL] /alpha > datasource db2 recover
DataSource 'db2' is now OFFLINE
```

The datasource recover command performs whatever steps are necessary to bring the datasource back into operation within the dataservice. Even for manually shunned datasources, there may be additional configuration or recovery steps required.

If the dataservice policy mode is *MANUAL* or *MAINTENANCE* modes, the datasource remains in the OFFLINE state until manually put ONLINE .

6.4. Policy Modes

The dataservice operates using a policy mode, which configures how the dataservice management system responds to different events and operations within the dataservice. The policy mode can be set at will and enables maintenance and administration to be carried out without triggering the automatic failure and recovery procedures for operations that would otherwise trigger an automated response.

The procedure for how these operations are carried out are defined through a series of rules, with different policies applying different sets of the individual rules. The setting of the policy mode is dataservice-wide and instantaneous.

	Policy Mode		
Ruleset	Automatic	Manual	Maintenance
Monitoring	Yes	Yes	Yes
Fault Detection	Yes	Yes	No
Failure Fencing	Yes	Yes	No
Failure Recovery	Yes	No	No

The individual policy modes are described below:

- *AUTOMATIC* Policy Mode

  In automatic mode, the following operations and status changes happen automatically, managed by the coordinator:

  - Failed Replica datasources are automatically marked as failed, temporarily removed from the dataservice, with application connections redirect to the other nodes in the dataservice. When the datasource becomes available, the node is automatically recovered to the dataservice.
  - Failed Primary datasources are automatically shunned and switched to the most up to date Replica within the dataservice, which becomes the Primary and remaining Replicas point to the newly promoted Primary.

    > **Note**
    >
    > Automatic policy mode operates within a single dataservice only. Within a composite dataservice there is no automatic failover.

- *MANUAL* Policy Mode

  In the *MANUAL* policy mode, the dataservice identifies and isolates datasources when they fail, but automatic failover (for Primary datasources) and recovery is disabled.

- *MAINTENANCE* Policy Mode

  In *MAINTENANCE* policy mode all rules are disabled. Maintenance mode should be used when performing datasource or host maintenance that would otherwise trigger an automated fencing or recovery process.

  MAINTENANCE mode should be used when administration or maintenance is required on the datasource, software, or operating system.

6.4.1. Setting Policy Modes

To set the policy, use the set command with the policy option. For example, to switch the current dataservice policy mode to manual:

```
[LOGICAL] /alpha > set policy manual
policy mode is now MANUAL
```

Policy mode changes are global, affecting the operation of all the members of the dataservice.

The current policy mode is shown when running ls within cctrl , see Section 6.3, “Checking Dataservice Status” .

6.4.2. Setting Policy Modes in a Composite cluster

In a Composite Active/Passive, Composite Active/Active or Composite Dynamic Active/Active cluster, the policy can also be set at the local cluster level or the top composite level.

Setting the policy at the local cluster level will only change that cluster, for example, setting the policy at the `alpha` cluster level will only apply to that cluster, all other clusters will remain in whatever policy they are currently in. At the top composite level this will then show as a *`MIXED`* policy, for example:

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

[LOGICAL] /alpha > ls
COORDINATOR[db2:MAINTENANCE:ONLINE]
...

[LOGICAL] /alpha > use global
[LOGICAL] /global > ls

COORDINATOR[db2:MIXED:ONLINE]
   alpha:COORDINATOR[db2:MAINTENANCE:ONLINE]
   beta:COORDINATOR[db6:AUTOMATIC:ONLINE]
...
```

Setting the policy at the top composite level will cascade down and apply to all clusters, for example:

```
[LOGICAL] /global > ls

COORDINATOR[db2:AUTOMATIC:ONLINE]
   alpha:COORDINATOR[db2:AUTOMATIC:ONLINE]
   beta:COORDINATOR[db6:AUTOMATIC:ONLINE]
...

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

[LOGICAL] /global > ls

COORDINATOR[db2:MAINTENANCE:ONLINE]
   alpha:COORDINATOR[db2:MAINTENANCE:ONLINE]
   beta:COORDINATOR[db6:MAINTENANCE:ONLINE]
```

6.5. Switching Primary Hosts

The Primary host within a dataservice can be switched, either automatically, or manually. Automatic switching occurs when the dataservice is in the *`AUTOMATIC`* policy mode, and a failure in the underlying datasource has been identified. The automatic process is designed to keep the dataservice running without requiring manual intervention.

Manual switching of the Primary can be performed during maintenance operations, for example during an upgrade or dataserver modification. In this situation, the Primary must be manually taken out of service, but without affecting the rest of the dataservice. By switching the Primary to another datasource in the dataservice, the original Primary can be put offline, or shunned, while maintenance occurs. Once the maintenance has been completed, the datasource can be re-enabled, and either remain as the a Replica, or switched back as the Primary datasource.

Switching a datasource, whether automatically or manually, occurs while the dataservice is running, and without affecting the operation of the dataservice as a whole. Client application connections through Tungsten Connector are automatically reassigned to the datasources in the dataservice, and application operation will be unaffected by the change. Switching the datasource manually requires a single command that performs all of the required steps, monitoring and managing the switch process.

Switching the Primary, manually or automatically, performs the following steps within the dataservice:

1. Set the Primary node to offline state. New connections to the Primary are rejected, and writes to the Primary are stopped.
2. On the Replica that will be promoted, switch the datasource offline. New connections are rejected, stopping reads on this Replica.
3. Kill any outstanding client connections to the Primary data source, except those belonging to the `tungsten` account.
4. Send a heartbeat transaction between the Primary and the Replica, and wait until this transaction has been received. Once received, the THL on Primary and Replica are up to date.
5. Perform the switch:

- Configure all remaining replicators offline
- Configure the selected Replica as the new Primary.
- Set the new Primary to the online state.
- New connections to the Primary are permitted.

6. Configure the remaining Replicas to use the new Primary as the Primary datasource.
7. Update the connector configurations and enable client connections to connect to the Primaries and Replicas.

The switching process is monitored by Tungsten Cluster, and if the process fails, either due to a timeout or a recoverable error occurs, the switch operation is rolled back, returning the dataservice to the original configuration. This ensures that the dataservice remains operational. In some circumstances, when performing a manual switch, the command may need to be repeated to ensure the requested switch operation completes.

The process takes a finite amount of time to complete, and the exact timing and duration will depend on the state, health, and database activity on the dataservice. The actual time taken will depend on how up to date the Replica being promoted is compared to the Primary. The switch will take place regardless of the current status after a delay period.

6.5.1. Automatic Primary Failover

When the dataservice policy mode is `AUTOMATIC`, the dataservice will automatically failover the Primary host when the existing Primary is identified as having failed or become unavailable.

For example, when the Primary host `db1` becomes unavailable because of a network problem, the dataservice automatically switches to `db3`. The dataservice status is updated accordingly, showing the automatically shunned `db2`:

```
[LOGICAL:EXPERT] /alpha > ls

COORDINATOR[db1:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@db1[7435](ONLINE, created=2, active=0)                            |
|connector@db2[7472](ONLINE, created=2, active=0)                            |
|connector@db3[7468](ONLINE, created=2, active=0)                            |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|db1(master:SHUNNED(FAILED-OVER-TO-db3), progress=8, THL latency=0.981)      |
|STATUS [SHUNNED] [2025/01/27 01:51:23 PM UTC]                               |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=DEGRADED)                                   |
|  DATASERVER(state=STOPPED)                                                 |
|  CONNECTIONS(created=4, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db2(slave:ONLINE, progress=8, latency=1.004)                                |
|STATUS [OK] [2025/01/27 01:51:40 PM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db3(master:ONLINE, progress=10, THL latency=0.380)                          |
|STATUS [OK] [2025/01/27 01:51:27 PM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=2, active=0)                                          |
+----------------------------------------------------------------------------+
```

The status for the original Primary [`db1`] identifies the datasource as shunned, and indicates which datasource was promoted to the Primary in the `FAILED-OVER-TO-db3`.

A automatic failover can be triggered by using the datasource fail command:

```
[LOGICAL:EXPERT] /alpha > datasource db1 fail
```

This triggers the automatic failover sequence, and simulates what would happen if the specified host failed.

If db1 becomes available again, the datasource is not automatically added back to the dataservice, but must be explicitly re-added to the dataservice. The status of the dataservice once db1 returns is shown below:

```
[LOGICAL:EXPERT] /alpha > ls

...

+----------------------------------------------------------------------------+
|db1(master:SHUNNED(FAILED-OVER-TO-db3), progress=8, THL latency=0.981)      |
|STATUS [SHUNNED] [2025/01/27 01:51:23 PM UTC]                               |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=DEGRADED)                                   |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=4, active=0)                                          |
+----------------------------------------------------------------------------+

...
```

Because db1 was previously the Primary, the datasource recover command verifies that the server is available, configures the node as a Replica of the newly promoted Primary, and re-enables the services:

```
[LOGICAL:EXPERT] /alpha > datasource db1 recover
RECOVERING DATASOURCE 'db1@alpha'
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'db1'
Verified that DB server notification 'db1' is in state 'ONLINE'
DATA SERVER 'db1' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'db1@alpha' TO A SLAVE USING 'db3@alpha' AS THE MASTER
SETTING THE ROLE OF DATASOURCE 'db1@alpha' FROM 'master' TO 'slave'
RECOVERY OF 'db1@alpha' WAS SUCCESSFUL
```

If the command is successful, then the node should be up and running as a Replica of the new Primary.

The recovery process can fail if the THL data and dataserver contents do not match, for example when statements have been executed on a Replica. For information on recovering from failures that recover cannot fix, see Section 6.6.1.3, “Replica Datasource Extended Recovery” .

6.5.2. Manual Primary Switch

In a single data service dataservice configuration, the Primary can be switched between nodes within the dataservice manually using cctrl . The switch command performs the switch operation, annotating the progress.

```
[LOGICAL:EXPERT] /alpha > switch
SET POLICY: AUTOMATIC => MAINTENANCE
EVALUATING SLAVE: db3(stored=14, applied=14, latency=0.682, datasource-group-id=0)
EVALUATING SLAVE: db2(stored=14, applied=14, latency=0.686, datasource-group-id=0)
SELECTED SLAVE: db3@alpha
Savepoint switch_2(cluster=alpha, source=db2, created=2025/01/27 13:54:48 UTC) created
PURGE REMAINING ACTIVE SESSIONS ON CURRENT MASTER 'db1@alpha'
PURGED A TOTAL OF 0 ACTIVE SESSIONS ON MASTER 'db1@alpha'
FLUSH TRANSACTIONS ON CURRENT MASTER 'db1@alpha'
PUT THE NEW MASTER 'db3@alpha' ONLINE
PUT THE PRIOR MASTER 'db1@alpha' ONLINE AS A SLAVE
SWITCH TO 'db3@alpha' WAS SUCCESSFUL
```

By default, switch chooses the most up to date Replica within the dataservice (db3 in the above example), but an explicit Replica can also be selected:

```
[LOGICAL:EXPERT] /alpha > switch to db2
SET POLICY: AUTOMATIC => MAINTENANCE
EVALUATING SLAVE: db2(stored=22, applied=22, latency=0.974, datasource-group-id=0)
SELECTED SLAVE: db2@alpha
Savepoint switch_4(cluster=alpha, source=db2, created=2025/01/27 13:56:07 UTC) created
PURGE REMAINING ACTIVE SESSIONS ON CURRENT MASTER 'db1@alpha'
PURGED A TOTAL OF 0 ACTIVE SESSIONS ON MASTER 'db1@alpha'
FLUSH TRANSACTIONS ON CURRENT MASTER 'db1@alpha'
PUT THE NEW MASTER 'db2@alpha' ONLINE
PUT THE PRIOR MASTER 'db1@alpha' ONLINE AS A SLAVE
SWITCH TO 'db2@alpha' WAS SUCCESSFUL
```

With the previous example, the switch occurred specifically to the node db2 .

6.6. Datasource Recovery Steps

When a datasource within the dataservice fails, the exact response by the dataservice is dependent on the dataservice policy mode. Different policy modes either cope with the failure or recovery process automatically, or a prescribed sequence must be followed.

Recovery can normally be achieved by following these basic steps:

- Use the recover command

  The recover command performs a number of steps to try and return the dataservice to the operational state, but works only if there is an existing Primary within the current configuration. Operations conducted automatically include Replica recovery, and reconfiguring roles. For example:

```
[LOGICAL] /alpha > recover
RECOVERING DATASERVICE 'alpha
SET POLICY: AUTOMATIC => MAINTENANCE
FOUND PHYSICAL DATASOURCE TO RECOVER: 'db1@alpha'
RECOVERING DATASOURCE 'db1@alpha'
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'db1'
Verified that DB server notification 'db1' is in state 'ONLINE'
DATA SERVER 'db1' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'db1@alpha' TO A SLAVE USING 'db3@alpha' AS THE MASTER
SETTING THE ROLE OF DATASOURCE 'db1@alpha' FROM 'master' TO 'slave'
RECOVERY OF DATA SERVICE 'alpha' SUCCEEDED
REVERT POLICY: MAINTENANCE => AUTOMATIC
RECOVERED 1 DATA SOURCES IN SERVICE 'alpha'
```

> **Note**
>
> The output from the recover command will be longer in Composite clusters as the recovery will also ensure remote cluster connectivity is established between the primary and relay hosts

Different scenarios may present different routes to recovery. The folloing chapters outline simple recovery of Primary and Replica nodes in the most common scenarios.

Where the recovery of a cluster is presenting issues that are not clear or easily recoverable using the steps outline, it is recommended to contact Continuent Support

6.6.1. Recover a failed Replica

As long as the cluster is operating in the *AUTOMATIC* policy mode, then in most cases the managers will attempt to recover a failed replica automatically.

If automatic recovery fails, further investigation will be required to establish the cause of failure. Once the fault has been resolved and the host is viable (ie no data corruption) and available again, it can be recovered back into Replica mode using either the recover command, or the specific single datasource command datasource db1 recover:

```
[LOGICAL:EXPERT] /alpha > recover
RECOVERING DATASERVICE 'alpha
SET POLICY: AUTOMATIC => MAINTENANCE
FOUND PHYSICAL DATASOURCE TO RECOVER: 'db1@alpha'
RECOVERING DATASOURCE 'db1@alpha'
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'db1'
Verified that DB server notification 'db1' is in state 'ONLINE'
DATA SERVER 'db1' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'db1@alpha' TO A SLAVE USING 'db3@alpha' AS THE MASTER
SETTING THE ROLE OF DATASOURCE 'db1@alpha' FROM 'master' TO 'slave'
RECOVERY OF DATA SERVICE 'alpha' SUCCEEDED
REVERT POLICY: MAINTENANCE => AUTOMATIC
RECOVERED 1 DATA SOURCES IN SERVICE 'alpha'
```

or:

```
[LOGICAL:EXPERT] /alpha > datasource db1 recover
RECOVERING DATASOURCE 'db1@alpha'
STARTING SERVICE 'db1/mysql'
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'db1'
Verified that DB server notification 'db1' is in state 'ONLINE'
DATA SERVER 'db1' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'db1@alpha' TO A SLAVE USING 'db3@alpha' AS THE MASTER
RECOVERY OF 'db1@alpha' WAS SUCCESSFUL
```

The single recover command will attempt to recover all the Replica resources in the cluster, bringing them all online and back into service. The command operates on all shunned or failed Replicas, and only works if there is an active Primary available.

In some cases, the datasource may show as `ONLINE` and the recover command does not bring the datasource online, particularly with the following error:

```
The datasource 'db1' is not FAILED or SHUNNED and cannot be recovered.
```

Checking the datasource status in cctrl the replicator service has failed, but the datasource shows as online:

```
+----------------------------------------------------------------------------+
|db1(slave:ONLINE, progress=-1, latency=-1.000)                              |
```

```
|STATUS [OK] [2025/01/27 01:59:33 PM UTC]                                      |
+------------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                       |
|  REPLICATOR(role=slave, master=db3, state=SUSPECT)                           |
|  DATASERVER(state=ONLINE)                                                    |
|  CONNECTIONS(created=4, active=0)                                            |
+------------------------------------------------------------------------------+
```

In this case, the datasource can be manually shunned, which will then enable the recover command to operate and bring the node back into operation.

6.6.1.1. Recover a Replica from manually shunned state

A Replica that has been manually shunned can be added back to the dataservice using the datasource recover command:

```
[LOGICAL:EXPERT] /alpha > ls

...
DATASOURCES:
+------------------------------------------------------------------------------+
|db1(slave:SHUNNED(MANUALLY-SHUNNED), progress=41, latency=59020.057)          |
|STATUS [SHUNNED] [2025/01/28 08:35:36 AM UTC]                                 |
+------------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                       |
|  REPLICATOR(role=slave, master=db3, state=ONLINE)                            |
|  DATASERVER(state=ONLINE)                                                    |
|  CONNECTIONS(created=0, active=0)                                            |
+------------------------------------------------------------------------------+
...

[LOGICAL:EXPERT] /alpha > datasource db1 recover
RECOVERING DATASOURCE 'db1@alpha'
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'db1'
Verified that DB server notification 'db1' is in state 'ONLINE'
DATA SERVER 'db1' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'db1@alpha' TO A SLAVE USING 'db3@alpha' AS THE MASTER
RECOVERY OF 'db1@alpha' WAS SUCCESSFUL

[LOGICAL:EXPERT] /alpha > ls

...
DATASOURCES:
+------------------------------------------------------------------------------+
|db1(slave:ONLINE, progress=41, latency=59020.000)                             |
|STATUS [OK] [2025/01/28 08:36:37 AM UTC]                                      |
+------------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                       |
|  REPLICATOR(role=slave, master=db3, state=ONLINE)                            |
|  DATASERVER(state=ONLINE)                                                    |
|  CONNECTIONS(created=0, active=0)                                            |
+------------------------------------------------------------------------------+
...
```

6.6.1.2. Provision or Reprovision a Replica

In the event that you cannot get the Replica to recover using the datasource recover command, you can re-provision the Replica from another Replica within your dataservice.

The command performs three operations automatically:

1. Performs a backup of a remote Replica
2. Copies the backup to the current host
3. Restores the backup

Warning

When using tprovision you must be logged in to the Replica that has failed or that you want to reprovision. You cannot reprovision a Replica remotely.

To use tprovision :

1. Log in to the failed Replica.
2. Select the active Replica within the dataservice that you want to use to reprovision the failed Replica. You may use the Primary but this will impact performance on that host. If you use MyISAM tables the operation will create some locking in order to get a consistent snapshot.

3. Run tprovision specifying the source you have selected:

```
shell> tprovision -s db2 -m xtrabackup

2025/01/27 15:59:15 | INFO Started script tprovision
2025/01/27 15:59:20 | INFO xtrabackup version is 8.0.35-31
2025/01/27 15:59:20 | INFO MySQL version is 8.0.40
2025/01/27 15:59:20 | INFO
source = db2
method = xtrabackup
parallel threads = 4
port = 22
topology = CLUSTER
...
...
Verified that DB server notification 'db1' is in state 'ONLINE'
DATA SERVER 'db1' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'db1@alpha' TO A SLAVE USING 'db3@alpha' AS THE MASTER
RECOVERY OF 'db1@alpha' WAS SUCCESSFUL
[LOGICAL] /alpha >
Exiting...

2025/01/27 15:59:48 | INFO Script ended
```

tprovision handles the cluster status, backup, restore, and repositioning of the replication stream so that restored Replica is ready to start operating again.

> **Note**
>
> For a full explanation of using tprovison see Section 9.23, “The tprovision Script”

> **Important**
>
> When using a Multi-Site/Active-Active topology the additional replicator must be put offline before restoring data and put online after completion.

```
shell> mm_trepctl offline
shell> tprovision -s db2 -m xtrabackup
shell> mm_trepctl online
shell> mm_trepctl status
```

6.6.1.3. Replica Datasource Extended Recovery

If the current Replica will not recover, but the replicator state and sequence number are valid, the Replica is pointing to the wrong Primary, or still mistakenly has the Primary role when it should be a Replica, then the Replica can be forced back into the Replica state.

For example, in the output from ls in cctrl below, the replicator on `db2` is mistakenly identified as the Primary, even though `db1` is correctly operating as the Primary.

> **Important**
>
> It is important to note that in the scenario shown below, as the `db2` host is in a `SHUNNED` state, and the Replicator is `OFFLINE`, it is therefore isolated from applications and won't be operating as the incorrectly labelled role. `db1` is the only functional primary node in the cluster.

```
COORDINATOR[db3:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@db1[2096](ONLINE, created=2, active=0)                            |
|connector@db2[2092](ONLINE, created=2, active=0)                            |
|connector@db3[2107](ONLINE, created=2, active=0)                            |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|db1(master:ONLINE, progress=43, THL latency=0.073)                          |
|STATUS [OK] [2025/01/28 08:39:09 AM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+

+----------------------------------------------------------------------------+
|db2(slave:SHUNNED(MANUALLY-SHUNNED), progress=-1, latency=-1.000)           |
|STATUS [SHUNNED] [2025/01/28 08:42:11 AM UTC]                               |
```

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

+----------------------------------------------------------------------+
|db3(slave:ONLINE, progress=43, latency=0.319)                         |
|STATUS [OK] [2025/01/28 08:42:14 AM UTC]                              |
+----------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                               |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                    |
|  DATASERVER(state=ONLINE)                                            |
|  CONNECTIONS(created=0, active=0)                                    |
+----------------------------------------------------------------------+
```

The datasource `db2` can be brought back online using this sequence:

1. Enable set force mode:

   ```
   [LOGICAL:EXPERT] /alpha > set force true
   FORCE: true
   ```

2. Switch the replicator offline:

   ```
   [LOGICAL:EXPERT] /alpha > replicator db2 offline
   Replicator 'db2' is now OFFLINE
   ```

3. Set the replicator to *`Replica`* operation:

   ```
   [LOGICAL:EXPERT] /alpha > replicator db2 slave
   Replicator 'db2' is now a slave of replicator 'db1'
   ```

   In some instances you may need to explicitly specify which node is your Primary when you configure the Replica; appending the Primary hostname to the command specifies the Primary host to use:

   ```
   [LOGICAL:EXPERT] /alpha > replicator db2 slave db1
   Replicator 'db2' is now a slave of replicator 'db1'
   ```

4. Switch the replicator service online:

   ```
   [LOGICAL:EXPERT] /alpha > replicator db2 online
   Replicator 'db2' is now ONLINE
   ```

5. Ensure the datasource is correctly configured as a Replica:

   ```
   [LOGICAL:EXPERT] /alpha > datasource db2 slave
   Datasource 'db2' now has role 'slave'
   ```

6. Recover the Replica back to the dataservice:

   ```
   [LOGICAL:EXPERT] /alpha > datasource db2 recover
   DataSource 'db2' is now OFFLINE
   ```

Datasource `db2` should now be back in the dataservice as a working datasource.

Similar processes can be used to force a datasource back into the *`Primary`* role if a switch or recover operation failed to set the role properly.

If the recover command fails, there are a number of solutions that may bring the dataservice back to the normal operational state. The exact method will depend on whether there are other active Replicas (from which a backup can be taken) or recent backups of the Replica are available, and the reasons for the original failure. Some potential solutions include

- If there is a recent backup of the failed Replica, restore the Replica using that backup. The latest backup can be restored using Section 6.11, “Restoring a Backup” .
- If there is no recent backup, but have another Replica from which you can recover the failed Replica, the node should be rebuilt using the backup from another Replica. See Section 6.11.3, “Restoring from Another Replica” .

6.6.2. Recover a failed Primary

When a Primary datasource is automatically failed over in *`AUTOMATIC`* policy mode, providing the faut has been resolved and the host is viable, then the datasource can be brought back into the dataservice as a Replica by using the recover command:

```
[LOGICAL:EXPERT] /alpha > ls
```

```
...
+----------------------------------------------------------------------------+
|db3(master:SHUNNED(FAILED-OVER-TO-db2), progress=43, THL latency=0.073)     |
|STATUS [SHUNNED] [2025/01/28 09:07:59 AM UTC]                               |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=DEGRADED)                                   |
|  DATASERVER(state=STOPPED)                                                 |
|  CONNECTIONS(created=6, active=0)                                          |
+----------------------------------------------------------------------------+
...

[LOGICAL:EXPERT] /alpha > datasource db3 recover
RECOVERING DATASOURCE 'db3@alpha'
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'db3'
Verified that DB server notification 'db3' is in state 'ONLINE'
DATA SERVER 'db3' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'db3@alpha' TO A SLAVE USING 'db2@alpha' AS THE MASTER
SETTING THE ROLE OF DATASOURCE 'db3@alpha' FROM 'master' TO 'slave'
RECOVERY OF 'db3@alpha' WAS SUCCESSFUL
```

The recovered datasource will be added back to the dataservice as a Replica.

6.6.2.1. Recover when there are no Primaries

When there are no Primaries available, due to a failover of a Primary, or multiple host failure it is possible to bring the cluster back online by manually forcing a node to be a new primary.

This action should not be taken blindly, first you must understand why the failure happened in the first place and ensure the node you wish to bring online as the primary is viable and the most up to date.

If this is not immediately known or obvious, the timestamps within each individual datasource block may provide some assistance. In the event that multiple nodes are labelled as `master`, it would indicate a series of rolling failovers, the latest time would indicate the last node to hold the primary role.

- **Warning**

  If in any doubt about the actions to be taken, contact Continuent Support for assistance.

  To use, first you should examine the state of the dataservice and choose which datasource is the most up to date or canonical. For example, within the following output, only `db3` is labelled as a master and therefore is the most obvious candidate

```
[LOGICAL] /alpha > ls

COORDINATOR[db3:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@db1[2096](ONLINE, created=0, active=0)                            |
|connector@db2[2092](ONLINE, created=0, active=0)                            |
|connector@db3[2107](ONLINE, created=0, active=0)                            |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|db1(slave:FAILED(DATASERVER 'db1@alpha' STOPPED), progress=40, latency=1.000)|
|STATUS [CRITICAL] [2025/01/27 04:10:47 PM UTC]                              |
|REASON[DATASERVER 'db1@alpha' STOPPED]                                      |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db3, state=SYNCHRONIZING)                   |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db2(slave:FAILED(DATASERVER 'db2@alpha' STOPPED), progress=40, latency=0.000)|
|STATUS [CRITICAL] [2025/01/27 04:10:47 PM UTC]                              |
|REASON[DATASERVER 'db2@alpha' STOPPED]                                      |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db3, state=SYNCHRONIZING)                   |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db3(master:SHUNNED(FAILOVER-ABORTED AFTER UNABLE TO COMPLETE FAILOVER FOR   |
|DATASOURCE 'db3'. CHECK COORDINATOR MANAGER LOG), progress=-1, THL           |
|latency=-1.000)                                                             |
|STATUS [SHUNNED] [2025/01/27 04:10:50 PM UTC]                               |
+----------------------------------------------------------------------------+
```

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

Since all nodes are either SHUNNED or FAILED a simple recover would fail as there are no online master nodes available:

```
[LOGICAL] /alpha > recover
RECOVERING DATASERVICE 'alpha
SET POLICY: AUTOMATIC => MAINTENANCE
REVERT POLICY: MAINTENANCE => AUTOMATIC
DATA SERVICE 'alpha' DOES NOT HAVE AN ACTIVE PRIMARY
```

Once a host has been chosen, call the set force true and welcome commands specifying the full hostname of the chosen datasource:

```
[LOGICAL] /alpha > set force true
FORCE: true

[LOGICAL] /alpha > datasource db3 welcome

WARNING: This is an expert-level command:
Incorrect use may cause data corruption
or make the cluster unavailable.

Do you want to continue? (y/n)> y
DataSource 'db3@alpha' is now OFFLINE
```

As the cluster is in the *AUTOMATIC* policy mode, and providing the remaining hosts are healthy, the cluster will automatically recover itself, as shown by the ls command executed shortly after:

```
[LOGICAL] /alpha > ls
COORDINATOR[db3:AUTOMATIC:ONLINE]

ROUTERS:
+-------------------------------------------------------------------------------+
|connector@db1[2096](ONLINE, created=2, active=0)                               |
|connector@db2[2092](ONLINE, created=2, active=0)                               |
|connector@db3[2107](ONLINE, created=2, active=0)                               |
+-------------------------------------------------------------------------------+

DATASOURCES:
+-------------------------------------------------------------------------------+
|db1(slave:ONLINE, progress=42, latency=0.213)                                  |
|STATUS [OK] [2025/01/28 08:36:37 AM UTC]                                       |
+-------------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                        |
|  REPLICATOR(role=slave, master=db3, state=ONLINE)                             |
|  DATASERVER(state=ONLINE)                                                     |
|  CONNECTIONS(created=0, active=0)                                             |
+-------------------------------------------------------------------------------+
+-------------------------------------------------------------------------------+
|db2(slave:ONLINE, progress=42, latency=0.240)                                  |
|STATUS [OK] [2025/01/28 08:34:29 AM UTC]                                       |
+-------------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                        |
|  REPLICATOR(role=slave, master=db3, state=ONLINE)                             |
|  DATASERVER(state=ONLINE)                                                     |
|  CONNECTIONS(created=0, active=0)                                             |
+-------------------------------------------------------------------------------+
+-------------------------------------------------------------------------------+
|db3(master:ONLINE, progress=42, THL latency=0.161)                             |
|STATUS [OK] [2025/01/28 08:34:28 AM UTC]                                       |
+-------------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                        |
|  REPLICATOR(role=master, state=ONLINE)                                        |
|  DATASERVER(state=ONLINE)                                                     |
|  CONNECTIONS(created=6, active=0)                                             |
+-------------------------------------------------------------------------------+
```

If the cluster was in the *MAINTENANCE* policy mode, or nodes do not automatically recover, you can issue the recover command manually followed by returning the cluster to *AUTOMATIC*

- If this does not recover the remaining Replicas within the cluster, these must be manually recovered. This can be achieved either by using Section 6.6.1, “Recover a failed Replica” , or if this is not possible, using Section 6.6.1.2, “Provision or Reprovision a Replica” .

6.6.2.2. Manually Failing over a Primary in *MAINTENANCE* policy mode

If the dataservice is in *MAINTENANCE* mode when the Primary fails, automatic recovery cannot sensibly make the decision about which node should be used as the Primary. In that case, the datasource service must be manually reconfigured.

In the sample below, db2 is the current Primary. The database has failed, but the datasource is still ONLINE, the following steps can be used to force a failover: steps can be used:

```
...
+----------------------------------------------------------------------------+
|db2(master:ONLINE, progress=45, THL latency=0.680)                          |
|STATUS [OK] [2025/01/28 09:08:04 AM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=STOPPED)                                                 |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
...
```

1. Enter expert mode in cctrl. This will suppress the "Do you want to continue?" prompts during the following steps.

```
[LOGICAL] /alpha > expert

WARNING: This is an expert-level command:
Incorrect use may cause data corruption
or make the cluster unavailable.

Do you want to continue? (y/n)> y
[LOGICAL:EXPERT] /alpha >
```

2. We then mark the node as failed using the following command:

```
[LOGICAL:EXPERT] /alpha > datasource db2 fail
DataSource 'db2@alpha' set to FAILED
```

We then see this reflected in the cctrl output:

```
...
+----------------------------------------------------------------------------+
|db2(master:FAILED(MANUALLY-FAILED), progress=51, THL latency=0.272)         |
|STATUS [CRITICAL] [2025/01/28 10:45:41 AM UTC]                              |
|REASON[MANUALLY-FAILED]                                                     |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=STOPPED)                                                 |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
...
```

3. Now we simply issue the failover and let the managers do all the necessary steps to promote a new primary:

```
[LOGICAL:EXPERT] /alpha >> failover
SET POLICY: MAINTENANCE => MAINTENANCE
EVALUATING SLAVE: db1(stored=51, applied=51, latency=0.355, datasource-group-id=0)
EVALUATING SLAVE: db3(stored=51, applied=51, latency=0.358, datasource-group-id=0)
SELECTED SLAVE: db3@alpha
EVALUATING SLAVE: db1(stored=51, applied=51, latency=0.355, datasource-group-id=0)
EVALUATING SLAVE: db3(stored=51, applied=51, latency=0.358, datasource-group-id=0)
SELECTED SLAVE: db3@alpha
Replicator 'db1' is now OFFLINE
THIS IS THE MOST UPTODATE SLAVE. NO ACTION IS NEEDED.
Savepoint failover_3(cluster=alpha, source=db1, created=2025/01/28 10:45:45 UTC) created
PURGE REMAINING ACTIVE SESSIONS ON CURRENT MASTER 'db2@alpha'
WAITING FOR REPLICATOR 'db2' TO REACH STATE DEGRADED
Replicator 'db2' is now in DEGRADED state
SHUNNING PREVIOUS MASTER 'db2@alpha'
PUT THE NEW MASTER 'db3@alpha' ONLINE
FAILOVER TO 'db3' WAS COMPLETED
```

```
...
+----------------------------------------------------------------------------+
|db2(master:SHUNNED(FAILED-OVER-TO-db3), progress=51, THL latency=0.272)     |
|STATUS [SHUNNED] [2025/01/28 10:45:46 AM UTC]                               |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=DEGRADED)                                   |
|  DATASERVER(state=STOPPED)                                                 |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db3(master:ONLINE, progress=53, THL latency=0.432)                          |
|STATUS [OK] [2025/01/28 10:45:50 AM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
```

```
|  DATASERVER(state=ONLINE)                                                    |
|  CONNECTIONS(created=6, active=0)                                            |
+------------------------------------------------------------------------------+
...
```

4. The recovery of the failed node can now be carried out using the methods outlined in Section 6.6.2, “Recover a failed Primary”

6.6.2.3. Split-Brain Discussion

A split-brain occurs when a cluster which normally has a single write Primary, has two write-able Primaries.

This means that some writes which should go to the “real” Primary are sent to a different node which was promoted to write Primary by mistake.

Once that happens, some writes exist on one Primary and not the other, creating two broken Primaries. Merging the two data sets is impossible, leading to a full restore, which is clearly NOT desirable.

We can say that a split-brain scenario is to be strongly avoided.

A situation like this is most often encountered when there is a network partition of some sort, especially with the nodes spread over multiple availability zones in a single region of a cloud deployment.

This would potentially result in all nodes being isolated, without a clear majority within the voting quorum.

A poorly-designed cluster could elect more than one Primary under these conditions, leading to the split-brain scenario.

Since a network partition would potentially result in all nodes being isolated without a clear majority within the voting quorum, the default action of a Tungsten Cluster is to `SHUN` all of the nodes.

Shunning ALL of the nodes means that no client traffic is being processed by any node, both reads and writes are blocked.

When this happens, it is up to a human administrator to select the proper Primary and recover the cluster.

For more information, please see Section 6.6.2.1, “Recover when there are no Primaries”.

6.6.3. Handling Multiple Host Failures

This section outline the procedure for ensuring the cluster can be operational with only one functional host, or "The last man standing"

Typically, when a node fails, the cluster will mark the node as failed and the cluster will continue to operate. If the failed node happend to be the primary then a failover will occur.

If a second node then fails, one of two things will happen.

If the manager process/host are still reachable on the failed node, then datasource will be marked as failed and the cluster will continue to operate.

If the manager process/host are no longer reachable, then the entire cluster goes into a failsafe state. This is intentional and a protection against possible split-brain, since the last remaining manager will not know if it cannot reach the other nodes because of a network partition, or a genuine outage of the host.

If the outage of the nodes is genuine, then you can manually bring up the last remaining node and still have a functional cluster whilst you begin the work of fixing the broken hosts.

The procedure below explains the steps. Note that in this scenario we are dealing with a 3-node cluster which failed in the following way:

- `db1` started as the *`master`* but then failed, `db3` was promoted.
- `db3` failed next.
- Since `db1` and `db3` were unreachable, a failover to `db2` did not happen and instead all nodes where placed into the `SHUNNED(FAILSAFE_SHUN)` state.

This shows the output within cctrl after the multiple host failures:

```
[LOGICAL] /alpha > ls

COORDINATOR[db2:AUTOMATIC:ONLINE]

ROUTERS:
+------------------------------------------------------------------------------+
|connector@db2[7519](ONLINE, created=0, active=0)                              |
+------------------------------------------------------------------------------+

DATASOURCES:
+------------------------------------------------------------------------------+
```

```
|db1(master:SHUNNED(FAILSAFE_SHUN))                                            |
|STATUS [SHUNNED] [2025/06/04 01:51:34 PM UTC]                                  |
+-------------------------------------------------------------------------------+
|  MANAGER(state=STOPPED)                                                       |
|  REPLICATOR(state=STATUS NOT AVAILABLE)                                       |
|  DATASERVER(state=UNKNOWN)                                                    |
|  CONNECTIONS(created=0, active=0)                                             |
+-------------------------------------------------------------------------------+
+-------------------------------------------------------------------------------+
|db2(slave:SHUNNED(FAILSAFE_SHUN), progress=-1, latency=-1.000)                 |
|STATUS [OK] [2025/06/04 01:49:28 PM UTC]                                       |
+-------------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                        |
|  REPLICATOR(role=slave, master=db3, state=OFFLINE)                            |
|  DATASERVER(state=ONLINE)                                                     |
|  CONNECTIONS(created=0, active=0)                                             |
+-------------------------------------------------------------------------------+
+-------------------------------------------------------------------------------+
|db3(master:SHUNNED(FAILSAFE_SHUN))                                             |
|STATUS [OK] [2025/06/04 01:51:38 PM UTC]                                       |
+-------------------------------------------------------------------------------+
|  MANAGER(state=STOPPED)                                                       |
|  REPLICATOR(state=STATUS NOT AVAILABLE)                                       |
|  DATASERVER(state=UNKNOWN)                                                    |
|  CONNECTIONS(created=0, active=0)                                             |
+-------------------------------------------------------------------------------+
```

Since we, as the human operator, knows that `db2` is a viable host, we can then issue the following steps to bring the host online as the primary:

```
[LOGICAL] /alpha > set policy maintenance
[LOGICAL] /alpha > set force true
[LOGICAL] /alpha > datasource db2 welcome
[LOGICAL] /alpha > datasource db2 master
[LOGICAL] /alpha > replicator db2 master
[LOGICAL] /alpha > datasource db2 online
[LOGICAL] /alpha > replicator db2 online
[LOGICAL] /alpha > set policy automatic
```

The cctrl ls output will now look like the following:

```
[LOGICAL] /alpha > ls

COORDINATOR[db2:AUTOMATIC:ONLINE]

ROUTERS:
+-------------------------------------------------------------------------------+
|connector@db2[7519](ONLINE, created=0, active=0)                               |
+-------------------------------------------------------------------------------+

DATASOURCES:
+-------------------------------------------------------------------------------+
|db1(master:SHUNNED(FAILSAFE_SHUN))                                             |
|STATUS [SHUNNED] [2025/06/04 01:51:34 PM UTC]                                  |
+-------------------------------------------------------------------------------+
|  MANAGER(state=STOPPED)                                                       |
|  REPLICATOR(state=STATUS NOT AVAILABLE)                                       |
|  DATASERVER(state=UNKNOWN)                                                    |
|  CONNECTIONS(created=0, active=0)                                             |
+-------------------------------------------------------------------------------+
+-------------------------------------------------------------------------------+
|db2(master:ONLINE, progress=7, THL latency=0.271)                  |
|STATUS [OK] [2025/06/04 01:49:28 PM UTC]                                       |
+-------------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                        |
|  REPLICATOR(role=master, state=ONLINE)                             |
|  DATASERVER(state=ONLINE)                                                     |
|  CONNECTIONS(created=0, active=0)                                             |
+-------------------------------------------------------------------------------+
+-------------------------------------------------------------------------------+
|db3(master:SHUNNED(FAILSAFE_SHUN))                                             |
|STATUS [OK] [2025/06/04 01:51:38 PM UTC]                                       |
+-------------------------------------------------------------------------------+
|  MANAGER(state=STOPPED)                                                       |
|  REPLICATOR(state=STATUS NOT AVAILABLE)                                       |
|  DATASERVER(state=UNKNOWN)                                                    |
|  CONNECTIONS(created=0, active=0)                                             |
+-------------------------------------------------------------------------------+
```

When the failed nodes have been recovered, if the node is viable and does not need any host or database restore, then once all software on the host has been started, you can issue:

```
[LOGICAL] /alpha > recover
```

```
or:

[LOGICAL] /alpha > datasource <host> recover
```

If these steps fail, then you will may need to reprovision the host using tprovision. For more details on this process, see Section 6.6.1, “Recover a failed Replica”

6.7. Composite Cluster Switching, Failover and Recovery

Switching of a dataservice is done to transfer the Active role from one cluster to another, usually in another datacenter site. This also has the effect of turning the original Primary node into a Relay node. The Active dataservice within a composite cluster can be forced to failover to the Passive dataservice in the event the Active dataservice is offline.

Switching the Active dataservice performs the following steps:

1. Set the Primary node to offline state. New connections to the Primary are rejected, and writes to the Primary are stopped.
2. On the relay in the target cluster, switch the datasource offline. New connections are rejected, stopping reads on this Primary.
3. Kill any outstanding client connections to the Primary data source, except those belonging to the `tungsten` account.
4. Send a heartbeat transaction between the old Primary and the new Primary, and wait until this transaction has been received. Once received, the THL on Primary and Replica are up to date.
5. Perform the switch:
   - Configure all remaining replicators offline
   - Configure the target cluster relay node as the new Primary.
   - Set the new Primary to the online state.
   - New connections to the Primary are permitted.
6. Configure the old Primary to be a relay datasource.
7. Configure the Replicas in the primary site to use the new Primary datasource.
8. Configure the Replicas in the Replica site to use the new relay datasource.
9. Update the connector configurations and enable client connections to connect to the Primaries and Replicas.

The switching process is monitored by Tungsten Cluster, and if the process fails, either due to a timeout or a recoverable error occurs, the switch operation is rolled back, returning the dataservice to the original configuration. This ensures that the dataservice remains operational. In some circumstances, when performing a manual switch, the command may need to be repeated to ensure the requested switch operation completes.

The process takes a finite amount of time to complete, and the exact timing and duration will depend on the state, health, and database activity on the dataservice. The actual time taken will depend on how up to date the Replica being promoted is compared to the Primary. The switch will take place regardless of the current status after a delay period.

6.7.1. Composite Cluster Site Switch

> **Note**
>
> These steps only apply to Composite Active/Passive clusters.

Our example cluster has two sites, `alpha` and `beta`. They are both members of composite cluster `global`. Site east has hosts db1, db2 and db3. Site west has hosts db4, db5 and db6.

> **Important**
>
> When working with composite clusters, using cctrl will by default connect you to the cluster service associated with the node in which you launched the tool, however you can easily move around to all clusters by simply using the use command as show in the examples below.

```
shell> cctrl
Tungsten Cluster 7.0.3 build 141
alpha: session established, encryption=false, authentication=false
jgroups: unencrypted, database: unencrypted
```

```
[LOGICAL] /alpha > use global
[LOGICAL] /global > ls

COORDINATOR[db2:AUTOMATIC:ONLINE]
   alpha:COORDINATOR[db2:AUTOMATIC:ONLINE]
   beta:COORDINATOR[db5:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@db1[43475](ONLINE, created=2, active=0)                           |
|connector@db2[75463](ONLINE, created=2, active=0)                           |
|connector@db3[43981](ONLINE, created=2, active=0)                           |
|connector@db4[59575](ONLINE, created=2, active=0)                           |
|connector@db5[59581](ONLINE, created=2, active=0)                           |
|connector@db6[59847](ONLINE, created=2, active=0)                           |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|alpha(composite master:ONLINE)                                              |
|STATUS [OK] [2025/01/28 11:35:18 AM UTC]                                    |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|beta(composite slave:ONLINE)                                                |
|STATUS [OK] [2025/01/28 11:35:13 AM UTC]                                    |
+----------------------------------------------------------------------------+
```

Composite Active Dataservice (Primary) - `alpha`

```
[LOGICAL] /global > use alpha
[LOGICAL] /alpha > ls

COORDINATOR[db2:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@db1[43475](ONLINE, created=0, active=0)                           |
|connector@db2[75463](ONLINE, created=0, active=0)                           |
|connector@db3[43981](ONLINE, created=0, active=0)                           |
|connector@db4[59575](ONLINE, created=0, active=0)                           |
|connector@db5[59581](ONLINE, created=0, active=0)                           |
|connector@db6[59847](ONLINE, created=0, active=0)                           |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|db1(master:ONLINE, progress=0, THL latency=1.061)                           |
|STATUS [OK] [2025/01/28 11:35:02 AM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db2(slave:ONLINE, progress=0, latency=1.691)                                |
|STATUS [OK] [2025/01/28 11:35:00 AM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db3(slave:ONLINE, progress=0, latency=1.382)                                |
|STATUS [OK] [2025/01/28 11:35:02 AM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
```

Composite Passive Dataservice (DR) - `beta`

```
[LOGICAL] /alpha]ha > use beta
[LOGICAL] /beta > ls

COORDINATOR[db5:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@db1[43475](ONLINE, created=2, active=0)                           |
```

```
|connector@db2[75463](ONLINE, created=2, active=0)                              |
|connector@db3[43981](ONLINE, created=2, active=0)                              |
|connector@db4[59575](ONLINE, created=2, active=0)                              |
|connector@db5[59581](ONLINE, created=2, active=0)                              |
|connector@db6[59847](ONLINE, created=2, active=0)                              |
+-------------------------------------------------------------------------------+

DATASOURCES:
+-------------------------------------------------------------------------------+
|db4(relay:ONLINE, progress=0, latency=1.000)                                   |
|STATUS [OK] [2025/01/28 11:35:02 AM UTC]                                       |
+-------------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                        |
|  REPLICATOR(role=relay, master=db1, state=ONLINE)                             |
|  DATASERVER(state=ONLINE)                                                     |
|  CONNECTIONS(created=0, active=0)                                             |
+-------------------------------------------------------------------------------+
+-------------------------------------------------------------------------------+
|db5(slave:ONLINE, progress=0, latency=1.515)                                   |
|STATUS [OK] [2025/01/28 11:35:12 AM UTC]                                       |
+-------------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                        |
|  REPLICATOR(role=slave, master=db4, state=ONLINE)                             |
|  DATASERVER(state=ONLINE)                                                     |
|  CONNECTIONS(created=6, active=0)                                             |
+-------------------------------------------------------------------------------+
+-------------------------------------------------------------------------------+
|db6(slave:ONLINE, progress=0, latency=1.592)                                   |
|STATUS [OK] [2025/01/28 11:35:12 AM UTC]                                       |
+-------------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                        |
|  REPLICATOR(role=slave, master=db4, state=ONLINE)                             |
|  DATASERVER(state=ONLINE)                                                     |
|  CONNECTIONS(created=6, active=0)                                             |
+-------------------------------------------------------------------------------+
```

Manually switch the composite Primary role to the other site:

```
[LOGICAL] /beta > use global
[LOGICAL] /global > switch

Savepoint switch_0(cluster=global, source=db2, created=2025/01/28 13:53:00 UTC) created
SELECTED SLAVE: 'beta@global'
FLUSHING TRANSACTIONS THROUGH 'db1@alpha'
REPLICATOR 'db1' IS NOW USING MASTER CONNECT URI 'thl://db4:2112/'
composite data source 'beta@global' is now OFFLINE
PUT THE NEW MASTER 'beta@global' ONLINE
PUT THE PRIOR MASTER 'alpha@global' ONLINE AS A SLAVE
REVERT POLICY: MAINTENANCE => AUTOMATIC
SWITCH TO 'beta@global' WAS SUCCESSFUL

[LOGICAL] /global > ls

COORDINATOR[db2:AUTOMATIC:ONLINE]
   alpha:COORDINATOR[db2:AUTOMATIC:ONLINE]
   beta:COORDINATOR[db5:AUTOMATIC:ONLINE]

ROUTERS:
+-------------------------------------------------------------------------------+
|connector@db1[43475](ONLINE, created=2, active=0)                              |
|connector@db2[75463](ONLINE, created=2, active=0)                              |
|connector@db3[43981](ONLINE, created=2, active=0)                              |
|connector@db4[59575](ONLINE, created=2, active=0)                              |
|connector@db5[59581](ONLINE, created=2, active=0)                              |
|connector@db6[59847](ONLINE, created=2, active=0)                              |
+-------------------------------------------------------------------------------+

DATASOURCES:
+-------------------------------------------------------------------------------+
|alpha(composite slave:ONLINE)                                                  |
|STATUS [OK] [2025/01/28 01:53:13 PM UTC]                                       |
+-------------------------------------------------------------------------------+
+-------------------------------------------------------------------------------+
|beta(composite master:ONLINE)                                                  |
|STATUS [OK] [2025/01/28 01:53:13 PM UTC]                                       |
+-------------------------------------------------------------------------------+
```

6.7.2. Composite Cluster Site Failover (Forced Switch)

In the event the Active site goes down, and a graceful manual switch is not possible, the composite Active role can be failed over to the Passive cluster using cctrl. The failover command performs the forced switch operation. It will try to update the configuration of the east data service but will not fail if not successful.

In this example, hosts db4 (the composite Primary), db5 and db6 in cluster beta have been shut down. To force dataservice alpha to become the primary, login to a node in that cluster and get into cctrl:

```
shell> cctrl -multi
Tungsten Cluster 7.0.3 build 141
west: session established
[LOGICAL] / > use global
[LOGICAL] /global > ls

COORDINATOR[db2:AUTOMATIC:ONLINE]
   alpha:COORDINATOR[db2:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@db1[43475](ONLINE, created=2, active=0)                           |
|connector@db2[75463](ONLINE, created=2, active=0)                           |
|connector@db3[43981](ONLINE, created=2, active=0)                           |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|alpha(composite slave:ONLINE)                                               |
|STATUS [OK] [2025/01/28 01:53:13 PM UTC]                                    |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|beta(composite master:SHUNNED(FAILSAFE_SHUN))                               |
|STATUS [SHUNNED] [2025/01/28 02:21:56 PM UTC]                               |
+----------------------------------------------------------------------------+
```

Mark the beta data service as failed to prevent further actions:

```
[LOGICAL] /global > datasource beta fail

WARNING: This is an expert-level command:
Incorrect use may cause data corruption
or make the cluster unavailable.

Do you want to continue? (y/n)> y
COMPOSITE DATA SOURCE 'beta' IS NOW IN THE FAILED STATE

[LOGICAL] /global > ls

COORDINATOR[db2:AUTOMATIC:ONLINE]
   alpha:COORDINATOR[db2:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@db1[43475](ONLINE, created=2, active=0)                           |
|connector@db2[75463](ONLINE, created=2, active=0)                           |
|connector@db3[43981](ONLINE, created=2, active=0)                           |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|alpha(composite slave:ONLINE)                                               |
|STATUS [OK] [2025/01/28 01:53:13 PM UTC]                                    |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|beta(composite master:FAILED(MANUALLY-FAILED))                              |
|STATUS [CRITICAL] [2025/01/28 02:29:53 PM UTC]                              |
|REASON[MANUALLY-FAILED]                                                     |
+----------------------------------------------------------------------------+
```

Issue the failover command to force the alpha dataservice to become the composite Primary:

```
[LOGICAL] /global > failover
WARNING: DATA SERVICE 'beta' IS NOT AVAILABLE. CANNOT GET STATE
EXCEPTION=Unable to continue with command because no manager is available in service 'beta'.

Savepoint failover_1(cluster=global, source=db2, created=2025/01/28 14:35:37 UTC) created
SELECTED SLAVE: 'alpha@global'
ENSURING THAT WE CATCH UP WITH THE MOST ADVANCED RELAY
composite data source 'alpha@global' is now OFFLINE
PUT THE NEW MASTER 'alpha@global' ONLINE
REVERT POLICY: MAINTENANCE => AUTOMATIC
FAILOVER TO 'alpha@global' WAS SUCCESSFUL

[LOGICAL] /global > ls

COORDINATOR[db2:AUTOMATIC:ONLINE]
   alpha:COORDINATOR[db2:AUTOMATIC:ONLINE]

ROUTERS:
```

```
+-------------------------------------------------------------------------------+
|connector@db1[43475](ONLINE, created=2, active=0)                              |
|connector@db2[75463](ONLINE, created=2, active=0)                              |
|connector@db3[43981](ONLINE, created=2, active=0)                              |
+-------------------------------------------------------------------------------+

DATASOURCES:
+-------------------------------------------------------------------------------+
|alpha(composite master:ONLINE)                                                 |
|STATUS [OK] [2025/01/28 02:36:15 PM UTC]                                       |
+-------------------------------------------------------------------------------+
+-------------------------------------------------------------------------------+
|beta(composite master:SHUNNED(MANUAL-FAILOVER))                                |
|STATUS [SHUNNED] [2025/01/28 02:35:46 PM UTC]                                  |
+-------------------------------------------------------------------------------+
```

6.7.3. Composite Cluster Site Recovery

When the site that was lost is returned to operation and all tungsten services have been restarted, if at all possible, the cluster will attempt automatic recovery ensuring the cluster is returned as a slave dataservice and all nodes online. For the automatic recovery to be attempted, the clusters must be in the *AUTOMATIC* policy mode.

If the nodes cannot be recovered, the first step in recovering the SHUNNED dataservice is to re-provision the nodes if the data has gotten out of sync. See Section 6.6.1.2, “Provision or Reprovision a Replica” for more information.

Once the failed site has been restored, the shunned/superseded dataservice can be brought back online using cctrl . The recover command performs this operation, annotating the progress.

```
...
DATASOURCES:
+-------------------------------------------------------------------------------+
|alpha(composite master:ONLINE)                                                 |
|STATUS [OK] [2025/01/28 02:36:15 PM UTC]                                       |
+-------------------------------------------------------------------------------+
+-------------------------------------------------------------------------------+
|beta(composite master:SHUNNED(SUPERSEDED))                                     |
|STATUS [SHUNNED] [2025/01/28 02:39:41 PM UTC]                                  |
+-------------------------------------------------------------------------------+
...
```

Use the recover to bring the SHUNNED dataservice back online as a composite Replica:

```
[LOGICAL] /global > recover
IDENTIFIED DATASOURCE 'beta@global' FOR RECOVERY
COULD NOT IDENTIFY ACTIVE PRIMARY FOR SERVICE 'beta'
ATTEMPTING TO IDENTIFY A FAILED PRIMARY FOR 'beta'
PHYSICAL DATA SERVICE 'beta' DOES NOT HAVE AN ACTIVE RELAY
FORCING THE PHYSICAL RELAY TO BE 'db4'
DATASOURCE 'db4@beta' IS NOW A RELAY
RECOVERED 2 DATA SOURCES IN SERVICE 'beta'
composite data source 'beta@global' role is now slave
composite data source 'beta' is now OFFLINE
REVERT SET POLICY AUTOMATIC
RECOVERY OF COMPOSITE SERVICE 'global' IS COMPLETE

[LOGICAL] /global > ls

COORDINATOR[db2:AUTOMATIC:ONLINE]
   alpha:COORDINATOR[db2:AUTOMATIC:ONLINE]
   beta:COORDINATOR[db4:AUTOMATIC:ONLINE]

ROUTERS:
+-------------------------------------------------------------------------------+
|connector@db1[43475](ONLINE, created=2, active=0)                              |
|connector@db2[75463](ONLINE, created=2, active=0)                              |
|connector@db3[43981](ONLINE, created=2, active=0)                              |
|connector@db4[2062](ONLINE, created=2, active=0)                               |
|connector@db5[2079](ONLINE, created=2, active=0)                               |
|connector@db6[2080](ONLINE, created=2, active=0)                               |
+-------------------------------------------------------------------------------+

DATASOURCES:
+-------------------------------------------------------------------------------+
|alpha(composite master:ONLINE)                                                 |
|STATUS [OK] [2025/01/28 02:36:15 PM UTC]                                       |
+-------------------------------------------------------------------------------+
+-------------------------------------------------------------------------------+
|beta(composite slave:ONLINE)                                                   |
|STATUS [OK] [2025/01/28 02:40:06 PM UTC]                                       |
+-------------------------------------------------------------------------------+
```

6.7.4. Composite Cluster Relay Recovery

If the Relay node in a Composite cluster should ever point to the incorrect Primary node, you can perform the following procedure to re-point the replicator to the desired Primary node.

For example, say we have a composite cluster `global`, with nodes db1, db2 and db3 in `alpha` and db4, db5 and db6 in `beta`. db1 is the Primary and db4 is the relay.

In the output below, the relay node db4 shows that its replicator is using db2 as the Primary instead of db1:

```
+----------------------------------------------------------------------------+
|db4(relay:ONLINE, progress=5, latency=0.352)                                |
|STATUS [OK] [2025/01/28 02:39:49 PM UTC]                                    |
+----------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                      |
| REPLICATOR(role=relay, master=db2, state=ONLINE)                           |
| DATASERVER(state=ONLINE)                                                   |
| CONNECTIONS(created=8108, active=0)                                        |
+----------------------------------------------------------------------------+
```

Use the cctrl replicator command to adjust the relay source:

```
shell> cctrl
Tungsten Cluster 7.0.3 build 141
west: session established

[LOGICAL] /beta > set policy maintenance
[LOGICAL] /beta > replicator db4 offline
[LOGICAL] /beta > replicator db4 relay alpha/db1
[LOGICAL] /beta > set policy automatic
[LOGICAL] /beta > ls
+----------------------------------------------------------------------------+
|db4(relay:ONLINE, progress=5, latency=0.352)                                |
|STATUS [OK] [2025/01/28 02:39:49 PM UTC]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=relay, master=db1, state=ONLINE)                          |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
```

6.8. Composite Active/Active Recovery

In an Composite Active/Active topology, a switch or a failover not only promotes a Replica to be a new Primary, but also will require the ability to reconfigure cross-site communications. This process therefore assumes that cross-site communication is online and working. In some situations, it may be possible that cross-site communication is down, or for some reason cross-site replication is in an `OFFLINE:ERROR` state - for example a DDL or DML statement that worked in the local cluster may have failed to apply in the remote cluster.

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

The guide explains how to recover from this situation.

The examples are based on a 2-cluster topology, named `nyc` and `london` and the composite dataservice named `global`. The cluster is configured with the following dataservers:

- `nyc` : `db1` (Primary), `db2` (Replica), `db3` (Replica)
- `nyc`: `db4` (Primary), `db5` (Replica), `db6` (Replica)

The cross site replicators in both clusters are in an `OFFLINE:ERROR` state due to failing DDL.

A switch was then issued, promoting `db3` as the new Primary in `nyc` and `db5` as the new Primary in `london`

At the top-level, the switch was a success, however, in the cctrl output below, you can see that all dataources in the `london_from_nyc` sub-service are in the `SHUNNED(SUBSERVICE-SWITCH-FAILED)` state, and partial reconfiguration has happened. The same can also be observed in the `nyc_from_london` sub-service

```
shell> cctrl
Tungsten Cluster 7.0.3 build 141
nyc: session established
[LOGICAL:EXPERT] / > use london_from_nyc
london_from_nyc: session established, encryption=false, authentication=false
[LOGICAL:EXPERT] /london_from_nyc > ls
```

```
COORDINATOR[db6:AUTOMATIC:ONLINE]

DATASOURCES:
+----------------------------------------------------------------------------+
|db4(relay:SHUNNED(SUBSERVICE-SWITCH-FAILED), progress=-1, latency=-1.000)   |
|STATUS [SHUNNED] [2025/01/29 11:55:42 AM UTC]                               |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=relay, master=db3, state=SUSPECT)                         |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db5(slave:SHUNNED(SUBSERVICE-SWITCH-FAILED), progress=-1, latency=-1.000)   |
|STATUS [SHUNNED] [2025/01/29 11:55:42 AM UTC]                               |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db4, state=SUSPECT)                         |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
+----------------------------------------------------------------------------+
|db6(slave:SHUNNED(SUBSERVICE-SWITCH-FAILED), progress=-1, latency=-1.000)   |
|STATUS [SHUNNED] [2025/01/29 11:55:42 AM UTC]                               |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db4, state=SUSPECT)                         |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
```

The role of `db5` should `relay` and `db4` should be `slave`.

The replicators for `db4` and `db6` should be Replicas of `db5`.

`db5` has correctly configured to the new Primary in `london` and `db3` in `nyc`.

The actual state of the cluster in each scenario maybe different depending upon the cause of the loss of cross-site communication. Using the steps below, apply the necessary actions that relate to your own cluster state, if in any doubt always contact Continuent Support for assistance.

1. The first step is to ensure the initial replication errors have been resolved and that the replicators are in an online state, the steps to resolve the replicators will depend on the reason for the error.

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

   ```
   shell> cctrl -expert
   ```

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

   ```
   [LOGICAL:EXPERT] / > use london_from_nyc
   london_from_nyc: session established, encryption=false, authentication=false
   ```

4. Enable override of commands issued

   ```
   [LOGICAL:EXPERT] /london_from_nyc > set force true
   ```

5. Bring the datasources online:

   ```
   [LOGICAL:EXPERT] /london_from_nyc > datasource db4 online
   [LOGICAL:EXPERT] /london_from_nyc > datasource db5 online
   [LOGICAL:EXPERT] /london_from_nyc > datasource db6 online
   ```

6. Issue the switch to to force the cluster to move the relay role to the correct node:

   ```
   [LOGICAL:EXPERT] /london_from_nyc > switch to db5

   WARNING: This is an expert-level command:
   Switch in a subservice will make the RELAY different from MASTER.
   This may cause data corruption or make the cluster unavailable.

   Do you want to continue? (y/n)> y
   SET POLICY: MAINTENANCE => MAINTENANCE
   EVALUATING SLAVE: db5(stored=12, applied=12, latency=257.083, datasource-group-id=0)
   SELECTED SLAVE: db5@london_from_nyc
   Savepoint switch_1(cluster=london_from_nyc, source=db4, created=2025/01/29 13:33:32 UTC) created
   PRIMARY IS REMOTE. USING 'thl://db3:2112/' for the MASTER URI
   PUT THE NEW RELAY 'db5@london_from_nyc' ONLINE
   PUT THE PRIOR RELAY 'db4@london_from_nyc' ONLINE AS A SLAVE
   SWITCH TO 'db5@london_from_nyc' WAS SUCCESSFUL
   ```

Note

In this example, cctrl warns about switching the relay. In normal operation this should be noted since the `relay` and `master` roles should be the same host. In this case we can ignore the warning since we are intentioanlly moving the `relay` to satisfy the condition that we are being warned to avoid!

In some situations the `relay` role may have switched, but the replicators are in a mixed state and not configured correctly, for example, `db5` may be showing correctly as the new `relay` however the replicator may not have updated and may be configured to point to the old `master` in the `nyc` cluster (`db1` instead of `db3`).

In that situation, the following steps should be followed:

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

```
shell> cctrl -expert
```

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

```
[LOGICAL:EXPERT] / > use london_from_nyc
london_from_nyc: session established, encryption=false, authentication=false
```

3. Enable override of commands issued

```
[LOGICAL:EXPERT] /london_from_nyc > set force true
```

4. Bring the relay datasource online:

```
[LOGICAL:EXPERT] /london_from_nyc > datasource db5 online
```

5. Next, place the service into *MAINTENANCE* Mode

```
[LOGICAL:EXPERT] /london_from_nyc > set policy maintenance
```

6. If you need to change the source for the relay replicator to the correct, new, Primary in the remote cluster, take the replicator offline. If the relay source is correct, then move on to step 9

```
[LOGICAL:EXPERT] /london_from_nyc > replicator db5 offline
```

7. Change the source of the relay replicator

```
[LOGICAL:EXPERT] /london_from_nyc > replicator db5 relay nyc/db3
```

8. Bring the relay replicator online

```
[LOGICAL:EXPERT] /london_from_nyc > replicator db5 online
```

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

```
[LOGICAL:EXPERT] /london_from_nyc > replicator {datasource} offline
[LOGICAL:EXPERT] /london_from_nyc > replicator {datasource} slave {relay}
[LOGICAL:EXPERT] /london_from_nyc > replicator {datasource} online
```

   For example:

```
[LOGICAL:EXPERT] /london_from_nyc > replicator db4 offline
[LOGICAL:EXPERT] /london_from_nyc > replicator db4 slave db5
[LOGICAL:EXPERT] /london_from_nyc > replicator db4 online
```

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

```
[LOGICAL:EXPERT] /london_from_nyc > cluster welcome
```

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

```
[LOGICAL:EXPERT] /london_from_nyc > datasource {datasource} online
```

   For example:

```
[LOGICAL:EXPERT] /london_from_nyc > datasource db4 online
```

12. Once all nodes are online, we can then return the cluster to *AUTOMATIC*

```
[LOGICAL:EXPERT] /london_from_nyc > set policy automatic
```

Whichever process you followed above, you would now repeat on the other cluster if required.

6.9. Managing Transaction Failures

Inconsistencies between a Primary and Replica dataserver can occur for a number of reasons, including:

- An update or insertion has occurred on the Replica independently of the Primary. This situation can occur if updates are allowed on a Replica that is acting as a read-only Replica for scale out, or in the event of running management or administration scripts on the Replica
- A switch or failover operation has lead to inconsistencies. This can happen if client applications are still writing to the Replica or Primary at the point of the switch.
- A database failure causes a database or table to become corrupted.

When a failure to apply transactions occurs, the problem must be resolved, either by skipping or ignoring the transaction, or fixing and updating the underlying database so that the transaction can be applied.

When a failure occurs, replication is stopped immediately at the first transaction that caused the problem, but it may not be the only transaction and this may require extensive examination of the pending transactions to determine what caused the original database failure and then to fix and address the error and restart replication.

6.9.1. Identifying a Transaction Mismatch

When a mismatch occurs, the replicator service will indicate that there was a problem applying a transaction on the Replica. The replication process stops applying changes to the Replica when the first transaction fails to be applied to the Replica. This prevents multiple-statements from failing

Within cctrl the status of the datasource will be marked as FAILED, and the replicator state as SUSPECT [145]:

```
LOGICAL] /alpha > ls

COORDINATOR[db2:AUTOMATIC:ONLINE]
...
+----------------------------------------------------------------------------+
|db2(slave:FAILED(REPLICATOR IS IN THE 'SUSPECT' STATE), progress=-1,        |
|latency=-1.000)                                                             |
|STATUS [CRITICAL] [2025/01/31 10:54:26 AM UTC]                              |
|REASON[REPLICATOR IS IN THE 'SUSPECT' STATE]                                |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=db3, state=SUSPECT)                         |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=2, active=0)                                          |
+----------------------------------------------------------------------------+
...
```

More detailed information about the status and the statement that failed can be obtained in a number of ways, either within cctrl using the replicator command:

```
[LOGICAL] /nyc > replicator db2 status
+----------------------------------------------------------------------------+
|replicator db2 status                                                       |
+----------------------------------------------------------------------------+
|            appliedLastEventId:    NONE                                     |
|        appliedLastGoodLatency:    -1.0                                     |
|          appliedLastGoodSeqno:    -1                                       |
|              appliedLastSeqno:    -1                                       |
|                appliedLatency:    -1.0                                     |
|           autoRecoveryEnabled:    false                                    |
|             autoRecoveryTotal:    0                                        |
|                     channels:    -1                                       |
|                  clusterName:    nyc                                      |
|               currentEventId:    NONE                                     |
|             currentTimeMillis:    1738320971988                            |
|                dataServerHost:    db2                                      |
|                    extensions:                                             |
|                          host:    db2                                      |
|             latestEpochNumber:    -1                                       |
|              masterConnectUri:    thl://db3:2112/                          |
|               masterListenUri:    thl://db2:2112/                          |
|           maximumStoredSeqNo:    -1                                        |
|           minimumStoredSeqNo:    -1                                        |
|              offlineRequests:    NONE                                     |
|pendingError:    Event application failed: seqno=18 fragno=0                |
|message=java.sql.SQLException: Statement failed on slave but succeeded on   |
|master                                                                      |
|             pendingErrorCode:    NONE                                     |
|          pendingErrorEventId:    mysql-bin.000003:0000000000001389;2010   |
|            pendingErrorSeqno:    18                                       |
```

```
|pendingExceptionMessage:    java.sql.SQLException: Statement failed on slave     |
|but succeeded on master                                                          |
|insert into regions values (1, "EMEA")                                           |
|               pipelineSource:    UNKNOWN                                        |
|             relativeLatency:    -1.0                                           |
|           resourceJdbcDriver:    org.drizzle.jdbc.DrizzleDriver                 |
|resourceJdbcUrl:                                                                 |
|jdbc:mysql:thin://db2:13306/${DBNAME}?jdbcCompliantTruncation=false&zeroDateT    |
|imeBehavior=convertToNull&tinyInt1isBit=false&allowMultiQueries=true&yearIsDa    |
|teType=false                                                                     |
|               resourceVendor:    mysql                                          |
|                     rmiPort:    10000                                          |
|                        role:    slave                                          |
|                    seqnoType:    java.lang.Long                                 |
|                  serviceName:    nyc                                            |
|                  serviceType:    unknown                                        |
|            simpleServiceName:    nyc                                            |
|                     siteName:    default                                        |
|                     sourceId:    db2                                            |
|                        state:    OFFLINE:ERROR                                  |
|           timeInStateSeconds:    114.834                                        |
|                     timezone:    GMT                                            |
|              transitioningTo:                                                   |
|                uptimeSeconds:    1811.377                                       |
|             useSSLConnection:    false                                          |
|                      version:    Tungsten Clustering 7.0.3 build 141            |
+---------------------------------------------------------------------------------+
```

By checking the replication status with trepctl, the *pendingError* and *pendingExceptionMessage* will show the error indicating the failure to insert the statement. For example:

```
shell> trepctl status
...
pendingError           : Event application failed: seqno=18 fragno=0 message=java.sql.SQLException: Statement failed on slave but succeeded on master
pendingErrorCode       : NONE
pendingErrorEventId    : mysql-bin.000003:0000000000001389;2010
pendingErrorSeqno      : 18
pendingExceptionMessage: java.sql.SQLException: Statement failed on slave but succeeded on master
                         insert into regions values (1, "EMEA")
...
```

The `trepsvc.log` log file will also contain the error information about the failed statement. For example:

```
...
2025/01/31 10:54:17.194 | com.continuent.tungsten.replicator.applier.ApplierException: java.sql.SQLException: Statement failed on slave but succeeded on m
2025/01/31 10:54:17.194 |       at com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier.applyStatementData(MySQLDrizzleApplier.java:162) ~[tungs
2025/01/31 10:54:17.194 |       at com.continuent.tungsten.replicator.applier.JdbcApplier.apply(JdbcApplier.java:1789) ~[tungsten-replicator.jar:?]
2025/01/31 10:54:17.194 |       at com.continuent.tungsten.replicator.applier.ApplierWrapper.apply(ApplierWrapper.java:101) ~[tungsten-replicator.jar:?]
2025/01/31 10:54:17.194 |       at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.apply(SingleThreadStageTask.java:1032) [tungsten-repl
2025/01/31 10:54:17.194 |       at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.runTask(SingleThreadStageTask.java:634) [tungsten-rep
2025/01/31 10:54:17.195 |       at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.run(SingleThreadStageTask.java:207) [tungsten-replica
2025/01/31 10:54:17.195 |       at java.lang.Thread.run(Thread.java:750) [?:1.8.0_432]
2025/01/31 10:54:17.195 | Caused by: java.sql.SQLException: Statement failed on slave but succeeded on master
2025/01/31 10:54:17.195 |       at com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier.applyStatementData(MySQLDrizzleApplier.java:124) ~[tungs
2025/01/31 10:54:17.195 |       ... 6 more
2025/01/31 10:54:17.195 | Caused by: java.sql.SQLIntegrityConstraintViolationException: Duplicate entry '1' for key 'regions.PRIMARY'
2025/01/31 10:54:17.195 |       at org.drizzle.jdbc.internal.SQLExceptionMapper.get(SQLExceptionMapper.java:92) ~[drizzle-jdbc-1.5.20-SNAPSHOT.jar:?]
2025/01/31 10:54:17.195 |       at org.drizzle.jdbc.DrizzleStatement.executeBatch(DrizzleStatement.java:951) ~[drizzle-jdbc-1.5.20-SNAPSHOT.jar:?]
2025/01/31 10:54:17.195 |       at com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier.applyStatementData(MySQLDrizzleApplier.java:110) ~[tungs
2025/01/31 10:54:17.195 |       ... 6 more
```

Once the error or problem has been found, the exact nature of the error should be determined so that a resolution can be identified:

1. Identify the reason for the failure by examining the full error message. Common causes are:

   - Duplicate primary key

     A row or statement is being inserted or updated that already has the same insert ID or would generate the same insert ID for tables that have auto increment enabled. The insert ID can be identified from the output of the transaction using thl. Check the Replica to identify the faulty row. To correct this problem you will either need to skip the transaction or delete the offending row from the Replica dataserver.

     In the above example, the error can be identified through the methods described. The log file output will also show the error direct from the database, and in this case that error is `Duplicate entry '1' for key 'regions.PRIMARY'`

     The error can be generated when an insert or update has taken place on the Replica rather than on the Primary.

     This can also be a common error is misconfigured Composite Active/Active clusters where generated sequence offsets have not been set correctly or where the application is not active/active aware and writes happen in both clusters using the same application generated ID causing PK violations.

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. The sequence number of the statement causing error can be found in the *pendingErrorSeqno* field. In our example, this is 18. For example, to view the THL for the sequence number:

```
shell> thl list -seqno 18
SEQ# = 18 / FRAG# = 0 (last frag)
- FILE = thl.data.0000000001
- TIME = 2025-01-31 10:53:33.0
- EPOCH# = 15
- EVENTID = mysql-bin.000003:0000000000001389;2010
- SOURCEID = db3
- METADATA = [mysql_server_id=12;mysql_thread_id=2010;dbms_type=mysql;tz_aware=true;service=nyc;shard=hr]
- 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 = 'NO_ENGINE_SUBSTITUTION,ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,IGNORE_SPACE,»
NO_ZERO_DATE,NO_ZERO_IN_DATE', character_set_client = 33, collation_connection = 33, collation_server = 255]
- SCHEMA = hr
- SQL(0) = insert into regions values (1, "EMEA")
```

In this example, an `INSERT` operation is inserting a new row and the ID has been explicity defined (1) Check the destination database and determine the what the current value of the corresponding row:

```
mysql> select * from regions where region_id = 1;
+-----------+-------------+
| region_id | region_name |
+-----------+-------------+
|         1 | EMEA        |
+-----------+-------------+
1 row in set (0.00 sec)
```

The actual row values are identicial, which means that it is likely this row was inserted into the replica database directly by mistake.

- 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'
  ```

- Incompatible table or schema

  A modified table structure on the Replica can cause application of the transaction to fail if there are missing or different column specifications for the table data.

  This particular error can be generated when changes to the table definition have been made, perhaps during a maintenance window.

  Check the table definition on the Primary and Replica and ensure they match.

2. Choose a resolution method:

   Depending on the data structure and environment, resolution can take one of the following forms:

   - Skip the transaction on the Replica

     If the data on the Replica is considered correct, or the data in both tables is the same or similar, the transaction from the Primary to the Replica can be skipped. This process involves placing the replicator online and specifying one or more transactions to be skipped or ignored. At the end of this process, the replicator should be in the `ONLINE` state.

     For more information on skipping single or multiple transactions, see Section 6.9.2, “Skipping Transactions”.

   - Delete the offending row or rows on the Replica

     If the data on the Primary is considered canonical, then the data on the Replica can be removed, and the replicator placed online.

     > **Warning**
     >
     > Deleting data on the Replica may cause additional problems if the data is used by other areas of your application, or has relations to foreign tables.

     For example:

     ```
     mysql> delete from regions where region_id = 1;
     Query OK, 1 row affected (0.01 sec)
     ```

     Now place the replicator online and check the status:

or, within cctrl:

```
[LOGICAL] /alpha > replicator host3 online
```

- Restore or reprovision the Replica

  If the transaction cannot be skipped, or the data safely deleted or modified, and only a single Replica is affected, a backup of an existing, working, Replica can be taken and restored to the broken Replica.

  The tprovision command automates this process. See Section 6.6.1.2, "Provision or Reprovision a Replica" for more information on reprovisioning.

  To perform a backup and restore, see Section 6.10, "Creating a Backup", or Section 6.11, "Restoring a Backup". To reprovision a Replica from the Primary or another Replica, see tprovision.

6.9.2. Skipping Transactions

When a failure caused by a mismatch or failure to apply one or more transactions, the transaction(s) can be skipped. Transactions can either be skipped one at a time, through a specific range, or a list of single and range specifications.

> **Warning**
>
> Skipping over events can easily lead to Replica inconsistencies and later replication errors. Care should be taken to ensure that the transaction(s) can be safely skipped without causing problems. See Section 6.9.1, "Identifying a Transaction Mismatch".

- Skipping a Single Transaction

  If the error was caused by only a single statement or transaction, the transaction can be skipped using trepctl online:

  ```
  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.

6.10. Creating a Backup

The datasource backup command for a datasource within cctrl backs up a datasource using the default backup tool. During installation, xtrabackup-full will be used if xtrabackup has been installed. Otherwise, the default backup tool used is mysqldump .

> **Important**
>
> For consistency, all backups should include a copy of all `tungsten_SERVICE` schemas. This ensures that when the Tungsten Replicator service is restarted, the correct start points for restarting replication are recorded with the corresponding backup data. Failure to include the `tungsten_SERVICE` schemas may prevent replication from being restart effectively.

Backing up a datasource can occur while the replicator is online:

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

```
Using the 'xtrabackup-full' backup agent.
Replicator 'db2' starting backup
Backup of dataSource 'db2' succeeded; uri=storage://file-system/store-0000000001.properties
```

or:

```
shell> trepctl backup
Backup completed successfully; URI=storage://file-system/store-0000000003.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 12
-rw-r--r--. 1 tungsten tungsten  71 Jan 31 11:34 storage.index
-rw-r--r--. 1 tungsten tungsten  70 Jan 31 11:34 store-0000000001-full_xtrabackup_2025-01-31_11-34_35
-rw-r--r--. 1 tungsten tungsten 336 Jan 31 11:34 store-0000000001.properties
drwxr-xr-x. 3 tungsten tungsten  49 Jan 31 11:34 xtrabackup
```

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 (for mysqldump) or the xtrabackup directory.

The properties of the backup file, including the tool used to create the backup, and the checksum information, are location in the corresponding `.properties` file. Note that each backup and property file is uniquely numbered so that it can be identified when restoring a specific backup.

A backup can also be initiated and run in the background by adding the & (ampersand) to the command:

```
[LOGICAL:EXPERT] /alpha > datasource db2 backup &
[1] datasource db2 backup - RUNNING
[LOGICAL:EXPERT] /alpha > Using the 'xtrabackup-full' backup agent.
Replicator 'db2' starting backup

Backup of dataSource 'db2' succeeded; uri=storage://file-system/store-0000000002.properties

[1] datasource db2 backup - SUCCESS
```

or:

```
shell> trepctl backup &
Backup completed successfully; URI=storage://file-system/store-0000000004.properties
```

6.10.1. Using a Different Backup Tool

If xtrabackup is installed when the dataservice is first created, xtrabackup will be used as the default backup method. Four built-in backup methods are provided:

- mysqldump — SQL dump to a single file. This is the easiest backup method but it is not appropriate for large data sets.
- xtrabackup — Full backup to a single tar file. This will take longer to take the backup and to restore.
- xtrabackup-full — Full backup to a directory (this is the default if xtrabackup is available and the backup method is not explicitly stated).
- xtrabackup-incremental — Incremental backup from the last xtrabackup-full or xtrabackup-incremental backup.
- mariabackup — Full backup to a single tar file. This will take longer to take the backup and to restore. As of version 6.1.18
- mariabackup-full — Full backup to a directory. As of version 6.1.18
- mariabackup-incremental — Incremental backup from the last mariabackup-full or mariabackup-incremental backup. As of version 6.1.18

The default backup tool can be changed, and different tools can be used explicitly when the backup command is executed. The Percona xtrabackup tool can be used to perform both full and incremental backups. Use of the this tool is optional and can configured during installation, or afterwards by updating the configuration using tpm .

To update the configuration to use xtrabackup , install the tool and then follow the directions for tpm update to apply the `--repl-backup-method=xtrabackup-full` [475] setting.

To use xtrabackup-full without changing the configuration, specify the backup agent to the datasource backup command within cctrl :

```
[LOGICAL:EXPERT] /alpha > datasource host2 backup xtrabackup-full
Replicator 'host2' starting backup
Backup of dataSource 'host2' succeeded; uri=storage://file-system/store-0000000006.properties
```

6.10.2. Automating Backups

Backups cannot be automated within Tungsten Cluster, instead a cron job should be used to automate the backup process. cluster_backup is packaged with Tungsten Cluster to provide a convenient interface with cron. The cron entry should be added to every datasource or active witness in the cluster. The command includes logic to ensure that it will only take one backup per cluster by only running on the current coordinator. See Section 9.8, "The cluster_backup Command" for more information.

```
shell> /opt/continuent/tungsten/cluster-home/bin/cluster_backup -v >> »
/opt/continuent/service_logs/cluster_backup.log
```

The command output will be appended to `/opt/continuent/service_logs/cluster_backup.log` for later review. Use your preferred mechanism to configure cron to execute this command on the desired schedule.

An example cron entry:

```
shell> crontab -l
00 00 * * * /opt/continuent/tungsten/cluster-home/bin/cluster_backup >> /opt/continuent/service_logs/cluster_backup.log 2>&1
```

All output will be appended to `/opt/continuent/service_logs/cluster_backup.log`.

Alternatively, you can call the backup command directly through cctrl. This method does not ensure the named datasource is `ONLINE` or even available to be backed up.

```
shell> echo "datasource host2 backup" | /opt/continuent/tungsten/tungsten-manager/bin/cctrl -expert
```

6.10.3. Using a Different Directory Location

The default backup location the `backups` directory of the Tungsten Cluster installation directory. For example, using the recommended installation location, backups are stored in `/opt/continuent/backups` .

See Section D.1.1.4, "Relocating Backup Storage" for details on changing the location where backups are stored.

6.10.4. Creating an External Backup

There are several considerations to take into account when you are using a tool other than Tungsten Cluster to take a backup. We have taken great care to build all of these into our tools. If the options provided do not meet your needs, take these factors into account when taking your own backup.

- How big is your data set?

  The mysqldump tool is easy to use but will be very slow once your data gets too large. We find this happens around 1GB. The xtrabackup tool works on large data sets but requires more expertise. Choose a backup mechanism that is right for your data set.

- Is all of your data in transaction-safe tables?

  If all of your data is transaction-safe then you will not need to do anything special. If not then you need to take care to lock tables as part of the backup. Both mysqldump and xtrabackup take care of this. If you are using other mechanisms you will need to look at stopping the replicator, stopping the database. If you are taking a backup of the Primary then you may need to stop all access to the database.

- Are you taking a backup of the Primary?

  The Tungsten Replicator stores information in a schema to indicate the restart position for replication. On the Primary there can be a slight lag between this position and the actual position of the Primary. This is because the database must write the logs to disk before Tungsten Replicator can read them and update the current position in the schema.

  When taking a backup from the Primary, you must track the actual binary log position of the Primary and start replication from that point after restoring it. See Section 6.11.2, "Restoring an External Backup" for more details on how to do that. When using mysqldump use the `--master-data=2` option. The xtrabackup tool will print the binary log position in the command output.

Using mysqldump can be a very simple way to take consistent backup. Be aware that it can cause locking on MyISAM tables so running it against your Primary will cause application delays. The example below shows the bare minimum for arguments you should provide:

```
shell> mysqldump --opt --single-transaction --all-databases --add-drop-database --master-data=2
```

6.11. Restoring a Backup

If a restore is being performed as part of the recovery procedure, consider using the tprovision tool. This will work for restoring from the Primary or a Replica and is faster when you do not already have a backup ready to be restored. For more information, see Section 6.6.1.2, "Provision or Reprovision a Replica".

To restore a backup, use the restore command to a datasource within cctrl, or the trepctl :

- Shun the datasource to be restored, and put the replicator service offline using cctrl :

```
[LOGICAL] /alpha > datasource host2 shun
[LOGICAL] /alpha > replicator host2 offline
```

- Restore the backup using cctrl :

```
[LOGICAL] /alpha > datasource host2 restore
```

By default, the restore process takes the latest backup available for the host being restored.

Tungsten Cluster *does not* automatically locate the latest backup within the dataservice across all datasources.

Restoring within Multi-Active environments

The steps above cover a basic restore process in a single cluster, however, if the topology in use is a Multi-Active Topology, a few additional steps need to be taken

Composite Active/Active: Prior to issuing the restore command, the additional sub-services and cross-site replicators need to be stopped, using the following examples as a guide:

```
cctrl> use east_from_west
cctrl> datasource host2 shun
host2-shell> trepctl -all-services offline
```

Multi-Site/Active-Active: Prior to issuing the restore command, the additional sub-services and cross-site replicators need to be stopped, using the following examples as a guide:

```
host2-shell> trepctl -all-services offline
host2-shell> mm_trepctl offline
```

After restoring the database, explicitly restart the cross site replicator:

```
host2-shell> mm_trepctl online
```

6.11.1. Restoring a Specific Backup

To restore a specific backup, specify the location of the corresponding properties file using the format:

```
storage://storage-type/location
```

For example, to restore the backup from the filesystem using the information in the properties file `store-0000000004.properties` , login to the failed host:

- Shun the datasource to be restored, and put the replicator service offline using cctrl :

```
[LOGICAL] /alpha > datasource host2 shun
[LOGICAL] /alpha > replicator host2 offline
```

- Restore the backup using cctrl :

```
[LOGICAL] /alpha > datasource host2 restore storage://file-system/store-0000000004.properties
```

  The supplied location is identical to that returned when a backup is performed.

- or:

6.11.2. Restoring an External Backup

If a backup has been performed outside of the Tungsten softare, for example from filesystem snapshot or a backup performed outside of the dataservice, follow these steps:

1. Shun the datasource to be restored, and put the replicator service offline using cctrl :

```
[LOGICAL:EXPERT] /alpha > datasource host2 shun
[LOGICAL:EXPERT] /alpha > replicator host2 offline
```

2. Reset the THL, either using thl or by deleting the files directly :

```
shell> thl -service alpha purge
```

3. Restore the data or files using the external tool. This may require the database server to be stopped. If so, you should restart the database server before moving to the next step.

> **Note**
>
> The backup must be complete and the `tungsten` specific schemas must be part of the recovered data, as they are required to restart replication at the correct point. See Section 6.10.4, "Creating an External Backup" for more information on creating backups.

4. There is some additional work if the backup was taken of the Primary server. There may be a difference between the binary log position of the Primary and what is represented in the `trep_commit_seqno` [693]. If these values are the same, you may proceed without further work. If not, the content of `trep_commit_seqno` [693] must be updated.

   - Retrieve the contents of `trep_commit_seqno` [693] :

     ```
     shell> echo "select seqno,source_id, eventid from tungsten_alpha.trep_commit_seqno" | tpm mysql
     seqno source_id eventid
     32033674 host1 mysql-bin.000032:0000000473860407;-1
     ```

   - Compare the results to the binary log position of the restored backup. For this example we will assume the backup was taken at `mysql-bin.000032:473863524`. Return to the Primary and find the correct sequence number for that position :

     ```
     shell> ssh host1

     shell> thl dsctl -event mysql-bin.000032:0000000473863524
     dsctl -service alpha set -reset -seqno 7748 -epoch 0 -event-id "mysql-bin.000032:0000000473863524" -source-id "db1-east.continuent.com"

     ~OR~

     shell> thl list -event mysql-bin.000032:0000000473863524 -headers
     SEQ# = 7748 / FRAG# = 0 (last frag)
     - FILE = thl.data.0000000010
     - TIME = 2014-10-17 16:58:11.0
     - EPOCH# = 0
     - EVENTID = mysql-bin.000032:0000000473863524;-1
     - SOURCEID = db1-east.continuent.com

     shell> exit
     ```

   - Return to the Replica node and run dsctl set to update the `trep_commit_seqno` [693] table :

     ```
     shell> dsctl -service alpha set -reset \
          -seqno 7748 \
          -epoch 0 \
          -source-id db1-east.continuent.com \
          -event-id mysql-bin.000032:0000000473863524
     ```

5. Recover the datasource using cctrl :

   ```
   [LOGICAL] /alpha > datasource host2 recover
   ```

   The recover command will start the dataserver if it was left running and then bring the replicator and other operations online.

6.11.3. Restoring from Another Replica

If a restore is being performed as part of the recovery procedure, consider using the tprovision tool. This is will work for restoring from the Primary or a Replica and is faster if you do not already have a backup ready to be restored. For more information, see Section 6.6.1.2, "Provision or Reprovision a Replica".

Data can be restored to a Replica by performing a backup on a different Replica, transferring the backup information to the Replica you want to restore, and then running restore process.

For example, to restore the `host3` from a backup performed on `host2` :

1. Run the backup operation on `host2` :

   ```
   [LOGICAL:EXPERT] /alpha > datasource host2 backup
   Using the 'xtrabackup' backup agent.
   Replicator 'host2' starting backup
   Backup of dataSource 'host2' succeeded; uri=storage://file-system/store-0000000006.properties
   ```

2. Copy the backup information from `host2` to `host3`. See Section D.1.1.3, "Copying Backup Files" for more information on copying backup information between hosts. If you are using xtrabackup there will be additional files needed before the next step. The example below uses scp to copy a mysqldump backup:

   ```
   shell> cd /opt/continuent/backups
   shell> scp store-[0]*6[\.-]* host3:$PWD/
   store-0000000006-mysqldump-812096863445699665.sql                    100%  234MB  18.0MB/s   00:13
   ```

```
store-0000000006.properties                                   100%  314     0.3KB/s   00:00
```

If you are using xtrabackup:

```
shell> cd /opt/continuent/backups/xtrabackup
shell> rsync -aze ssh full_xtrabackup_2014-08-16_15-44_86 host3:$PWD/
```

3. Shun the datasource to be restored, and put the replicator service offline using cctrl :

```
[LOGICAL] /alpha > datasource host2 shun
[LOGICAL] /alpha > replicator host2 offline
```

4. Restore the backup using cctrl :

```
[LOGICAL] /alpha > datasource host2 restore
```

Once the restore operation has completed, the datasource will be placed into the online state.

> **Note**
>
> Check the ownership of files if you have trouble transferring files or restoring the backup. They should be owned by the Tungsten system user to ensure proper operation.

6.11.4. Manually Recovering from Another Replica

In the event that a restore operation fails, or due to a significant failure in the dataserver, an alternative option is to seed the failed dataserver directly from an existing running Replica.

For example, on the host `host2` , the data directory for MySQL has been corrupted, and mysqld will no longer start. This status can be seen from examining the MySQL error log in `/var/log/mysql/error.log:`

```
130520 14:37:08 [Note] Recovering after a crash using /var/log/mysql/mysql-bin
130520 14:37:08 [Note] Starting crash recovery...
130520 14:37:08 [Note] Crash recovery finished.
130520 14:37:08 [Note] Server hostname (bind-address): '0.0.0.0'; port: 13306
130520 14:37:08 [Note]   - '0.0.0.0' resolves to '0.0.0.0';
130520 14:37:08 [Note] Server socket created on IP: '0.0.0.0'.
130520 14:37:08 [ERROR] Fatal error: Can't open and lock privilege tables: Table 'mysql.host' doesn't exist
130520 14:37:08 [ERROR] /usr/sbin/mysqld: File '/var/run/mysqld/mysqld.pid' not found (Errcode: 13)
130520 14:37:08 [ERROR] /usr/sbin/mysqld: Error reading file 'UNKNOWN' (Errcode: 9)
130520 14:37:08 [ERROR] /usr/sbin/mysqld: Error on close of 'UNKNOWN' (Errcode: 9)
```

Performing a restore operation on this Replica may not work. To recover from another running Replica, `host3` , the MySQL data files can be copied over to `host2` directly using the following steps:

1. Shun the `host2` datasource to be restored, and put the replicator service offline using cctrl :

```
[LOGICAL] /alpha > datasource host2 shun
[LOGICAL] /alpha > replicator host2 offline
```

2. Shun the `host3` datasource to be restored, and put the replicator service offline using cctrl :

```
[LOGICAL] /alpha > datasource host3 shun
[LOGICAL] /alpha > replicator host3 offline
```

3. Stop the mysqld service on `host2`:

```
shell> sudo /etc/init.d/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. Recover `host3` back to the dataservice:

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

9. Update the ownership and permissions on the data files on `host2`:

```
host2 shell> sudo chown -R mysql:mysql /var/lib/mysql
host2 shell> sudo chmod 770 /var/lib/mysql
```

10. Clear out the THL files on the target node `host2` so the Replica replicator service may start cleanly:

```
host2 shell> thl purge
```

11. Recover `host2` back to the dataservice:

```
[LOGICAL:EXPERT] /alpha > datasource host2 recover
```

The recover command will start MySQL and ensure that the server is accessible before restarting replication. if the MySQL instance does not start; correct any issues and attempt the recover command again.

6.11.5. Reprovision a MySQL Replica using rsync

The steps below will guide you through the process of restoring a MySQL Replica node by using rsync.

The following process has the following caveats:

- You can sustain downtime on the node used as a source.
- You can either ssh between hosts as root, or have root level access to temporarily change file ownership.

Steps

1. Establish `ssh` as root between hosts, or if you can't set up ssh as root, on the target host, `chown` ownership of mysql target directories to tungsten, then run rysnc as command in Step 3 as the tungsten user.
2. On failed host:
   - Shut down mysql if it is still running.
   - Determine all mysql data directories (datadir, binary log dir, etc) e.g. `/var/lib/mysql`
   - Clear out database files from the failed host.

     ```
     shell> rm -rf /var/lib/mysql/*
     ```
3. On Source host:
   - Within cctrl, shun the node that will used as the source.

     ```
     cctrl> datasource sourcenode shun
     ```
   - Shut down mysql.
   - Use rsync to copy the datafiles to the target (specify 'z' if CPU available for compression)

     ```
     sourcehost> rsync -avz --progress /source/dir/ targetHost:/target/dir/
     ```
4. Wait for the rsync to complete.
5. On Source host:
   - Restart mysql.
   - Bring the replicator online.

     ```
     shell> trepctl online
     ```
   - Wait for replication to catch up, then use cctrl to recover the node.

     ```
     cctrl> datasource sourcenode recover
     ```

6. On restored, target, host:

   - Fix the ownership on ALL data directories, e.g.

     ```
     shell> chown -R mysql: /var/lib/mysql
     ```

   - Start mysql.
   - Bring the replicator online.

     ```
     shell> trepctl online
     ```

   - Wait for replication to catch up, then use cctrl to recover the node

     ```
     cctrl> datasource targetnode recover
     ```

6.11.6. Rebuilding a Lost Datasource

If a datasource has been lost within the dataservice, for example, a complete hardware failure or disk crash, the datasource can be added back to the cluster once the operating system and other configuration have been completed. Essentially, the process is the same as when initially setting up your node, with the node being re-confirmed as part of the running service, installing and configuring only the returning node to the cluster.

In the following steps, the host `host3` is being recovered into the cluster:

1. Setup the host with the prerequisites, as described in Appendix B, *Prerequisites* .
2. Restore a snapshot of the data taken from another Replica into the dataserver. If you have existing backups of this Replica or another, they should be used. If not, take a snapshot of an existing Replica and use this to apply the data to the Replica. This will need to be performed outside of the Tungsten Cluster service using the native restore method for the backup method you have chosen. The backup must include the entire schema of your database, including the `tungsten` schemas for your services.
3. The next steps depend on the availability of the hostname. If the hostname of the datasource that was lost can be reused, then the host can be reconfigured within the existing service. If the hostname is not available, the service must be reconfigured to remove the old host, and add the new host.

   Reusing an Existing Hostname

   a. Obtain a copy of the `/etc/tungsten/tungsten.ini` from an existing node and copy this to the same location on the new host

   b. Download the same version of the clustering software used on the other nodes and place this within /opt/continuent/software.

   c. Extract the tar file and change into the new directory that will be created, for example:

      ```
      shell> tar zxvf tungsten-clustering-7.0.3-141.tar.gz
      shell> cd tungsten-clustering-7.0.3-141
      ```

   d. Install the software:

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

   Removing and Adding a new Host

   a. Remove the existing (lost) datasource from the cluster using cctrl . First switch to administrative mode:

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

      Remove the host from the dataservice:

      ```
      [ADMIN] /alpha > rm host3

      WARNING: This is an expert-level command:
      Incorrect use may cause data corruption
      or make the cluster unavailable.

      Do you want to continue? (y/n)>
      ```

   b. Update the `/etc/tungsten/tungsten.ini` on all hosts and change the hostname to match the new host being replaced.

   c. Update the installation across all the hosts:

      ```
      shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
      shell> tools/tpm update --replace-release
      ```

   d. Now follow the same steps in the section above "Reusing an Existing Hostname" to install the software on the new host

4. Use cctrl to check and confirm the operation of the restore datasource.

The restored host should be part of the cluster and accepting events from the Primary as configured.

6.11.7. Resetting an Entire Dataservice from Filesystem Snapshots

To restore an entire dataservice from filesystem snapshots, the steps below should be followed. The same snapshot should be used on each host so that data on each host is the same. The following steps should be followed:

1. Set the dataservice into the *MAINTENANCE* policy mode:

```
[LOGICAL:EXPERT] /alpha > set policy maintenance
```

2. The following steps must be completed on each server before completing the next step:

   a. Stop the Tungsten Cluster services:

   ```
   shell> stopall
   ```

   b. Stop MySQL:

   ```
   shell> sudo /etc/init.d/mysql stop
   ```

   c. Replace the MySQL data files with the filesystem or snapshot data.

   d. Delete the THL files for each of the services that need to be reset:

   ```
   shell> rm /opt/continuent/thl/alpha/*
   ```

   e. Start MySQL to perform maintenance on the Tungsten schemas:

   ```
   shell> sudo /etc/init.d/mysql start
   ```

   f. Delete any Tungsten service schemas:

   ```
   mysql> DROP DATABASE tungsten_alpha;
   ```

   Once these steps have been executed on all the servers in the cluster, the services can be restarted.

3. On the current Primary, start the Tungsten Cluster services:

```
shell> startall
```

Now start the services using the same command on each of the remaining servers.

6.12. Migrating and Seeding Data

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

If you are migrating an existing MySQL native replication deployment to use Tungsten Cluster or the standalone Tungsten Replicator the configuration must be updated to match the status of the Replica.

1. Deploy Tungsten Cluster using the model or system appropriate according to Chapter 2, *Deployment*. Ensure that the Tungsten Cluster is not started automatically by excluding the `--start` [514] or `--start-and-report` [514] options from the tpm commands.

2. On each Replica

   Confirm that native replication is working on all Replica nodes :

   MySQL 8.4+

```
shell> echo 'SHOW REPLICA STATUS\G' | tpm mysql | \
egrep 'Source_Host| Last_Error| Replica_SQL_Running'
                Source_Host: tr-ssl1
           Replica_SQL_Running: Yes
                 Last_Error:
```

   Upto MySQL 8.0

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

3. On the Primary and each Replica

   Reset the Tungsten Replicator position on all servers :

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

4. On the Primary

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

   ```
   shell> startall
   shell> trepctl online
   ```

5. On the Primary

   Put the cluster into *MAINTENANCE* mode using cctrl to prevent Tungsten Cluster automatically reconfiguring services:

   ```
   cctrl > set policy maintenance
   ```

6. On each Replica

   MySQL 8.4+

   Record the current Replica log position (as reported by the `Relay_Source_Log_File` and `Exec_Source_Log_Pos` output from `SHOW REPLICA STATUS`. Ideally, each Replica should be stopped at the same position:

   ```
   shell> echo 'SHOW REPLICA STATUS\G' | tpm mysql | \
   egrep 'Source_Host| Last_Error| Relay_Source_Log_File| Exec_Source_Log_Pos'
                   Source_Host: tr-ssl1
           Relay_Source_Log_File: mysql-bin.000025
                      Last_Error: Error executing row event: 'Table 'tungsten_alpha.heartbeat' doesn't exist'
             Exec_Source_Log_Pos: 181268
   ```

   Upto MySQL 8.0

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

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

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

7. On the Primary

   Take the replicator offline and clear the THL:

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

8. On the Primary

   Start replication, using the *lowest* binary log file and log position from the Replica information determined previously.

   ```
   shell> trepctl online -from-event 000025:181268
   ```

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

9. On each Replica

   a. Disable native replication to prevent native replication being accidentally started on the Replica.

      MySQL 8.4+

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

      Upto MySQL 8.0

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

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

```
shell> trepctl online
shell> startall
```

The Replica will start reading from the binary log position configured on the Primary.

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

```
shell> trepctl online -from-event 000025:188249
shell> startall
```

10. Check that replication is operating correctly by using trepctl status on the Primary and each Replica to confirm the correct position.
11. Use cctrl to confirm that replication is operating correctly across the dataservice on all hosts.
12. Put the cluster back into *AUTOMATIC* mode:

```
cctrl> set policy automatic
```

13. Update your applications to use the installed connector services rather than a direct connection.
14. Remove the `master.info` file on each Replica to ensure that when a Replica restarts, it does not connect up to the Primary MySQL server again.

Once these steps have been completed, Tungsten Cluster should be operating as the replication service for your MySQL servers. Use the information in Chapter 6, *Operations Guide* to monitor and administer the service.

6.12.2. Migrating from MySQL Native Replication Using a New Service

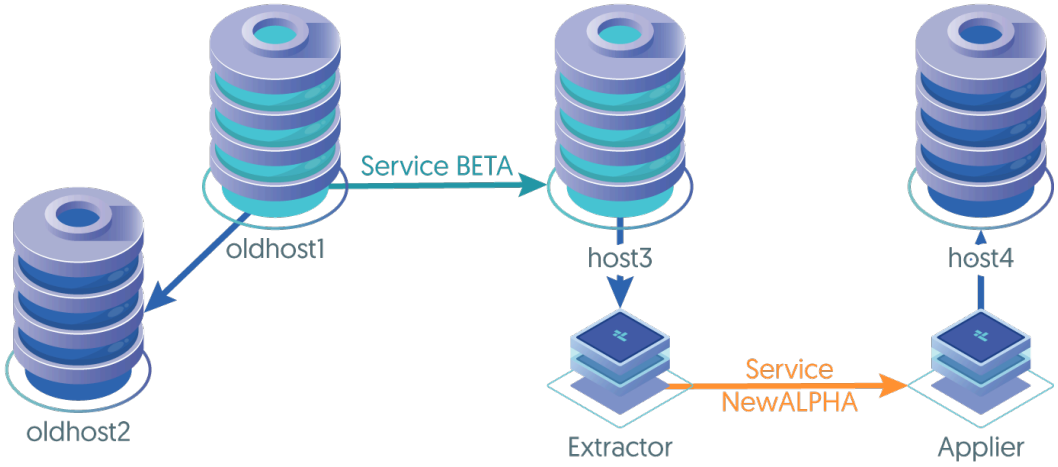
When running an existing MySQL native replication service that needs to be migrated to a Tungsten Cluster service, one solution is to create the new Tungsten Cluster service, synchronize the content, and then install a service that migrates data from the existing native service to the new service while 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 6.1, "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 6.1. Migration: Migrating Native Replication using a New Service



To configure the service:

1. Stop replication on a Replica for the existing native replication installation :

   MySQL 8.4+

```
mysql> STOP REPLICA;
```

Obtain the current Replica position within the Primary binary log :

```
mysql> SHOW REPLICA STATUS\G
...
                Source_Host: host3
       Relay_Source_Log_File: mysql-bin.000002
         Exec_Source_Log_Pos: 559
...
```

Upto MySQL 8.0

```
mysql> STOP SLAVE;
```

Obtain the current Replica position within the Primary binary log :

```
mysql> SHOW SLAVE STATUS\G
...
                Master_Host: host3
       Relay_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 Primary may be used if you do not have a Replica 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 Replica using native replication :

   MySQL 8.4+

   ```
   mysql> START REPLICA;
   ```

   Upto MySQL 8.0

   ```
   mysql> START SLAVE;
   ```

4. On the Primary and each Replica within the new service, restore the backup data and start the database service
5. Setup the new Tungsten Cluster 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 Primary, and the Primary of `newalpha`.

   For more information, see Section 3.8, “Replicating Data Into an Existing Dataservice”.

   Do not start the new service.
7. Set the replication position for `beta` using the dsctl set command to set the position to the point within the binary logs where the backup was taken:

   ```
   shell> /opt/replicator/tungsten/tungsten-replicator/bin/dsctl -service beta set -reset \
       -seqno 0 -epoch 0 \
       -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 Primary 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.

6.13. Resetting a Tungsten Cluster Dataservice

Follow these steps to reset replication for an entire dataservice. The current Primary will remain the Primary. Use the switch after completion to change the Primary.

Warning

The procedure in this section is specific to a single cluster only. Do not follow this if Composite clusters are in use.

See Section 6.11.7, "Resetting an Entire Dataservice from Filesystem Snapshots" if you would like to restore a file system snapshot to every server as part of this process.

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

```
cctrl> set policy maintenance
```

2. Put each Tungsten Connector offline:

```
cctrl> router * offline
```

> **Warning**
>
> This step is essential if applications are still active. Allowing data changes during this process could result in data loss.
>
> Ideally, all applications activity should be suspended until these steps are complete.

3. On each datasource, reset the service:

```
shell> trepctl offline
shell> trepctl reset -all -y
```

4. The connectors can now be re-enabled and the cluster returned to operational state:

```
cctrl> router * online
cctrl> set policy automatic
cctrl> cluster heartbeat
```

Any servers not matching the Primary must be reprovisioned. Use the tprovision tool to reprovision from the Primary or valid Replica server.

6.13.1. Reset a Single Site in a Multi-Site/Active-Active Topology

> **Warning**
>
> The procedures in this section are designed for the Multi-Site/Active-Active topology ONLY. Do NOT use these procedures with Composite Active/Active, Composite Active/Passive or Composite Dynamic Active/Active clusters.

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

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

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

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

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

2. Stop all services running on the primary node in `east`:

```
shell east> stopall
shell east> mm_replicator stop
```

3. Stop replicator services running on the replica nodes in `east`:

```
shell east> mm_replicator stop
```

4. Disable cross-site replication in `west`:

```
shell west> mm_trepctl offline
```

5. Reprovision the Primary node in `east` using one of the replica nodes in `west` as the source

6. Restart the services on the Primary node in `east`:

```
shell east> startall
```

7. Reprovision the replica nodes in `east` using one of the primary node in `east` as the source

8. Offline and reset the replicators on all nodes in `east`

```
shell east> trepctl offline
shell east> trepctl reset -all -y
```

```
shell east> trepctl online
```

9. Ensure all nodes in the east cluster are online by checking in cctrl

```
shell east> echo ls | cctrl
shell east> trepctl status
```

10. Reset the cross-site replicators in `west`, and bring back online:

```
shell east> mm_trepctl reset -all -y
shell east> mm_trepctl online
```

11. Bring the cross-site replicators in `east`, online:

```
shell east> mm_replicator start
```

12. Set the cluster to normal operational state:

```
cctrl> router * online
cctrl> set policy automatic
```

6.13.2. Reset All Sites in a Multi-Site/Active-Active topology

Warning

The procedures in this section are designed for the Multi-Site/Active-Active topology ONLY. Do NOT use these procedures with the Composite Active/Active topology.

For Composite Active/Active Clustering, please refer to Deploying Composite Active/Active Clustering.

Warning

It is important to ensure no data changes take place during the reset operation as this could result in data loss

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

1. Put the both dataservices into *`MAINTENANCE`* mode. This ensures that Tungsten Cluster will not attempt to automatically recover the service.

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

```
cctrl [west]> set policy maintenance
```

2. Put the Tungsten Connector for both dataservices offline:

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

```
cctrl [west]> router * offline
```

3. Place replicators `offline` in both clusters:

```
shell> trepctl offline
shell> mm_trepctl offline
```

4. Reset the replicators in both clusters:

```
shell> trepctl reset -all -y
shell> mm_trepctl reset -all -y
```

5. Place replicators `online` in both clusters:

```
shell> trepctl online
shell> mm_trepctl online
```

6. Return both clusters back to *`AUTOMATIC`* (This step will also bring the connectors online)

```
cctrl [east]> set policy automatic
```

```
cctrl [west]> set policy automatic
```

6.14. Replicator Fencing

Tungsten Cluster can be configured to handle failures during replication automatically and to fence the failure so that the issues do not lead to issues with the rest of the cluster, which may lead to problems with applications operating against the cluster. By default, the cluster is designed to take no specific action aside from indicating and registering the replicator so that the node will be identified as being within the `DIMINISHED` [149] or `CRITICAL` state.

This behavior can be changed so that the failed replicator is fenced, with configuration operating on either the Primary, or Replica replicators. When fencing has been enabled, the node will be placed into either the OFFLINE state if the node is a Replica or a failover will occur if the node is a Primary.

6.14.1. Fencing a Replica Node Due to a Replication Fault

If the replicator should be placed into the OFFLINE state when replicator stops or raises an error, the following option can be set through tpm on the cluster configuration to set the *policy.fence.slaveReplicator* to true:

```
shell> property=policy.fence.slaveReplicator=true
```

The delay before the fencing operation takes place can be configured using the *policy.fence.slaveReplicator.threshold* parameter, which configures the delay before taking action, with the value multiplied by 10. For example, a setting of 6 implies a delay of 60 seconds. The delay enables transient errors, such as network failures, to be effectively managed without automatically fencing the Replica.

```
shell> property=policy.fence.slaveReplicator.threshold=6
```

Once a Replica has been fenced, the state will automatically be cleared when the replicator returns to the ONLINE state. Once this has been identified, the node will be placed in the ONLINE state.

6.14.2. Fencing Primary Replicators

In the event of a Primary replicator failure, the fencing operation places the datasource into the FAILED state, triggering an automatic failover (see Section 6.5.1, “Automatic Primary Failover”. Because this triggers a failover in the event of fencing the replicator, the configuration should only be enabled if it critical for your business that replication errors/stops should trigger a significant operation as failover.

To enable fencing of the Primary node due to replication faults, use the *policy.fence.masterReplicator* configuration property when configuring the cluster:

```
shell> property=policy.fence.masterReplicator=true
```

The delay before the fencing operation takes place can be configured using the *policy.fence.masterReplicator.threshold* property. The default value is 6, or 60 seconds.

```
shell> property=policy.fence.masterReplicator.threshold=6
```

When the replicator is identified as available, the Primary datasource is not placed back into the online state. Instead, the failed datasource and must be explicitly recovered using the recover or datasource host recover commands.

6.15. Upgrading Tungsten Cluster

To upgrade an existing installation of Tungsten Cluster, 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, However the process that takes place should not present as an outage to your applications providing steps when upgrading the connectors are followed carefully. Any downtime is deliberately kept to a minimum, and the cluster should be in the same operational state once the upgrade has finished as it was when the upgrade was started.

Warning

During the update process, the cluster will be in *MAINTENANCE* mode. This is intentional to prevent unwanted failovers during the process, however it is important to understand that should the primary fail for genuine reasons NOT associated with the upgrade, then failover will also not happen at that time.

It is important to ensure clusters are returned to the *AUTOMATIC* state as soon as all Maintenance operations are complete and the cluster is stable.

Note

It is NOT advised to perform rolling upgrades of the tungsten software to avoid miscommunication between components running older/newer versions of the software that may prevent switches/failovers from occuring, therefore it is recommended to upgrade all nodes in place. The process of the upgrade places the cluster into *MAINTENANCE* mode which in itself avoids outages whilst components are restarted, and allows for a successful upgrade.

Warning

From version 7.1.0 onwards, the JGroup libraries were upgraded, this means that when upgrading to any release to 7.1.0 onwards FROM any release OLDER than 7.1.0 (for example 6.1.0 -> 7.1.4), all nodes must be upgraded before full cluster communication will be restored. For that reason upgrades to 7.1.0 onwards from an OLDER release MUST be performed together, ensuring the cluster is only running with a mix of manager versions for as little time as possible. When upgrading nodes, do NOT SHUN the node otherwise you will not be able to recover the node into the clus-

ter until all nodes are upgraded, which could result in an outage to your applications. Additionally, do NOT perform a switch until all nodes are upgraded. This means you should upgrade the master node in situ. Providing the cluster is in *MAINTENANCE*, this will not cause an outage and the cluster can still be upgraded with no visible outage to your applications.

6.15.1. Upgrading using the Staging Method

Important

As of v8, staging methods of installation and upgrade is deprecated and will be removed in a future release.

It is advisable to use the INI deployment method for existing and future deployments

To convert from staging to INI, see Section 10.3.1, "Using the `translatetoini.pl` Script"

Note

For INI file upgrades, see Section 6.15.2, "Upgrading when using INI-based configuration"

Warning

Before performing and upgrade, please ensure that you have checked the Appendix B, *Prerequisites*, as software and system requirements may have changed between versions and releases.

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

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

```
shell> tar zxf tungsten-clustering-7.0.3-141.tar.gz
```

3. Change to the extracted directory:

```
shell> cd tungsten-clustering-7.0.3-141
```

4. The next step depends on your existing deployment:

   - If you are upgrading a Multi-Site/Active-Active deployment:

     If you installed the original service by making use of the `$CONTINUENT_PROFILES` and `$REPLICATOR_PROFILES` environment variables, no further action needs to be taken to update the configuration information. Confirm that these variables are set before performing the validation and update.

     If you did not use these environment variables when deploying the solution, you must load the existing configuration from the current hosts in the cluster before continuing by using tpm fetch:

```
shell> ./tools/tpm fetch --hosts=east1,east2,east3,west1,west2,west3 \
    --user=tungsten --directory=/opt/continuent
```

     #### Important

     You must specify ALL the hosts within both clusters within the current deployment when fetching the configuration; use of the `autodetect` keyword will not collect the correct information.

   - If you are upgrading any other deployment:

     If you are are using the `$CONTINUENT_PROFILES` variable to specify a location for your configuration, make sure that the variable has been set correctly.

     If you are not using `$CONTINUENT_PROFILES`, a copy of the existing configuration must be fetched from the installed Tungsten Cluster installation:

```
shell> ./tools/tpm fetch --hosts=host1,host2,host3,autodetect \
    --user=tungsten --directory=/opt/continuent
```

     #### Important

     You must use the version of tpm from within the staging directory (./tools/tpm) of the new release, not the tpm installed with the current release.

     The current configuration information will be retrieved to be used for the upgrade:

```
shell> ./tools/tpm fetch --hosts=host1,host2,host3 --user=tungsten --directory=/opt/continuent
.......
NOTE  >> Configuration loaded from host1,host2,host3
```

5. Check that the update configuration matches what you expect by using tpm reverse:

```
shell> ./tools/tpm reverse
# Options for the dsone data service
tools/tpm configure dsone \
--application-password=password \
--application-port=3306 \
--application-user=app_user \
--connectors=host1,host2,host3 \
--datasource-log-directory=/var/log/mysql \
--install-directory=/opt/continuent \
--master=host1 \
--members=host1,host2,host3 \
'--profile-script=~/.bashrc' \
--replication-password=password \
--replication-port=13306 \
--replication-user=tungsten \
--start-and-report=true \
--user=tungsten \
--witnesses=192.168.0.1
```

6. Run the upgrade process:

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

> **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 Cluster updates. These issues should be addressed and the tpm update command re-executed.

The following additional options are available when updating:

- `--no-connectors` [504] (optional)

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

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

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

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

```
.....................................................................................................
Getting cluster status on host1
Tungsten Clustering 7.0.3 build 141
connect to 'dsone@host1'
dsone: session established
[LOGICAL] /dsone > ls

COORDINATOR[host3:AUTOMATIC:ONLINE]

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

...

####################################################################
# Next Steps
####################################################################
We have added Tungsten environment variables to ~/.bashrc.
Run `source ~/.bashrc` to rebuild your environment.

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

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

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

NOTE  >> Command successfully completed
```

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

6.15.2. Upgrading when using INI-based configuration

To perform an upgrade of an individual node, tpm can be used on the individual host.

> **Warning**
>
> Before performing and upgrade, please ensure that you have checked the Appendix B, *Prerequisites*, as software and system requirements may have changed between versions and releases.

> **Important**
>
> Application traffic to the nodes will be disconnected when the connector restarts. Use the `--no-connectors` [504] tpm option when you upgrade to prevent the connectors from restarting until later when you want them to.

6.15.2.1. Upgrading a Single Host using `tpm`

> **Note**
>
> For more information on performing maintenance across a cluster, see Section 6.16.3, "Performing Maintenance on an Entire Dataservice".

To upgrade a single host using the tpm command:

1. Download the release package.
2. Unpack the release package:

   ```
   shell> tar zxf tungsten-clustering-7.0.3-141.tar.gz
   ```

3. Change to the extracted directory:

   ```
   shell> cd tungsten-clustering-7.0.3-141
   ```

4. Execute tpm update, specifying the installation directory. This will update only this host:

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

The following additional options are available when updating, and would be recommended in production environments for hosts running connector services:

- `--no-connectors` [504] (optional)

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

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

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

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

6.15.3. Upgrade/Convert: From Multi-Site/Active-Active (MSAA) to Composite Active/Passive (CAP)

These steps are designed to guide you in the safe conversion of an existing Multi-Site/Active-Active (MSAA) topology to a Composite Active/Passive (CAP) topology, based on an ini installation.

For details of the difference between these two topologies, please review the following pages:

- Section 3.5, "Deploying Multi-Site/Active-Active Clustering"

- Section 3.2, “Deploying Composite Active/Passive Clustering”

> **Warning**
>
> It is very important to follow all the below steps and ensure full backups are taken when instructed. These steps can be destructive and without proper care and attention, data loss, data corruption or a split-brain scenario can happen.

> **Warning**
>
> Parallel apply MUST be disabled before starting your upgrade. You may re-enable it once the upgrade has been fully completed. See Section 4.1.5.3, “How to Disable Parallel Replication Safely” and Section 4.1.2, “Enabling Parallel Apply During Install” for more information.

> **Note**
>
> The examples in this section are based on three clusters named 'nyc', 'london' and 'tokyo'
>
> Each cluster has two dedicated connectors on separate hosts.
>
> The converted cluster will consist of a Composite Service named 'global' and the 'nyc' cluster will be the Active cluster, with 'london' and 'tokyo' as Passive clusters.
>
> If you do not have exactly three clusters, please adjust this procedure to match your environment.
>
> Examples of before and after tungsten.ini files can be downloaded here:
>
> - Multi-Site/Active-Active Config (Before)
> - Composite Active/Passive Config (After)

If you are currently installed using a staging-based installation, you must convert to an INI based installed for this process to be completed with minimal risk and minimal interuption. For notes on how to perform the staging to INI file conversion using the `translatetoini.pl` script, please visit Section 10.3.1, “Using the `translatetoini.pl` Script”.

6.15.3.1. Conversion Prerequisites

> **Warning**
>
> Parallel apply MUST be disabled before starting your upgrade. You may re-enable it once the upgrade has been fully completed. See Section 4.1.5.3, “How to Disable Parallel Replication Safely” and Section 4.1.2, “Enabling Parallel Apply During Install” for more information.

- Obtain the latest Tungsten Cluster software build and place it within `/opt/continuent/software`

  If you are not upgrading, just converting, then this step is not required since you will already have the extracted software bundle available.

- Extract the package
- The examples below refer to the `tungsten_prep_upgrade` script, this can be located in the extracted software package within the `tools` directory.

6.15.3.2. Step 1: Backups

Take a full and complete backup of one node - this can be a Replica, and preferably should be either performed by:

- Percona xtrabackup whilst database is open
- Manual backup of all datafiles after stopping the database instance

6.15.3.3. Step 2: Redirect Client Connections

A big difference between Multi-Site/Active-Active (MSAA) and Composite Active/Passive (CAP) is that with MSAA, clients can write into all clusters. With CAP clients only write into a single cluster.

To be able to complete this conversion process with minimal interuption and risk, it is essential that clients are redirected and only able to write into a single cluster. This cluster will become the ACTIVE cluster after the conversion. For the purpose of this procedure, we will use the 'nyc' cluster for this role.

After redirecting your client applications to connect through the connectors associated with the 'nyc' cluster, stop the connectors associated with the remaining clusters as an extra safeguard against writes happening

On every connector node associated with london and tokyo :

```
shell> connector stop
```

6.15.3.4. Step 3: Enter *MAINTENANCE* Mode

Enable *MAINTENANCE* mode on all clusters using the cctrl command:

```
shell> cctrl
cctrl> set policy maintenance
```

6.15.3.5. Step 4: Stop the Cross-site Replicators

Important

Typically the cross-site replicators will be installed within `/opt/replicator`, if you have installed this in a different location you will need to pass this to the script in the examples using the --path option

1. The following commands tell the replicators to go offline at a specific point, in this case when they receive an explicit heartbeat. This is to ensure that all the replicators stop at the same sequence number and binary log position. The replicators will NOT be offline until the explicit heartbeat has been issued a bit later in this step.

   - On every nyc node:

     ```
     shell> ./tungsten_prep_upgrade -o
     ~or~
     shell> ./tungsten_prep_upgrade --service london --offline
     shell> ./tungsten_prep_upgrade --service tokyo --offline
     ```

   - On every london node:

     ```
     shell> ./tungsten_prep_upgrade -o
     ~or~
     shell> ./tungsten_prep_upgrade --service nyc --offline
     shell> ./tungsten_prep_upgrade --service tokyo --offline
     ```

   - On every tokyo node:

     ```
     shell> ./tungsten_prep_upgrade -o
     ~or~
     shell> ./tungsten_prep_upgrade --service london --offline
     shell> ./tungsten_prep_upgrade --service tokyo --offline
     ```

2. Next, on the Primary hosts within each cluster we issue the heartbeat, execute the following using the cluster-specific trepctl, typically in `/opt/continuent`:

   ```
   shell> trepctl heartbeat -name offline_for_upg
   ```

   Ensure that every cross-site replicator on every node is now in the `OFFLINE:NORMAL` state:

   ```
   shell> mmtrepctl status
   ~or~
   shell> mmtrepctl --service {servicename} status
   ```

3. Capture the position of the cross-site replicators on all nodes in all clusters.

   The service name provided should be the name of the remote service(s) for this cluster, so for example in the london cluster you get the positions for nyc and tokyo, and in nyc you get the position for london and tokyo, etc.

   - On every london node:

     ```
     shell> ./tungsten_prep_upgrade -g
     ~or~
     shell> ./tungsten_prep_upgrade --service nyc --get
     (NOTE: saves to ~/position-nyc-YYYYMMDDHHMMSS.txt)
     shell> ./tungsten_prep_upgrade --service tokyo --get
     (NOTE: saves to ~/position-tokyo-YYYYMMDDHHMMSS.txt)
     ```

   - On every nyc node:

     ```
     shell> ./tungsten_prep_upgrade -g
     ~or~
     shell> ./tungsten_prep_upgrade --service london --get
     (NOTE: saves to ~/position-london-YYYYMMDDHHMMSS.txt)
     shell> ./tungsten_prep_upgrade --service tokyo --get
     (NOTE: saves to ~/position-tokyo-YYYYMMDDHHMMSS.txt)
     ```

- On every tokyo node:

```
shell> ./tungsten_prep_upgrade -g
~or~
shell> ./tungsten_prep_upgrade --service london --get
(NOTE: saves to ~/position-london-YYYYMMDDHHMMSS.txt)
shell> ./tungsten_prep_upgrade --service nyc --get
(NOTE: saves to ~/position-nyc-YYYYMMDDHHMMSS.txt)
```

4. Finally, to complete this step, stop the cross-site replicators on all nodes:

```
shell> ./tungsten_prep_upgrade --stop
```

6.15.3.6. Step 5: Export the tracking schema databases

On every node in each intended Passive cluster (london and tokyo), export the tracking schema associated the intended Active cluster (nyc)

Note the generated dump file is called `tungsten_global.dmp`. global refers to the name of the intended Composite Cluster service, if you choose a different service name, change this accordingly.

- On every london node:

```
shell> mysqldump --opt --single-transaction tungsten_nyc > ~/tungsten_global.dmp
```

- On every tokyo node:

```
shell> mysqldump --opt --single-transaction tungsten_nyc > ~/tungsten_global.dmp
```

6.15.3.7. Step 6: Uninstall the Cross-site Replicators

To uninstall the cross-site replicators, execute the following on every node:

```
shell> cd {replicator software path}
shell> tools/tpm uninstall --i-am-sure
```

6.15.3.8. Step 7: Create Composite Tracking Schema

In this step, we pre-create the database for the composite service tracking schema, we are using global as the service name in this example, if you choose a different Composite service name, adjust this accordingly

On every node in all clusters:

```
shell> mysql -e 'set session sql_log_bin=0; create database tungsten_global'
```

6.15.3.9. Step 8: Reload the tracking schema for Passive clusters

This step reloads the tracking schema associated with the intended Active cluster (nyc) into the tracking schema we created in the previous step. This should ONLY be carried out within the intended Passive clusters at this stage.

We DO NOT want the reloading of this schema to appear in the binary logs on the Primary, therefore the reload needs to be performed on each node individually:

- On every london node:

```
shell> mysql -e 'set session sql_log_bin=0; use tungsten_global; source ~/tungsten_global.dmp;'
```

- On every tokyo node:

```
shell> mysql -e 'set session sql_log_bin=0; use tungsten_global; source ~/tungsten_global.dmp;'
```

6.15.3.10. Step 9: Stop local cluster Replicators

On every node in every cluster:

```
shell> replicator stop
```

Warning

The effect of this step will now mean that only the Primary node in the Active cluster will be up to date with ongoing data changes. You must ensure that your applications handle this accordingly until the replicators are restarted at Step 14

6.15.3.11. Step 10: Remove THL

Warning

This step, if not followed correctly, could be destructive to the entire conversion. It is CRITICAL that this step is NOT performed on the intended Active cluster (nyc)

By default, THL files will be located within `/opt/continuent/thl`, if you have configured this in a different location you will need to adjust the path below accordingly

- On every london node:

```
shell> cd /opt/continuent/thl
shell> rm */thl*
```

- On every tokyo node:

```
shell> cd /opt/continuent/thl
shell> rm */thl*
```

6.15.3.12. Step 11: Export the tracking schema database on Active cluster

On every node within the intended Active cluster (nyc), export the tracking schema associated with the local service

Note the generated dump file is called `tungsten_global.dmp`. global refers to the name of the intended Composite Cluster service, if you choose a different service name, change this accordingly.

- On every nyc node:

```
shell> mysqldump --opt --single-transaction tungsten_nyc > ~/tungsten_global.dmp
```

6.15.3.13. Step 12: Reload the tracking schema for Active cluster

This step reloads the tracking schema associated with the intended Active cluster (nyc) into the tracking schema we created in the earlier step.

We DO NOT want the reloading of this schema to appear in the binary logs on the Primary, therefore the reload needs to be performed on each node individually:

- On every nyc node:

```
shell> mysql -e 'set session sql_log_bin=0; use tungsten_global; source ~/tungsten_global.dmp;'
```

6.15.3.14. Step 13: Update Configuration

Update `/etc/tungsten/tungsten.ini` to a valid Composite Active/Passive config. An example of a valid config is as follows, a sample can also be downloaded from Section 6.15.3.1, “Conversion Prerequisites” above:

Important

Within a Composite Active/Passive topology, the ini file must be identical on EVERY node, including Connector Nodes

```
[defaults]
user=tungsten
home-directory=/opt/continuent
application-user=app_user
application-password=secret
application-port=3306
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=secret
mysql-allow-intensive-checks=true
skip-validation-check=THLSchemaChangeCheck
rest-api-admin-user=apiuser
rest-api-admin-password=secret
connector-rest-api-address=0.0.0.0
manager-rest-api-address0.0.0.0
replicator-rest-api-address0.0.0.0

[nyc]
topology=clustered
master=db1
slaves=db2,db3
connectors=nyc-conn1,nyc-conn2
```

```
[london]
topology=clustered
master=db4
slaves=db5,db6
connectors=ldn-conn1,ldn-conn2b6
relay-source=nyc

[tokyo]
topology=clustered
master=db7
slaves=db8,db9
connectors=tky-conn1,tky-conn2
relay-source=nyc

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

6.15.3.15. Step 14: Install the Software on Active Cluster

Validate and install the new release on all nodes in the Active (nyc) cluster only:

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> tools/tpm validate-update
```

If validation shows no errors, run the install:

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

6.15.3.16. Step 15: Start Local Replicators on Active cluster

After the installation is complete on all nodes in the Active cluster, restart the replicator services:

```
shell> replicator start
```

After restarting, check the status of the replicator using the trepctl and check that all replicators are ONLINE:

```
shell> trepctl status
```

6.15.3.17. Step 16: Install the Software on remaining Clusters

Validate and install the new release on all nodes in the remaining Passive clusters (london and tokyo):

> **Important**
>
> The update should be performed on the Primary nodes within each cluster first, validation will report and error that the roles conflict (Primary vs Relay). This is expected and to override this warning the `-f` options should be used on the Primary nodes only

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> tools/tpm validate-update
```

If validation shows no errors, run the install:

```
On Primary Nodes:
shell> tools/tpm update --replace-release -f

On Replica Nodes:
shell> tools/tpm update --replace-release
```

6.15.3.18. Step 17: Start Local Replicators on remaining clusters

After the installation is complete on all nodes in the Active cluster, restart the replicator services:

```
shell> replicator start
```

After restarting, check the status of the replicator using the trepctl and check that all replicators are ONLINE:

```
shell> trepctl status
```

6.15.3.19. Step 18: Convert Datasource roles for Passive clusters

Following the upgrades, there are a number of "clean-up" steps that we need to perform within cctrl to ensure the datasource roles have been converted from the previous "master" roles to "relay" roles.

The following steps can be performed in a single cctrl session initiated from any node within any cluster

```
shell> cctrl

Connect to Active cluster
cctrl> use nyc

Check Status and verify all nodes online
cctrl> ls

Connect to COMPOSITE service
cctrl> use global

Place Active service online
cctrl> datasource nyc online

Connect to london Passive service
cctrl> use london

Convert old Primary to relay
cctrl> set force true
cctrl> datasource oldPrimaryhost offline
cctrl> datasource oldPrimaryhost relay

Repeat on tokyo Passive service
cctrl> use tokyo
cctrl> set force true
cctrl> datasource oldPrimaryhost offline
cctrl> datasource oldPrimaryhost relay

Connect to COMPOSITE service
cctrl> use global

Place Passive services online
cctrl> datasource london online
cctrl> datasource tokyo online

Place all clusters into AUTOMATIC
cctrl> set policy automatic
```

6.15.3.20. Step 19: Upgrade the Software on Connectors

Validate and install the new release on all connectors nodes:

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> tools/tpm validate-update
```

If validation shows no errors, run the install:

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

After upgrading previously stopped connectors, you will need to restart the process:

```
shell> connector restart
```

Warning

Upgrading a running connector will initiate a restart of the connector services, this will result in any active connections being terminated, therefore care should be taken with this process and client redirection should be handled accordingly prior to any connector upgrade/restart

6.15.4. Upgrade/Convert: From Multi-Site/Active-Active (MSAA) to Composite Active/Active (CAA)

These steps are specifically for the safe and successful upgrade (or conversion) of an existing Multi-Site/Active-Active (MSAA) topology, to a Composite Active/Active (CAA) topology.

Warning

It is very important to follow all the below steps and ensure full backups are taken when instructed. These steps can be destructive and without proper care and attention, data loss, data corruption or a split-brain scenario can happen.

Warning

Parallel apply MUST be disabled before starting your upgrade/conversion. You may re-enable it once the process has been fully completed. See Section 4.1.5.3, "How to Disable Parallel Replication Safely" and Section 4.1.2, "Enabling Parallel Apply During Install" for more information.

Note

The examples in this section are based on three clusters named 'nyc', 'london' and 'tokyo'

If you do not have exactly three clusters, please adjust this procedure to match your environment.

Click here for a video of the upgrade procedure, showing the full process from start to finish...

6.15.4.1. Supported Upgrade Paths

If you are currently installed using a staging-based installation, you must convert to an INI, since INI based installation is the only option supported for the Composite Active/Active deployments. For notes on how to perform the staging to INI file conversion using the `translatetoini.pl` script, please visit Section 10.3.1, “Using the `translatetoini.pl` Script”.

Click here for a video of the INI conversion procedure, showing the full process from start to finish...

Path	Supported
ini, in place	Yes
ini, with Primary switch	No
Staging	No
Staging, with --no-connectors	No

6.15.4.2. Upgrade Prerequisites

Warning

Parallel apply MUST be disabled before starting your upgrade. You may re-enable it once the upgrade has been fully completed. See Section 4.1.5.3, “How to Disable Parallel Replication Safely” and Section 4.1.2, “Enabling Parallel Apply During Install” for more information.

- Obtain the latest v7 (or greater) Tungsten Cluster software build and place it within `/opt/continuent/software`

  If you are not upgrading, just converting, then this step is not required since you will already have the extracted software bundle available. However you must be running v6 or greater of Tungsten Cluster to deploy a CAA topology.

- Extract the package
- The examples below refer to the `tungsten_prep_upgrade` script, this can be located in the extracted software package within the `tools` directory.

6.15.4.3. Step 1: Backups

Take a full and complete backup of one node - this can be a Replica, and preferably should be either performed by:

- Percona xtrabackup whilst database is open
- Manual backup of all datafiles after stopping the database instance

6.15.4.4. Step 2: Stop the Cross-site Replicators

Important

Typically the cross-site replicators will be installed within `/opt/replicator`, if you have installed this in a different location you will need to pass this to the script in the examples using the --path option

1. The following commands tell the replicators to go offline at a specific point, in this case when they receive an explicit heartbeat. This is to ensure that all the replicators stop at the same sequence number and binary log position. The replicators will NOT be offline until the explicit heartbeat has been issued a bit later in this step.

   - On every nyc node:

     ```
     shell> ./tungsten_prep_upgrade -o
     ~or~
     shell> ./tungsten_prep_upgrade --service london --offline
     shell> ./tungsten_prep_upgrade --service tokyo --offline
     ```

   - On every london node:

     ```
     shell> ./tungsten_prep_upgrade -o
     ```

```
~or~
shell> ./tungsten_prep_upgrade --service nyc --offline
shell> ./tungsten_prep_upgrade --service tokyo --offline
```

- On every tokyo node:

```
shell> ./tungsten_prep_upgrade -o
~or~
shell> ./tungsten_prep_upgrade --service london --offline
shell> ./tungsten_prep_upgrade --service tokyo --offline
```

2. Next, on the Primary hosts within each cluster we issue the heartbeat, execute the following using the cluster-specific trepctl, typically in `/opt/continuent`:

```
shell> trepctl heartbeat -name offline_for_upg
```

Ensure that every cross-site replicator on every node is now in the `OFFLINE:NORMAL` state:

```
shell> mmtrepctl status
~or~
shell> mmtrepctl --service {servicename} status
```

3. Capture the position of the cross-site replicators on all nodes in all clusters.

The service name provided should be the name of the remote service(s) for this cluster, so for example in the london cluster you get the positions for nyc and tokyo, and in nyc you get the position for london and tokyo, etc.

- On every london node:

```
shell> ./tungsten_prep_upgrade -g
~or~
shell> ./tungsten_prep_upgrade --service nyc --get
(NOTE: saves to ~/position-nyc-YYYYMMDDHHMMSS.txt)
shell> ./tungsten_prep_upgrade --service tokyo --get
(NOTE: saves to ~/position-tokyo-YYYYMMDDHHMMSS.txt)
```

- On every nyc node:

```
shell> ./tungsten_prep_upgrade -g
~or~
shell> ./tungsten_prep_upgrade --service london --get
(NOTE: saves to ~/position-london-YYYYMMDDHHMMSS.txt)
shell> ./tungsten_prep_upgrade --service tokyo --get
(NOTE: saves to ~/position-tokyo-YYYYMMDDHHMMSS.txt)
```

- On every tokyo node:

```
shell> ./tungsten_prep_upgrade -g
~or~
shell> ./tungsten_prep_upgrade --service london --get
(NOTE: saves to ~/position-london-YYYYMMDDHHMMSS.txt)
shell> ./tungsten_prep_upgrade --service nyc --get
(NOTE: saves to ~/position-nyc-YYYYMMDDHHMMSS.txt)
```

4. Finally, to complete this step, stop the replicators on all nodes:

```
shell> ./tungsten_prep_upgrade --stop
```

6.15.4.5. Step 3: Export the tungsten_* Databases

On every node in each cluster, export the tracking schema for the cross-site replicator

Similar to the above step 2 when you captured the cross-site position, the same applies here, in london you export/backup nyc and tokyo, and in nyc you export/backup london and tokyo, and finally in tokyo you export/backup nyc and london.

- On every london node:

```
shell> ./tungsten_prep_upgrade -d --alldb
~or~
shell> ./tungsten_prep_upgrade --service nyc --dump
shell> ./tungsten_prep_upgrade --service tokyo --dump
```

- On every nyc node:

```
shell> ./tungsten_prep_upgrade -d --alldb
~or~
shell> ./tungsten_prep_upgrade --service london --dump
shell> ./tungsten_prep_upgrade --service tokyo --dump
```

- On every tokyo node:

```
shell> ./tungsten_prep_upgrade -d --alldb
~or~
shell> ./tungsten_prep_upgrade --service london --dump
shell> ./tungsten_prep_upgrade --service nyc --dump
```

6.15.4.6. Step 4: Uninstall the Cross-site Replicators

To uninstall the cross-site replicators, execute the following on every node:

```
shell> cd {replicator software path}
shell> tools/tpm uninstall --i-am-sure
```

6.15.4.7. Step 5: Reload the tracking schema

We DO NOT want the reloading of this schema to appear in the binary logs on the Primary, therefore the reload needs to be performed on each node individually:

- On every london node:

```
shell> ./tungsten_prep_upgrade -s nyc -u tungsten -w secret -r
shell> ./tungsten_prep_upgrade -s tokyo -u tungsten -w secret -r
~or~
shell> ./tungsten_prep_upgrade --service nyc --user tungsten --password secret --restore
shell> ./tungsten_prep_upgrade --service tokyo --user tungsten --password secret --restore
```

- On every tokyo node:

```
shell> ./tungsten_prep_upgrade -s london -u tungsten -w secret -r
shell> ./tungsten_prep_upgrade -s nyc -u tungsten -w secret -r
~or~
shell> ./tungsten_prep_upgrade --service london --user tungsten --password secret --restore
shell> ./tungsten_prep_upgrade --service nyc --user tungsten --password secret --restore
```

- On every nyc node:

```
shell> ./tungsten_prep_upgrade -s london -u tungsten -w secret -r
shell> ./tungsten_prep_upgrade -s tokyo -u tungsten -w secret -r
~or~
shell> ./tungsten_prep_upgrade --service london --user tungsten --password secret --restore
shell> ./tungsten_prep_upgrade --service tokyo --user tungsten --password secret --restore
```

6.15.4.8. Step 6: Update Configuration

Update `/etc/tungsten/tungsten.ini` to a valid CAA configuration. An example of a valid configuration is as follows:

```
[defaults]
user=tungsten
home-directory=/opt/continuent
application-user=app_user
application-password=secret
application-port=3306
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=secret
mysql-allow-intensive-checks=true
skip-validation-check=THLSchemaChangeCheck
start-and-report=true
rest-api-admin-user=apiuser
rest-api-admin-password=secret
connector-rest-api-address=0.0.0.0
manager-rest-api-address0.0.0.0
replicator-rest-api-address0.0.0.0

[nyc]
topology=clustered
master=db1
members=db1,db2,db3
connectors=db1,db2,db3

[london]
topology=clustered
master=db4
members=db4,db5,db6
connectors=db4,db5,db6

[tokyo]
topology=clustered
```

```
master=db7
members=db8,db8,db9
connectors=db7,db8,db9

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

Warning

It is critical that you ensure the `master=` entry in the configuration matches the current, live Primary host in your cluster for the purpose of this process.

6.15.4.9. Step 7: Enter *MAINTENANCE* Mode

Enable *MAINTENANCE* mode on all clusters using the cctrl command:

```
shell> cctrl
cctrl> set policy maintenance
```

6.15.4.10. Step 8: Stop Managers

Stop the manager process on all nodes:

```
shell> manager stop
```

6.15.4.11. Step 9: Install/Update the Software

Run the update as follows:

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

Important

If you had start-and-report=false you may need to restart manager services

Warning

Until all nodes have been updated, the output from cctrl may show services in an OFFLINE, STOPPED, or UNKNOWN state. This is to be expected until all the new managers are online

6.15.4.12. Step 10: Start Managers

After the installation is complete on all nodes, start the manager services:

```
shell> manager start
```

6.15.4.13. Step 11: Return to *AUTOMATIC* Mode

Return all clusters to *AUTOMATIC* mode using the cctrl command:

```
shell> cctrl
cctrl> set policy automatic
```

6.15.4.14. Step 12: Validate

1. Identify the cross-site service name[s]:

   ```
   shell> trepctl services
   ```

   In our example, the local cluster service will one of `london`, `nyc` or `tokyo` depending on the node you are on. The cross site replication services would be:

   ```
   (within the london cluster)
   london_from_nyc
   london_from_tokyo

   (within the nyc cluster)
   nyc_from_london
   nyc_from_tokyo

   (within the tokyo cluster)
   ```

```
tokyo_from_london
tokyo_from_nyc
```

2. Upon installation, the new cross-site replicators will come online, it is possible that they may be in an `OFFLINE:ERROR` state due to a change in Epoch numbers, check this on the Primary in each cluster by looking at the output from the trepctl command.

   Check each service as needed based on the status seen above:

```
shell> trepctl -service london_from_nyc status
shell> trepctl -service london_from_tokyo status
~or~
shell> trepctl -service nyc_from_london status
shell> trepctl -service nyc_from_tokyo status
~or~
shell> trepctl -service tokyo_from_london status
shell> trepctl -service tokyo_from_nyc status
```

3. If the replicator is in an error state due to an epoch difference, you will see an error similar to the following:

```
pendingErrorSeqno      : -1
pendingExceptionMessage: Client handshake failure: Client response
validation failed: Log epoch numbers do not match: master source
ID=db1 client source ID=db4 seqno=4 server epoch number=0 client
epoch number=4
pipelineSource         : UNKNOWN
```

   The above error is due to the epoch numbers changing as a result of the replicators being restarted, and the new replicators being installed.

   To resolve, simply force the replicator online as follows:

```
shell> trepctl -service london_from_nyc online -force
shell> trepctl -service london_from_tokyo online -force
~or~
shell> trepctl -service nyc_from_london online -force
shell> trepctl -service nyc_from_tokyo online -force
~or~
shell> trepctl -service tokyo_from_london online -force
shell> trepctl -service tokyo_from_nyc online -force
```

4. If the replicator shows an error state similar to the following:

```
pendingErrorSeqno      : -1
pendingExceptionMessage: Client handshake failure: Client response
validation failed: Master log does not contain requested
transaction: master source ID=db1 client source ID=db2 requested
seqno=1237 client epoch number=0 master min seqno=5 master max
seqno=7
pipelineSource         : UNKNOWN
```

   The above error is possible if during install the Replica replicators came online before the Primary.

   Providing the steps above have been followed, just bringing the replicator online should be enough to get the replicator to retry and carry on successfully:

```
shell> trepctl -service london_from_nyc online
shell> trepctl -service london_from_tokyo online
~or~
shell> trepctl -service nyc_from_london online
shell> trepctl -service nyc_from_tokyo online
~or~
shell> trepctl -service tokyo_from_london online
shell> trepctl -service tokyo_from_nyc online
```

Important

Known Issue (CT-569)

During an upgrade, the tpm process will incorrectly create additional, empty, tracking schemas based on the service names of the auto-generated cross-site services.

For example, if your cluster has service names *east* and *west*, you should only have tracking schemas for *tungsten_east* and *tungsten_west*

In some cases, you will also see *tungsten_east_from_west* and/or *tungsten_west_from_east*

These *tungsten_x_from_y* tracking schemas will be empty and unused. They can be safely removed by issuing DROP DATABASE *`tungsten_x_from_y`* on a Primary node, or they can be safely ignored

6.15.5. Installing an Upgraded JAR Patch

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

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

1. Determine your staging directory or untarred installation directory:

   ```
   shell> tpm query staging
   ```

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

   ```
   shell>  cd tungsten-clustering-7.0.3-141
   ```

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

   ```
   shell> cd tungsten-replicator/lib
   ```

3. Copy the existing JAR to a backup file:

   ```
   shell> cp tungsten-replicator.jar tungsten-replicator.jar.orig
   ```

4. Copy the replacement JAR into the directory:

   ```
   shell> cp /tmp/tungsten-replicator.jar .
   ```

5. Change back to the root directory of the staging directory:

   ```
   shell> cd ../..
   ```

6. Update the release:

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

6.15.6. Upgrading to v7

> **Warning**
>
> v7 is a major release with many changes, specifically to security. At this time, upgrading directly to v7 is only supported from v5 onwards. If security is NOT enabled in your installation, then upgrading from an older release may work, however any issues encountered will not be addressed and upgrading to v6 first will be the advised route.

> **Warning**
>
> Whilst every care has been taken to ensure upgrades are as smooth and easy as possible, ALWAYS ensure full backups are taken before proceeding, and if possible, test the upgrade on a non-Production environment first.

6.15.6.1. Background

6.15.6.1.1. v6 (and earlier) behavior

Prior to v7, Tungsten came with security turned OFF through the tpm flag `disable-security-controls` [490] set to `true` by default. This flag, when set to `false` would translate to the following settings being applied:

```
file-protection-level=0027
rmi-ssl=true
thl-ssl=true
rmi-authentication=true
jgroups-ssl=true
```

This would enable SSL communication between Tungsten components. However, connection to the database remained unencrypted, which would translate to the following settings being applied:

```
datasource-enable-ssl=false
connector-ssl=false
```

Setting these to true is possible, however there are many more manual steps that would have been required.

6.15.6.1.2. New behavior in v7

v7 enables full security by default, so the `disable-security-controls` [490] flag will default to `false` when not specified.

In addition to the default value changing, `disable-security-controls` [490] now enables encrypted communication to the database. Setting this value to `false`, now translates to the following settings being applied:

```
file-protection-level=0027
rmi-ssl=true
thl-ssl=true
rmi-authentication=true
jgroups-ssl=true
datasource-enable-ssl=true
connector-ssl=true
```

6.15.6.1.3. Summary

In summary, this change in behavior means that upgrades need to be handled with care and appropriate decisions being made, both by the tpm process, and by the "human" to decide on what end result is desired. The various options and examples are outlined in the following sections of this document.

6.15.6.2. Upgrade Decisions

6.15.6.2.1. Keep existing level of security

This is the easiest and smoothest approach. tpm will process your configuration and do its best to maintain the same level of security. In order to achieve that, tpm will dynamically update your configuration with additional properties to adjust the level of security to match.

The properties that tpm will add to your configuration will be some or all of the following depending on the initial starting point of your configuration:

```
disable-security-controls
connector-rest-api-ssl
manager-rest-api-ssl
replicator-rest-api-ssl
datasource-enable-ssl
enable-connector-ssl
```

You can now proceed with the upgrade, refer to Section 6.15.6.7, “Steps to upgrade using tpm” for the required steps

6.15.6.2.2. Apply new recommendations and setup security

The following security setting levels can be enabled, and will require user action prior to upgrading. These are:

1. Internal Encryption and Authentication
2. Tungsten to Database Encryption
3. Application (Connector) to Database Encryption
4. API SSL

Applying all of the above steps will bring full security, equivalent to the default.

The steps to enable will depend on what (if any) security is enabled in your existing installation. The following sections outline the steps required to be performed to enable security for each of the various layers. To understand whether you have configured any of the various layers of security, the following summary will help to understand your configuration:

No Security

If no security has been configured, the installation that you are starting from will have `disable-security-controls=true` [490] (or it will not supplied at all if upgrading from v6) and no additional security properties will be supplied.

Partial Security

The installation that you are starting from will have partial security in place. This could be a combination of any of the following:

- Internal encryption is configured (`disable-security-controls=false` [490]), and/or
- Connector encryption is enabled (`enable-connector-ssl=true` [491]) and/or
- Cluster to the database encryption is enabled (`datasource-enable-ssl=true` [486] or `repl-datasource-enable-ssl=true` [486])

To upgrade and enable security, you should follow one or more of the following steps based on your requirements. At a minimum, the first step should always be included, the remaining steps are optional.

1. Section 6.15.6.3, "Setup internal encryption and authentication"
2. Section 6.15.6.4, "Enable Tungsten to Database Encryption" (if required)
3. Section 6.15.6.5, "Enable Connector to Database Encryption" (if required)

6.15.6.3. Setup internal encryption and authentication

Prior to running the upgrade, you need to manually create the keystore, to do this follow these steps on one host, and then copy the files to all other hosts in your topology:

```
db1> mkdir /etc/tungsten/secure
db1> keytool -genseckey -alias jgroups -validity 3650 -keyalg Blowfish -keysize 56 \
-keystore /etc/tungsten/secure/jgroups.jceks -storepass tungsten -keypass tungsten -storetype JCEKS
```

If this is the only level of security you plan on configuring you should now copy these new keystores to all other hosts in your topology.

```
db1> for host in db2 db3 db4 db5 db6; do
ssh ${host} mkdir /etc/tungsten/secure
scp /etc/tungsten/secure/*.jceks ${host}:/etc/tungsten/secure
done
```

Enabling internal encryption and authentication will also enable API SSL by default.

If you need to enable encryption to the underlying database, now proceed to the next step Section 6.15.6.4, "Enable Tungsten to Database Encryption" before running the upgrade, otherwise you can then start the upgrade by following the steps in Section 6.15.6.7, "Steps to upgrade using tpm".

The following additional configuration properties will need adding to your existing configuration.

```
disable-security-controls=false
connector-rest-api-ssl=true
manager-rest-api-ssl=true
replicator-rest-api-ssl=true
java-jgroups-keystore-path=/etc/tungsten/secure/jgroups.jceks
```

6.15.6.4. Enable Tungsten to Database Encryption

The following prerequisite steps must be performed before continuing with this step

- Section 6.15.6.3, "Setup internal encryption and authentication"
- Section 6.15.6.6, "Enable MySQL SSL"

In this step, you pre-create the various keystores required and register the MySQL certificates for Tungsten. Execute all of the following steps on a single host, for example, db1. In the example below it is assumed that the mysql certificates reside in `/etc/mysql/certs`. If you use the example syntax below, you will also need to ensure the following directory exists: `/etc/tungsten/secure`

These commands will import the MySQL certificates into the required Tungsten truststores.

```
db1> keytool -importkeystore -srckeystore /etc/mysql/certs/client-cert.p12 -srcstoretype PKCS12 \
-destkeystore /etc/tungsten/secure/keystore.jks -deststorepass tungsten -srcstorepass tungsten

db1> keytool -import -alias mysql -file /etc/mysql/certs/ca.pem -keystore /etc/tungsten/secure/truststore.ts \
-storepass tungsten -noprompt
```

If you do not intend to configure SSL for your applications (via Connectors), or if your connectors reside on remote, dedicated hosts, you should now copy all of the generated keystores and truststores to all of the other hosts.

```
db1> for host in db2 db3 db4 db5 db6; do
ssh ${host} mkdir /etc/tungsten/secure
scp /etc/tungsten/secure/*.jceks ${host}:/etc/tungsten/secure
scp /etc/tungsten/secure/*.jks ${host}:/etc/tungsten/secure
scp /etc/tungsten/secure/*.ts ${host}:/etc/tungsten/secure
done
```

If you need to enable encryption to the underlying database from the connectors, now proceed to Section 6.15.6.5, "Enable Connector to Database Encryption" before running the upgrade, alternatively you can now follow the steps outlined in Section 6.15.6.7, "Steps to upgrade using tpm"

The following additional configuration properties will need adding to your existing configuration.

```
datasource-enable-ssl=true
java-truststore-path=/etc/tungsten/secure/truststore.ts
```

```
java-truststore-password=tungsten
java-keystore-path=/etc/tungsten/secure/keystore.jks
java-keystore-password=tungsten
datasource-mysql-ssl-cert=/etc/mysql/certs/client-cert.pem
datasource-mysql-ssl-key=/etc/mysql/certs/client-key.pem
datasource-mysql-ssl-ca=/etc/mysql/certs/ca.pem
```

6.15.6.5. Enable Connector to Database Encryption

The steps outlined in this section will need to be performed on all nodes where the Connector has been installed

If you are also enabling Internal Encryption, you would have followed the steps in Section 6.15.6.3, "Setup internal encryption and authentication" and you would have a number of files already in /etc/tungsten/secure. This next step will pre-create the keystore and truststore, and register the MySQL certificates for the Connectors. Execute all of the following steps on a single host, in this example, db1. In the example below it is assumed the mysql certificates reside in `/etc/mysql/certs`. If you use the example syntax below, you will need to ensure the following directory also exists: `/etc/tungsten/secure`

```
db1> keytool -importkeystore -srckeystore /etc/mysql/certs/client-cert.p12 \
-srcstoretype PKCS12 -destkeystore /etc/tungsten/secure/tungsten_connector_keystore.jks \
-deststorepass tungsten -srcstorepass tungsten

db1> keytool -import -alias mysql -file /etc/mysql/certs/ca.pem \
-keystore /etc/tungsten/secure/tungsten_connector_truststore.ts -storepass tungsten -noprompt
```

Now that all of the necessary steps have been taken to create the various keystores, you now need to copy all of these files to all other hosts in your topology.

```
db1> for host in db2 db3 db4 db5 db6; do
ssh ${host} mkdir /etc/tungsten/secure
scp /etc/tungsten/secure/*.jceks ${host}:/etc/tungsten/secure
scp /etc/tungsten/secure/*.jks ${host}:/etc/tungsten/secure
scp /etc/tungsten/secure/*.ts ${host}:/etc/tungsten/secure
done
```

Once the steps above have been performed, you can then continue with the upgrade, following the steps outlined in Section 6.15.6.7, "Steps to upgrade using tpm"

The following additional configuration properties will need adding to your existing configuration.

```
enable-connector-ssl=true
java-connector-keystore-path=/etc/tungsten/secure/tungsten_connector_keystore.jks
java-connector-keystore-password=tungsten
java-connector-truststore-path=/etc/tungsten/secure/tungsten_connector_truststore.ts
java-connector-truststore-password=tungsten
```

6.15.6.6. Enable MySQL SSL

A prerequisite to enabling full security, is to enable SSL within your database if this isn't already configured. To do this, we can use the mysql_ssl_rsa_setup tool supplied with most distributions of MySQL. If you do not have this tool, or require more detail, you can refer to Section 5.13.1, "Enabling Database SSL". The steps below summarise the process using the mysql_ssl_rsa_setup

1. The first step is to setup the directories for the certs, perform this on ALL hosts in your topology:

   ```
   shell> sudo mkdir -p /etc/mysql/certs
   shell> sudo chown -R tungsten: /etc/mysql/certs/
   ```

   NB: The ownership is temporarily set to tungsten so that the subsequent scp will work between hosts.

2. This next step should be performed on just one single host, for the purpose of this example we will use db1 as the host:

   ```
   db1> mysql_ssl_rsa_setup -d /etc/mysql/certs/
   db1> openssl pkcs12 -export -inkey /etc/mysql/certs/client-key.pem \
   -name mysql -in /etc/mysql/certs/client-cert.pem -out /etc/mysql/certs/client-cert.p12 \
   -passout pass:tungsten
   ```

   Important

   When using OpenSSL 3.0 with Java 1.8, you MUST add the `-legacy` option to the openssl command.

   ```
   db1> for host in db2 db3 db4 db5 db6; do
   scp /etc/mysql/certs/* ${host}:/etc/mysql/certs
   done
   ```

3. Next, on every host we need to reset the directory ownership

   ```
   shell> sudo chown -R mysql: /etc/mysql/certs/
   shell> sudo chmod g+r /etc/mysql/certs/client-*
   ```

4. Now on every host, we need to reconfigure MySQL. Add the following properties into your `my.cnf`

```
[mysqld]
ssl-ca=/etc/mysql/certs/ca.pem
ssl-cert=/etc/mysql/certs/server-cert.pem
ssl-key=/etc/mysql/certs/server-key.pem

[client]
ssl-cert=/etc/mysql/certs/client-cert.pem
ssl-key=/etc/mysql/certs/client-key.pem
ssl-ca=/etc/mysql/certs/ca.pem
```

5. Next, place your cluster(s) into *MAINTENANCE* mode

```
shell> cctrl
cctrl> set policy maintenance
```

6. Restart MySQL for the new settings to take effect

```
shell> sudo service mysqld restart
```

7. Finally, return your cluster(s) into *AUTOMATIC* mode

```
shell> cctrl
cctrl> set policy automatic
```

6.15.6.7. Steps to upgrade using tpm

When you are ready to perform the upgrade, the following steps should be followed:

1. Ensure you place your cluster(s) into *MAINTENANCE* mode
2. If no additional steps taken, and you wish to maintain the same level of security, skip skip the next step.
3. Update your `/etc/tungsten/tungsten.ini` and include some, or all, of the options below depending on which steps you took earlier. All entries should be placed within the `[defaults]` stanza.

```
disable-security-controls=false
connector-rest-api-ssl=true
manager-rest-api-ssl=true
replicator-rest-api-ssl=true
java-jgroups-keystore-path=/etc/tungsten/secure/jgroups.jceks
```

If "Tungsten to Database Encryption" IS configured, also add:

```
datasource-enable-ssl=true
java-truststore-path=/etc/tungsten/secure/truststore.ts
java-truststore-password=tungsten
java-keystore-path=/etc/tungsten/secure/keystore.jks
java-keystore-password=tungsten
datasource-mysql-ssl-cert=/etc/mysql/certs/client-cert.pem
datasource-mysql-ssl-key=/etc/mysql/certs/client-key.pem
datasource-mysql-ssl-ca=/etc/mysql/certs/ca.pem
```

If "Tungsten to Database Encryption" IS NOT configured, also add:

```
datasource-enable-ssl=false
```

If "Application (Connector) to Database Encryption" IS configured, also add:

```
enable-connector-ssl=true
java-connector-keystore-path=/etc/tungsten/secure/tungsten_connector_keystore.jks
java-connector-keystore-password=tungsten
java-connector-truststore-path=/etc/tungsten/secure/tungsten_connector_truststore.ts
java-connector-truststore-password=tungsten
```

If "Application (Connector) to Database Encryption" IS NOT configured, also add:

```
enable-connector-ssl=false
```

> **Important**
>
> If `start-and-report=true` [514], remove this value or set to `false`

4. Obtain the TAR or RPM package for your installation. If using a TAR file unpack this into your software staging tree on every host, typically `/opt/continuent/software`.
5. Change into the directory for the software

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
```

6. Issue the following command on all hosts.

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

When upgrading the connectors, you could include the optional `--no-connectors` option if you wish to control the restart of the connectors manually

7. For Multi-Site/Active-Active topologies, you will also need to repeat the steps for the cross-site replicators

8. Finally, before returning the cluster(s) to *AUTOMATIC*, you will need to sync the new certificates, created by the upgrade, to all hosts. This step will be required even if you have disabled security as these files will be used by the API and also, if you choose to enable it, THL Encryption.

From one host, copy the certificate and keystore files to ALL other hosts in your topology. The following scp command is an example assuming you are issuing from db1, and the install directory is /opt/continuent:

```
db1> for host in db2 db3 db4 db5 db6; do
scp /opt/continuent/share/[jpt]* ${host}:/opt/continuent/share
scp /opt/continuent/share/.[jpt]* ${host}:/opt/continuent/share
done
```

Note

The examples assume you have the ability to scp between hosts as the tungsten OS user. If your security restrictions do not permit this, you will need to use alternative procedures appropriate to your environment to ensure these files are in sync across all hosts before continuing.

If the files are not in sync between hosts, the software will fail to start!

9. You will also need to repeat this if you have a Multi-Site/Active-Active topology for the cross-site replicators:

```
db1> for host in db2 db3 db4 db5 db6; do
scp /opt/replicator/share[jpt]* ${host}:/opt/replicator/share
scp /opt/replicator/share.[jpt]* ${host}:/opt/replicator/share
done
```

10. Restart all tungsten components, one host at a time

```
shell> manager restart
shell> replicator restart
shell> connector restart
```

11. Return the cluster(s) to *AUTOMATIC* mode

6.15.6.8. Optional Post-Upgrade steps to configure API

Once the upgrade has been completed, if you plan on using the API you will need to complete a few extra steps before you can use it. By default, after installation the API will only allow the ping method and the createAdminUser method.

To open up the API and access all of its features, you will need to configure the API User. To do this, execute the following on all hosts (Setting the value of pass to your preferred password):

```
shell> curl -k -H 'Content-type: application/json' --request POST 'https://127.0.0.1:8096/api/v2/createAdminUser?i-am-sure=true' \
> --data-raw '{
>   "payloadType": "credentials",
>   "user":"tungsten",
>   "pass":"security"
> }'
```

For more information on using the new API, please refer to Chapter 11, *Tungsten REST APIv2*

6.16. Performing Database or OS Maintenance

When performing database or operating system maintenance, datasources should be temporarily removed from the dataservice and the replicator should be disabled. Follow these rules for the best results. Detailed steps are provided below for different scenarios.

Important

If you are upgrading MySQL from any 5.x release to version 8.x, and plan to maintain replication, you may encounter errors due to differences in SQL_MODEs and collations. To avoid such errors you will need to make temporary use of two additional filters, `dropsqlmode` and `mapcharset`

> After ALL nodes are running the same release of MySQL these filters can be removed.
>
> For a full explanation of this, see Section 6.16.4.2, "Upgrading between MySQL 5.x and MySQL 8.x"

- For maintenance operations on a Primary, the current Primary should be switched, the required maintenance steps performed, and then the Primary switched back.
- Disable a datasource using the datasource shun command.
- Put the replicator offline using trepctl offline.
- If you are using the Multi-Site/Active-Active topology, put the extra replicator offline using mm_trepctl offline. The mm_trepctl alias will only work if you configured Tungsten Replicator with the `executable-prefix=mm` [493] option.
- When making changes to a MySQL system the binary log should be disabled for your session. This will prevent corrective actions from replicating to other servers. Ignore this suggestion if you are making changes to a Primary that should be replicated.

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

- Restart replication and recover the datasource after maintenance is complete using datasource recover, trepctl online and optionally mm_trepctl online.

6.16.1. Performing Maintenance on a Single Replica

Performing maintenance on a single Replica can be achieved by temporarily shunning the Replica (while in *AUTOMATIC* policy mode) and doing the necessary maintenance. Shunning a datasource in this way will temporarily remove it from the dataservice, and prevent active and new connections from using the datasource for operations.

The steps are:

1. Shun the Replica:

   ```
   [LOGICAL] /alpha > datasource db2 shun
   ```

   Shunning a datasource does not put the replicator offline, so the replicator should also be put in the offline state to prevent replication and changes being applied to the database:

   ```
   [LOGICAL] /alpha > replicator db2 offline
   ```

2. Perform the required maintenance, including updating the operating system, software or hardware changes.
3. Recover the Replica back to the dataservice:

   ```
   [LOGICAL] /alpha > datasource db2 recover
   ```

Once the datasource is added back to the dataservice, the status of the node should be checked to ensure that the datasource has been correctly added back, and the node is `ONLINE` and up to date.

While the datasource is shunned, the node can be shutdown, restarted, upgraded, or any other maintenance. Throughout the process, the Replica should be monitored to ensure that the datasource is correctly added back into the dataservice, and has caught up with the Primary. Any problems should be addressed immediately.

6.16.2. Performing Maintenance on a Primary

Primary maintenance must be carried out when the Primary has been switched to a Replica, and then shunned. The Primary can be temporarily switched to a Replica, taken out of the dataservice through shunning, and then added back to the dataservice and then switched back again to be the Primary.

The complete sequence and commands required to perform maintenance on an active Primary are shown in the table below. The table assumes a dataservice with three datasources:

Step	Description	Command	host1	host2	host3
1	Initial state		Primary	Replica	Replica
2	Set the *MAINTENANCE* policy	set policy maintenance	Primary	Replica	Replica
3	Switch Primary	switch to host2	Replica	Primary	Replica
4	Shun `host1`	datasource host1 shun	Shunned	Primary	Replica
5	Perform maintenance		Shunned	Primary	Replica
6	Recover the Replica (`host1`)	datasource host1 recover	Replica	Primary	Replica

Step	Description	Command	host1	host2	host3
7	Ensure the Replica has caught up		Replica	Primary	Replica
8	Switch Primary back to `host1` (*optional*)	switch to host1	Primary	Replica	Replica
9	Set `AUTOMATIC` policy	set policy automatic	Primary	Replica	Replica

6.16.3. Performing Maintenance on an Entire Dataservice

To perform maintenance on all of the machines within a dataservice, a rolling sequence of maintenance must be performed carefully on each machine in a structured way. In brief, the sequence is as follows

1. Perform maintenance on each of the current Replicas
2. Switch the Primary to one of the already maintained Replicas
3. Perform maintenance on the old Primary (now in Replica state)
4. Switch the old Primary back to be the Primary again

Warning

The "Rolling Maintenance" procedure outlined here should NOT be used when upgrading Tungsten Software between major versions, for example from 6.1 to 7.0, or 7.0 to 7.1.

In most cases the switch will not work due to differences within the manager communications and this could cause unexpected outages.

See Section 6.15, “Upgrading Tungsten Cluster” for more details on upgrading Tungsten software.

A more detailed sequence of steps, including the status of each datasource in the dataservice, and the commands to be performed, is shown in the table below. The table assumes a three-node dataservice (one Primary, two Replicas), but the same principles can be applied to any Primary/Replica dataservice:

Step	Description	Command	host1	host2	host3
1	Initial state		Primary	Replica	Replica
2	Set `MAINTENANCE` policy	set policy maintenance	Primary	Replica	Replica
3	Shun Replica `host2`	datasource host2 shun	Primary	Shunned	Replica
4	Perform maintenance		Primary	Shunned	Replica
5	Recover the Replica `host2` back	datasource host2 recover	Primary	Replica	Replica
6	Ensure the Replica (`host2`) has caught up		Primary	Replica	Replica
7	Shun Replica `host3`	datasource host3 shun	Primary	Replica	Shunned
8	Perform maintenance		Primary	Replica	Shunned
9	Recover Replica `host3` back	datasource host3 recover	Primary	Replica	Replica
10	Ensure the Replica (`host3`) has caught up		Primary	Replica	Replica
11	Switch Primary to `host2`	switch to host2	Replica	Primary	Replica
12	Shun `host1`	datasource host1 shun	Shunned	Primary	Replica
13	Perform maintenance		Shunned	Primary	Replica
14	Recover the Replica `host1` back	datasource host1 recover	Replica	Primary	Replica
15	Ensure the Replica (`host1`) has caught up		Primary	Replica	Replica
16	Switch Primary back to `host1`	switch to host1	Primary	Replica	Replica
17	Set `AUTOMATIC` policy	set policy automatic	Primary	Replica	Replica

6.16.4. Upgrading MySQL

The process for upgrading MySQL should follow the sequences outlined in Section 6.16.2, “Performing Maintenance on a Primary” whereby you would first upgrade the replicas, perform a switch and then upgrade the last remaining node.

In the majority of cases this will be sufficient, but when upgrading between major MySQL releases, there are often changes in behavior with MySQL that could either cause issues with the Tungsten Components, or with your application.

In all cases, it is paramount that your applications have been fully tested against the newer version of MySQL to ensure that your application is not impacted by any behavior changes within the core MySQL engine.

Issues that may arise, and the subsequent workarounds are outlined in the following sections of this document. Where a workaround within Tungsten is not possible, this will be clearly identified.

6.16.4.1. Connector Behavior during upgrades

Important

The issue outlined below only affects connectors running in Proxy Mode, and more imporantly the behavior all depends on when connectors are restarted as part of the process.

When the connectors are running in Proxy Mode, they advertise a cached, static, MySQL version that is retrieved at connector startup from the database, and is assumed by the connectors to be the global version of all database nodes.

In versions prior to 7.2.0, this request was made on a replica (to avoid unneeded load on the primary). With versions 7.2.0 onwards, the request is now done on the primary by default, however this behavior can be changed to the pre-7.2.0 behavior by adding the following property to your configuration:

```
property=readAutodetectConfigFromPrimary=false
```

In situations where a rolling upgrade is being performed, and you have upgraded the replica node to a higher version of MySQL, for example MySQL 8.0, the connectors will either still report MySQL 5.7 as the version that was cached, or if they get restarted they will advertise MySQL 8.0.

If your application relies on reading this version to ensure it behaves accordingly, this advertised version may cause the application to behave in a way that would fail if the advertised version does not match the actual version of the database that is handling a specific request, for example the advertised version could be 8.0 however the request is directed to the primary node which has yet to be upgraded and is in fact running 5.7.

This is most commonly an issue if you are using hibernate.

To workaround this, there are a number of options you could employ:

- Temporarily switch the connectors to run in Bridge Mode. In this mode the connectors do not advertise the version of MySQL and therefore will always be correct and mirror the version of the database that the connection is established too.

  There is a very small window of downtime in this approach at the point at which a switch is issued to move the primary node. This will only be a matter of seconds.

  On all connector nodes, edit your `/etc/tungsten/tungsten.ini` and add (or modify) the `connector-bridge-mode` [480] property, setting the value to `true`, followed by tpm update

  #### Warning

  Do NOT add the `--replace-release` switch to the update to ensure the connectors do not get restarted

- If you have a Composite Active/Passive or Composite Active/Active cluster, and have the ability to redirect traffic easily, simply point all traffic to one cluster whilst upgrading the other, then switch between clusters and redirect traffic. Once both clusters have been upgraded, rebalance traffic as normal. This could also be combined with the above procedure to switch to Bridge Mode if desired.

  High level of this procedure would be as follows:

  1. Direct all traffic to SiteA (Only required for Composite Active/Active clusters)
  2. Upgrade all nodes in SiteB to the new version of MySQL (Including the connectors)
  3. If this is a Composite Active/Passive cluster, issue a switch to promote SiteB to be active and SiteA to be passive.
  4. Direct all traffic to SiteB (Only required for Composite Active/Active clusters)
  5. Upgrade all nodes in SiteA to the new version of MySQL (Including the connectors)
  6. If this is a Composite Active/Passive cluster, and you wish to return SiteA to be active, then issue the switch to promote SiteA to be active and SiteB to be passive.
  7. Rebalance traffic across both clusters (Only required for Composite Active/Active clusters)

- Add the `connector-db-version` [480] property to your `/etc/tungsten/tungsten.ini` on the connectors. This will force the connectors to advertise a specific mysql version.

  This option could still result in the wrong version being advertised to the applications, and therefore should only be used when you are confident that the application will not misbehave when the version does not match. It is not recommended for hibernate users.

6.16.4.2. Upgrading between MySQL 5.x and MySQL 8.x

As is common when providers release new versions of their software, there are often small subtle changes that often go unnoticed, and then there are some changes which can completely break your application. The latter is quite common when moving between major versions of MySQL.

In MySQL 8.x this is no different, and for the most part these changes won't impact Tungsten operations.

To allow, and enable, you to upgrade your underlying database whilst maintaining your application's availability, having your topology running with different versions of MySQL on each node is perfectly normal and supported.

Between versions 5.x and 8.x of MySQL, a number of changes were made to the `SQL_MODE` - some that existed in 5.7 were removed, and some new ones added in MySQL 8, additionally there are some collation (or Character Set) changes. Within your applications, you may not even notice this, but this can cause an issue with replication. Part of the workflow when Tungsten Replicator applies is to ensure the same `SQL_MODE` and collations that were in play when the original transaction was written in the source, are enabled when we write into the target, by extracting this information as part of the metadata. If we try to enable a `SQL_MODE` or enable a collation that doesn't exist, then your replicators will go into an error state, with a message similar to the following:

```
java.sql.SQLSyntaxErrorException: Variable 'sql_mode' can't be »
set to the value of 'NO_AUTO_CREATE_USER'

  or

pendingError : Event application failed: seqno=2915 fragno=0 »
message=Failed to apply session variables

  or

Caused by: java.sql.SQLException: Unknown collation: '255'
```

The `SQL_MODE` exceptions may happen when replicating between two different versions in either direction, i.e. from 5.x to 8.x or vice versa. The collation mapping error is only an issue when replicating down versions, i.e. from 8.x to 5.x

So that you can seamlessly upgrade your underlying MySQL databases, and avoid this particular error, you will need to temporarily enable up to two additional filters and leave running whilst you have a mix of MySQL versions replicating to each other.

When replicating between lower to higher versions (MySQL 5.x to MySQL 8.x), you need to enable the `dropsqlmode` filter using the following syntax and then by running tpm update:

```
svc-applier-filters=dropsqlmode
```

When replicating between higher to lower versions (MySQL 8.x to MySQL 5.x), you need to enable both the `dropsqlmode` filter and the `mapcharset` filter, additionally you will need to configure the `dropsqlmode` filter differently, the following syntax can be used:

```
svc-applier-filters=dropsqlmode,mapcharset
property=replicator.filter.dropsqlmode.modes=TIME_TRUNCATE_FRACTIONAL
```

Once all nodes have been upgraded, simply reverse the process by removing the syntax from the configuration and re-running tpm update.

Important

The `mapcharset` MUST be removed when replicating between the same MySQL versions

For more information on the filters mentioned, see Section 12.3.17, "`dropsqlmode.js` Filter" and Section 12.3.27, "mapcharset Filter"

6.16.5. Upgrading or Updating your JVM

When upgrading your JVM version or installation, care should be taken as changing the JVM will momentarily remove and replace required libraries and components which may upset the operation of Tungsten Cluster while the upgrade or update takes place.

For this reason, JVM updates or changes must be treated as an OS upgrade or event, requiring a Primary switch and controlled stopping/shunning of services during the update process.

A sample sequence for this in a 3-node cluster is described below:

Step	Description	Command	host1	host2	host3
1	Initial state		Primary	Replica	Replica

Step	Description	Command	host1	host2	host3
2	Shun Replica `host2`	datasource host2 shun	Primary	Shunned	Replica
3	Stop all services on `host2`.	stopall	Primary	Stopped	Replica
4	Update the JVM		Primary	Stopped	Replica
5	Start all services on `host2` Replica.	startall	Primary	Replica	Replica
6	Recover Replica back	datasource host2 recover	Primary	Replica	Replica
7	Ensure the Replica (`host2`) has caught up	ls	Primary	Replica	Replica
8	Shun Replica `host3`	datasource host3 shun	Primary	Replica	Shunned
9	Stop all services on `host3`.	stopall	Primary	Replica	Stopped
10	Update the JVM		Primary	Replica	Stopped
11	Start all services on `host3` Replica.	startall	Primary	Replica	Replica
12	Recover Replica back	datasource host3 recover	Primary	Replica	Replica
13	Ensure the Replica (`host3`) has caught up	ls	Primary	Replica	Replica
14	Switch Primary to `host2`	switch to host2	Replica	Primary	Replica
15	Shun `host1`	datasource host1 shun	Shunned	Primary	Replica
16	Stop all services on `host1`.	stopall	Stopped	Primary	Replica
17	Update the JVM		Stopped	Primary	Replica
18	Start all services on `host1` Replica.	startall	Replica	Primary	Replica
19	Recover `host1` back	datasource host1 recover	Replica	Primary	Replica
20	Ensure the Replica (`host1`) has caught up	ls	Replica	Primary	Replica
21	Switch Primary back to `host1`	switch to host1	Primary	Replica	Replica

6.16.6. Making 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 schema changes, such as the addition, removal or modification of an existing table definition will be correctly replicated to Replicas, other operations, such as creating new indexes, adding/removing columns or migrating table data between table definitions, is best performed individually on each dataserver while it has been temporarily taken out of the dataservice.

As with all maintenance operations it is advisable to have fully tested your DDL in a staging environment. In some cases, the imapct of DDL change is minimal and therefore can safely applied to the Primary node and allowing the change to be replicated down to the Replicas.

In situations where the overhead of the DDL change would cause an outage to your application through table locking, use the rolling maintenance procedure below which is specific for DDL changes.

The basic process comprises of a number of steps, these are as follows:

1. If the DDL adds or removes columns, then enable the `colnames` and `dropcolumn` filters
2. If the DDL adds or removes tables, and you do not want to simply apply to the Primary and allow replication to handle it, then enable the `replicate` filters
3. Perform schema changes following the process summarised in the table below
4. Optionally, remove the filters enabled in the first step

> **Note**
>
> The first two steps listed above are optional for Multi-Site/Active-Active topologies, the process will still work without this step however this may cause cross-site replication to go into an error state until the DDL changes have been applied to all hosts.
>
> Therefore, within Multi-Site/Active-Active environments, instead of using the `colnames` and `dropcolumn` filters mentioned above, you may choose to temporarily stop the cross site replicators. Doing so will reduce the risk of errors in the replication flow whilst the maintenance is in progress, however you will introduce latency between clusters for the duration of the time the replicators are offline.

> For Composite Active/Active topologies, it is not possible to isolate the cross-site replictions in the same way, therefore these steps are strongly advised as the resulting error state of the cross-site replicator may prevent a successful switch between Primary and Replica nodes.

Enable filters for column changes

The use of the `colnames` and `dropcolumn` filters allow you to make changes to the structure of tables, without impacting the flow of replication.

> **Important**
>
> During these schema changes, and whilst the filters are in place, applications *MUST* be forwards and backwards compatible, but *MUST NOT* alter data within any columns that are filtered out from replication until after the process has been completed on all hosts, and the filters disabled. Data changes to filtered columns will cause data drift and inconsistencies, resulting in potentially unexpected behaviour

- To enable the filters, first create a file called `schemachange.json` in a directory accessible by the OS user that the software is running as.

  Typically, this will be the `tungsten` user and the default location for this file will be `/opt/continuent/share`

  The file needs to contain a JSON block outlining ALL the columns being added and removed from all tables affected by the changes.

  In the example below, we are removing the column, `operator_code` and adding `operator_desc` to the `system_operators` table and adding the column `action_date` to the `system_actions` table:

```
[
    {
      "schema": "ops",
      "table": "system_operators",
      "columns": ["operator_code","operator_desc"]
    },
    {
      "schema": "ops",
      "table": "system_actions",
      "columns": ["action_date"]
    }
  ]
```

- Place your cluster into *`MAINTENANCE`* mode

```
shell> cctrl
cctrl> set policy maintenance
```

> **Note**
>
> If running a Composite Active/Passive or Composite Active/Active topology, issue the command at the top global level to place all clusters in *`MAINTENANCE`*, or execute individually within each cluster

- Next, enable the filters within your configuration by adding the following two parameters to the `tungsten.ini` (if running in INI method) to the [defaults] section on EVERY cluster node:

```
svc-extractor-filters=colnames,dropcolumn
property=replicator.filter.dropcolumn.definitionsFile=/opt/continuent/share/schemachange.json
```

  Followed by tpm update to apply the changes:

- Or, if running as a staging install:

```
shell> cd staging_dir
shell> tools/tpm update alpha \
  --svc-extractor-filters=colnames,dropcolumn \
  --property=replicator.filter.dropcolumn.definitionsFile=/opt/continuent/share/schemachange.json
```

- Monitor replication to ensure there are no errors before continuing

Enable filters for adding/removing tables

The optional use of the `replicate` filter allows you to add and remove tables without impacting the flow of replication.

> **Important**
>
> During these schema changes, and whilst the filter is in place, applications *MUST* be forwards and backwards compatible, but *MUST NOT* modify data in any new tables until after the process has been completed on all hosts, and the filters disabled. Data changes to filtered tables will cause data drift and inconsistencies, resulting in potentially unexpected behaviour.

- Place your cluster into *`MAINTENANCE`* mode.

```
shell> cctrl
cctrl> set policy maintenance
```

> **Note**
>
> If running a Composite Active/Active or Composite Active/Passive topology, issue the command at the top global level to place all clusters in *MAINTENANCE*, or individually within each cluster.

- Next, enable the filters within your configuration by adding the following two parameters to the `tungsten.ini` (if running in INI method) to the `[defaults]` section on EVERY cluster node.

  In this example we plan to ADD the table `system_actions` and REMOVE the table `system_operations`, both within the `ops` schema:

  ```
  svc-extractor-filters=replicate
  property=replicator.filter.replicate.ignore=ops.system_actions,ops.system_operations
  ```

  Followed by tpm update to apply the changes

- Or, if running as a staging install:

  ```
  shell> cd staging_dir
  shell> tools/tpm update alpha \
    --svc-extractor-filters=replicate \
    --property=replicator.filter.replicate.ignore=ops.system_actions,ops.system_operations
  ```

- Monitor replication to ensure there are no errors before continuing.

Apply DDL Changes

- Follow the steps outlined in the table below to make the DDL changes to all nodes, in all clusters.
- If filtering columns, once all the changes have been complete, edit the `schemachange.json` to contain an empty document:

  ```
  shell> echo "[]" > /opt/continuent/share/schemachange.json
  ```

  Then, restart the replicators:

  ```
  shell> replicator restart
  ```

- If filtering tables, repeat the process of adding the replicate filter removing any tables from the ignore parameter that you have ADDED to your database.
- You can optionally fully remove the filters if you wish by removing the entries from the configuration and re-running tpm update however it is also perfectly fine to leave them in place. There is a potentially small CPU overhead in very busy clusters by having the filters in place, but otherwise should not have any impact.

  It is advisable to monitor the system usage and make the decision based on your own business needs.

> **Note**
>
> Leaving the `colnames` filter in place will increase the size of THL on disk, therefore if disk space is limited in your environment, it would be avisable to remove these filters.

The following method assumes a schema update on the entire dataservice by modifying the schema on the Replicas first. The schema shows three datasources being updated in sequence, Replicas first, then the Primary.

Step	Description	Command	host1	host2	host3
1	Initial state		Primary	Replica	Replica
2	Set the Replica `host2` offline	trepctl -host host2 offline	Primary	Offline	Replica
3	Connect to dataserver for `host2` and update schema	set sql_log_bin=0; run ddl statements	Primary	Offline	Replica
4	Set the Replica online	trepctl -host host2 online	Primary	Replica	Replica
5	Ensure the Replica (`host2`) has caught up	trepctl -host host2 status	Primary	Replica	Replica
6	Set the Replica `host3` offline	trepctl -host host3 offline	Primary	Replica	Offline
7	Connect to dataserver for `host3` and update schema	set sql_log_bin=0; run ddl statements	Primary	Replica	Offline

Step	Description	Command	host1	host2	host3
8	Set the Replica (`host3`) online	trepctl -host host3 online	Primary	Replica	Replica
9	Ensure the Replica (`host3`) has caught up	trepctl -host host3 status	Primary	Replica	Replica
10	Switch Primary to `host2`	See Section 6.5, "Switching Primary Hosts"	Replica	Primary	Replica
11	Set the Replica `host1` offline	trepctl -host host1 offline	Offline	Primary	Replica
12	Connect to dataserver for `host1` and update schema	set sql_log_bin=0; run ddl statements	Offline	Primary	Replica
13	Set the Replica `host1` online	trepctl -host host1 online	Replica	Primary	Replica
14	Ensure the Replica (`host1`) has caught up	trepctl -host host1 status	Primary	Replica	Replica
15	Switch Primary back to `host1`	See Section 6.5, "Switching Primary Hosts"	Primary	Replica	Replica

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.

6.16.7. Updating Database User Passwords

Rotating user passwords is a common procedure that all businesses follow on a regular basis. This should be no different when Tungsten Clustering is in use however the process needs to be handled with care to ensure uninterupted operations.

The procedure below provides the basic steps required to change the database user account associated with Tungsten. This would be the user that is configured by the `replication-user` [510] tpm option that is used by the Tungsten processes.

1. The first step is to place the cluster into *`MAINTENANCE`* mode:

```
shell> cctrl
cctrl> set policy maintenance
```

2. On the primary node, connect to mysql as the root user and issue the password change (This example assumes the DB user is called `tungsten`:

```
shell> mysql -u root -p
mysql> set password for tungsten@'%' = password('new_pass');
mysql> flush privileges
```

3. On all remaining nodes, connect to mysql as the root user and issue the following:

```
shell> mysql -u root -p
mysql> flush privileges
```

4. At this point, tungsten services will now show as offline, however because we are in *`MAINTENANCE`*, applications will still be connected.

   To bring process online with the new password, we now need to update the `/etc/tungsten/tungsten.ini` and update the password accordingly for the `replication-user` [510] entry. After updating this on every host, issue the following, also on every host:

```
shell> tpm update
```

5. Return the cluster to *`AUTOMATIC`* mode:

```
shell> cctrl
cctrl> set policy automatic
```

Using MySQL 8.0.14+?

If you are running version 8.0.14 or later of MySQL, you may want to consdier making use of a new MySQL feature that allows user accounts to have dual passwords. Doing so would eliminate the offline state of Tungsten components in the above example. This process would look like the following:

1. The first step is to place the cluster into *`MAINTENANCE`* mode:

```
shell> cctrl
cctrl> set policy maintenance
```

2. On the primary node, connect to mysql as the root user and issue the DDL below (This example assumes the DB user is called `tungsten`:

```
shell> mysql -u root -p
mysql> ALTER USER tungsten@'%' IDENTIFIED BY 'second_pass' RETAIN CURRENT PASSWORD;
mysql> flush privileges
```

3. On all remaining nodes, connect to mysql as the root user and issue the following:

```
shell> mysql -u root -p
mysql> flush privileges
```

4. We now need to update the `/etc/tungsten/tungsten.ini` and update the password accordingly for the `replication-user` [510] entry. After updating this on every host, issue the following, also on every host:

```
shell> tpm update
```

5. Return the cluster to *`AUTOMATIC`* mode:

```
shell> cctrl
cctrl> set policy automatic
```

6. When all systems that utilise the account have been updated to use the new password, issue the following:

```
shell> mysql -u root -p
mysql> ALTER USER tungsten@'%' DISCARD OLD PASSWORD;
mysql> flush privileges
```

Updating passwords for Connectors and Applications

It is not possible to update application user passwords without incurring downtime, the same applies if you are updating the user associated with the `application-user` [474] tpm option that the connectors user.

To change these user accounts the process would be as follows:

1. On the primary node, connect to mysql as the root user and issue the password change

```
shell> mysql -u root -p
mysql> set password for app_user@'%' = password('new_pass');
mysql> flush privileges
```

2. On all remaining nodes, connect to mysql as the root user and issue the following:

```
shell> mysql -u root -p
mysql> flush privileges
```

3. At this point, applications and/or connectors will be offline.

   To bring connectors online with the new password, we need to update the `/etc/tungsten/tungsten.ini` and update the password accordingly for the `application-user` [474] entry. After updating this on every connector host, issue the following, also on every connector host:

```
shell> tpm update
```

4. If you are using Proxy mode, you will also need to update the password for application users within the `user.map` file.

   After updating the file, the connector will automatically detect a change and reload.

As with the main Tungsten account, if you are running MySQL 8.0.14+ you can also make the process simpler with minimal to zero downtime by utilising the DUAL password feature. For application and connector users, this process is expalined in Section 7.6.2, “Dual-Paswords and MySQL 8”

6.17. Monitoring Tungsten Cluster

It is your responsibility to properly monitor your deployments of Tungsten Cluster and Tungsten Replicator. The minimum level of monitoring must be done at three levels. Additional monitors may be run depending on your environment but these three are required in order to ensure availability and uptime.

1. Make sure the appropriate Tungsten Cluster and Tungsten Replicator services are running.
2. Make sure all datasources and replication services are `ONLINE`.
3. Make sure replication latency is within an acceptable range.

> **Important**
>
> Special consideration must be taken if you have multiple installations on a single server. That applies for clustering and replication or multiple replicators.

These three points must be checked for all directories where Tungsten Cluster or Tungsten Replicator are installed. In addition, all servers should be monitored for basic health of the processors, disk and network. Proper alerting and graphing will prevent many issues that will cause system failures.

6.17.1. Managing Log Files with logrotate

You can manage the logs generated using logrotate.

- Connector Log File: `connector.log`

```
/opt/continuent/tungsten/tungsten-connector/log/connector.log {
        notifempty
        daily
        rotate 3
        missingok
        compress
        copytruncate
}
```

- Manager Log File: `tmsvc.log`

```
/opt/continuent/tungsten/tungsten-manager/log/tmsvc.log {
        notifempty
        daily
        rotate 3
        missingok
        compress
        copytruncate
}
```

- Replicator Log File: `trepsvc.log`

```
/opt/continuent/tungsten/tungsten-replicator/log/trepsvc.log {
        notifempty
        daily
        rotate 3
        missingok
        compress
        copytruncate
}
```

6.17.2. Monitoring Status Using cacti

Graphing Tungsten Replicator data is supported through Cacti extensions. These provide information gathering for the following data points:

- Applied Latency
- Sequence Number (Events applied)
- Status (Online, Offline, Error, or Other)

To configure the Cacti services:

1. Download both files from https://github.com/continuent/monitoring/tree/master/cacti
2. Place the PHP script into `/usr/share/cacti/scripts`.
3. Modify the installed PHP file with the appropriate `$ssh_user` and `$tungsten_home` location from your installation:
   - `$ssh_user` should match the `user` used during installation.
   - `$tungsten_home` is the installation directory and the `tungsten` subdirectory. For example, if you have installed into `/opt/continuent`, use `/opt/continuent/tungsten`.

   Add SSH arguments to specify the correct `id_rsa` file if needed.
4. Ensure that the configured `$ssh_user` has the correct SSH authorized keys to login to the server or servers being monitored. The user must also have the correct permissions and rights to write to the cache directory.
5. Test the script by running it by hand:

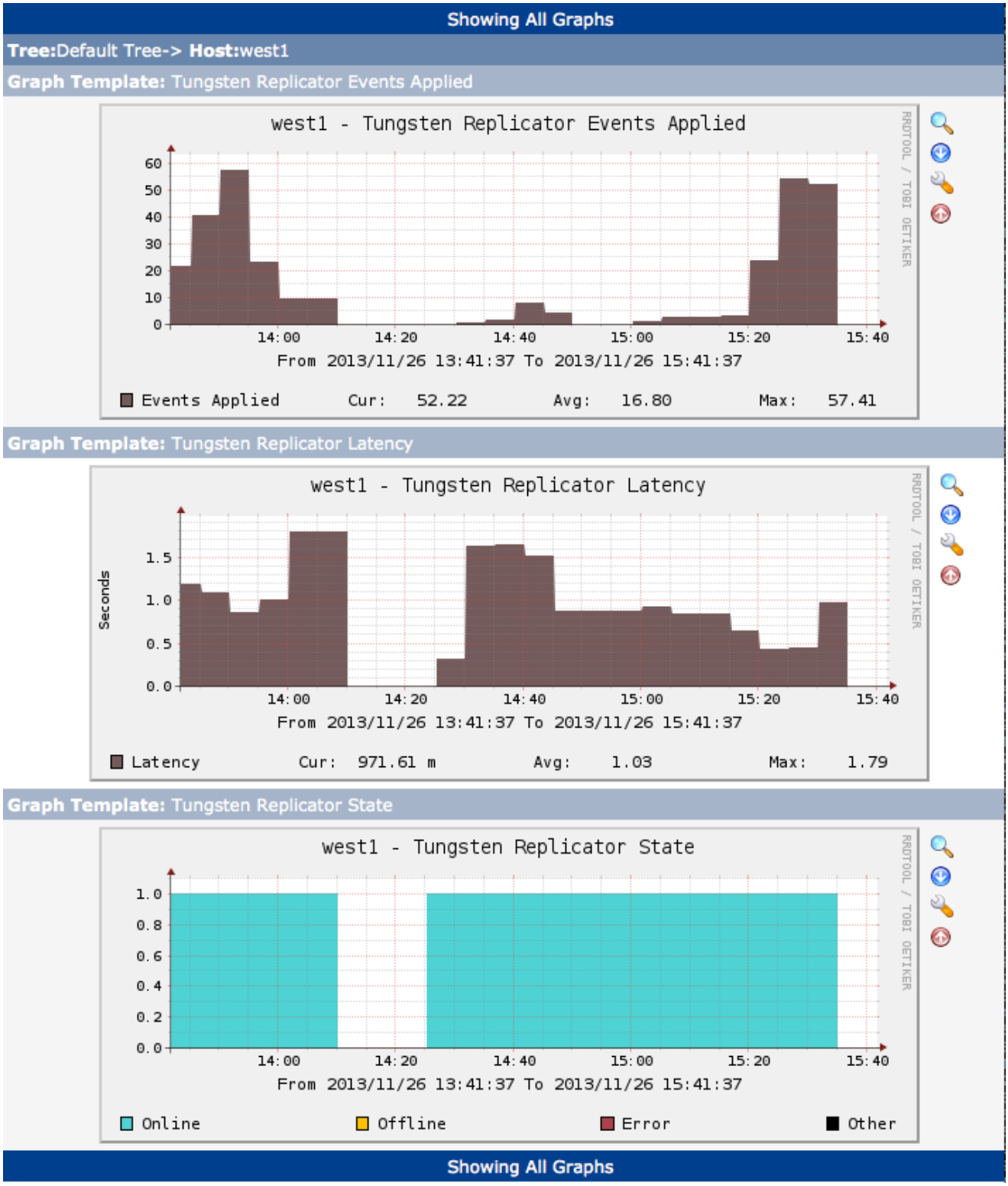
   ```
   shell> php -q /usr/share/cacti/scripts/get_replicator_stats.php --hostname replserver
   ```

   If you are using multiple replication services, add `--service servicename` to the command.
6. Import the XML file as a Cacti template.

7. Add the desired graphs to your servers running Tungsten Cluster. If you are using multiple replications services, you'll need to specify the desired service to graph. A graph must be added for each individual replication service.

Once configured, graphs can be used to display the activity and availability.

Figure 6.2. Cacti Monitoring: Example Graphs



6.17.3. Monitoring Status Using nagios

Integration with Nagios is supported through a number of scripts that output information in a format compatible with the Nagios NRPE plugin. Using the plugin the check commands, such as check_tungsten_latency can be executed and the output parsed for status information.

The available commands are:

- check_tungsten_latency
- check_tungsten_online
- check_tungsten_policy
- check_tungsten_progress
- check_tungsten_services

To configure the scripts to be executed through NRPE:

1. Install the Nagios NRPE server.
2. Start the NRPE daemon:

   ```
   shell> sudo /etc/init.d/nagios-nrpe-server start
   ```

3. Add the IP of your Nagios server to the `/etc/nagios/nrpe.cfg` configuration file. For example:

   ```
   allowed_hosts=127.0.0.1,192.168.2.20
   ```

4. Add the Tungsten check commands that you want to execute to the `/etc/nagios/nrpe.cfg` configuration file. For example:

   ```
   command[check_tungsten_online]=/opt/continuent/tungsten/cluster-home/bin/check_tungsten_online
   ```

5. Restart the NRPE service:

   ```
   shell> sudo /etc/init.d/nagios-nrpe-server start
   ```

6. If the commands need to be executed with superuser privileges, the `/etc/sudo` or `/etc/sudoers` file must be updated to enable the commands to be executed as root through sudo as the `nagios` user. This can be achieved by updating the configuration file, usually performed by using the visudo command:

   ```
   nagios          ALL=(tungsten)  NOPASSWD: /opt/continuent/tungsten/cluster-home/bin/check*
   ```

   In addition, the sudo command should be added to the Tungsten check commands within the Nagios `nrpe.cfg`, for example:

   ```
   command[check_tungsten_online]=/usr/bin/sudo -u tungsten /opt/continuent/tungsten/cluster-home/bin/check_tungsten_online
   ```

   Restart the NRPE service for these changes to take effect.

7. Add an entry to your Nagios `services.cfg` file for each service you want to monitor:

   ```
   define service {
           host_name database
           service_description     check_tungsten_online
           check_command           check_nrpe! -H $HOSTADDRESS$  -t 30 -c check_tungsten_online
           retry_check_interval    1
           check_period            24x7
           max_check_attempts      3
           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.

6.17.4. Monitoring Status Using Prometheus Exporters

Continuent has introduced basic support for using Prometheus and Grafana to monitor Tungsten nodes. As of Tungsten software v6.1.4, five key Prometheus exporters have been added to the distribution.

6.17.4.1. Monitoring Status with Exporters Overview

The exporters will allow a Prometheus server to gather metrics for:

- The underlying node "hardware" using an external `node_exporter` binary added to the distribution.
- The running MySQL server using an external `mysqld_exporter` binary added to the distribution.
- The Tungsten Replicator using new built-in functionality in the replicator binary.
- The Tungsten Manager using new built-in functionality in the manager binary.
- The Tungsten Connector using new built-in functionality in the connector binary.

A new command, tmonitor, has been included to assist with the management and testing of the exporters.

To manually test any exporter, get the URL `http://{theHost}:{thePort}/metrics` either via the curl command or a browser.

```
shell> curl -s 'http://localhost:9100/metrics' | wc -l
916

shell> curl -s 'http://localhost:9100/metrics' | head -10
# HELP go_gc_duration_seconds A summary of the GC invocation durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 1.0129e-05
go_gc_duration_seconds{quantile="0.25"} 1.3347e-05
go_gc_duration_seconds{quantile="0.5"} 1.8895e-05
go_gc_duration_seconds{quantile="0.75"} 3.095e-05
go_gc_duration_seconds{quantile="1"} 0.00104028
go_gc_duration_seconds_sum 10.43582891
go_gc_duration_seconds_count 173258
# HELP go_goroutines Number of goroutines that currently exist.
```

The below table describes the exporters and the ports they listen on.

Exporter	Port	Description	Scope
node	9100	Metrics for the underlying node "hardware"	External
mysql	9104	Metrics for the MySQL server	External
replicator	8091	Metrics for the Tungsten Replicator	Internal
manager	8092	Metrics for the Tungsten Manager	Internal
connector	8093	Metrics for the Tungsten Connector	Internal

6.17.4.2. Customizing the Prometheus Exporter Configuration

If the default ports [`8091`, `8092` & `8093`] are in conflict, it is possible to change them.

Add one line per component to `/etc/tungsten/tungsten.ini` under the `[defaults]` section:

```
shell> vi /etc/tungsten/tungsten.ini
[defaults]
property=replicator.prometheus.exporter.port=28091
property=manager.prometheus.exporter.port=28092
property=connector.prometheus.exporter.port=28093
...
```

Warning

Either SHUN the individual nodes one at a time and WELCOME each one after running the update, or set policy to *MAINTENANCE* via cctrl and update all nodes, then set the policy back to *AUTOMATIC* when all nodes have been completed.

Update the configuration:

```
shell> tpm update
```

Important

You may need to restart the Manager, Replicator or Connector, depending on what was changed.

Verify that the port is listening:

```
shell> sudo netstat -pan | grep 28091
```

6.17.4.3. Disabling the Prometheus Exporters

Add the following to `/etc/tungsten/tungsten.ini` under the `[defaults]` section:

The Prometheus exporters are all enabled by default. It is simple to disable them using the following procedure.

Add one line per component to `/etc/tungsten/tungsten.ini` under the `[defaults]` section:

```
shell> vi /etc/tungsten/tungsten.ini
[defaults]
property=manager.prometheus.exporter.enabled=false
property=replicator.prometheus.exporter.enabled=false
property=connector.prometheus.exporter.enabled=false
...
```

Warning

Either SHUN the individual nodes one at a time and WELCOME each one after running the update, or set policy to *MAINTENANCE* via cctrl and update all nodes, then set the policy back to *AUTOMATIC* when all nodes have been completed.

Update the configuration:

```
shell> tpm update
```

Important

You may need to restart the Manager, Replicator or Connector, depending on what was changed.

Verify that the port is no longer listening:

```
shell> sudo netstat -pan | grep 8091
```

6.17.4.4. Managing and Testing Exporters Using the tmonitor Command

tmonitor is a simple tool for the management and testing of Prometheus exporters.

Exporters that require an external binary to function (i.e. node_exporter and mysqld_exporter) are considered to have an External Scope.

Exporters that do not require an external binary to function (i.e. manager, replicator and connector) are considered to have an Internal Scope.

By default, tmonitor {action} will act upon all available exporters.

If any exporter is specified on the CLI, then only those listed on the CLI will be acted upon.

The curl command must be available in the PATH for the `status` and `test` actions to function.

tmonitor status will use curl to test the exporter on `localhost`.

tmonitor test will use curl to fetch and print the metrics from one or more exporters on `localhost`.

tmonitor install will configure the specified exporter (or all external exporters if none is specified) to start at boot.

tmonitor remove will stop the specified exporter (or all external exporters if none is specified) from starting at boot.

Both the `install` and `remove` actions will attempt to auto-detect the boot sub-system. Currently, `init.d` and `systemd` are supported.

```
shell> tmonitor help
```

```
...
>>> Usage:

   tmonitor [args] {action}

   = Actions available for all exporter scopes:

      status - validate the service via curl (short output)
      test - validate the service via curl (full output)

   = Actions available for External-scope exporters only:

      start - launch the exporter process
      stop - kill the exporter process

      install - configure the exporter to start at boot
      remove - stop the exporter from starting at boot


>>> Arguments:

   [-h|--help]
   [-v|--verbose]
   [--force] Required for certain MySQL-specific operations

   [-f|--filter {string}] Limit the `tmonitor test` output based on this string match
   [-t|--tungsten] Set the `tmonitor test` filter to 'tungsten_'

   [-i|--internal] Only act upon exporters with an internal scope
   [-e|-x|--external] Only act upon exporters with an external scope

   --internal and --external may not be specified together.

   = Internal Scope Exporters:

     [-C|--connector] Specify the Tungsten Connector exporter
     [-M|--manager] Specify the Tungsten Manager exporter
     [-R--replicator] Specify the Tungsten Replicator exporter

   = External Scope Exporters:

     [-m|--mysql|--mysqld] Specify the MySQL exporter
     [-n|--node] Specify the Node exporter
```

Example: View the status of all exporters:

```
shell> tmonitor status
Tungsten Connector exporter running ok on port 8093
Tungsten Manager exporter running ok on port 8092
MySQL exporter running ok on port 9104
Node exporter running ok on port 9100
Tungsten Replicator exporter running ok on port 8091
All 5 exporters are running ok (Up: Tungsten Connector, Tungsten Manager, MySQL, Node, Tungsten Replicator)
```

Example: Start all exporters:

```
shell> tmonitor start
tungsten@db1-demo:/home/tungsten # tmonitor start
Node exporter started successfully on port 9100.
MySQL exporter started successfully on port 9104.

shell> tmonitor start
The Node exporter is already running
The MySQL exporter is already running
```

Example: Test all exporters:

```
shell> tmonitor test | wc -l
3097

shell> tmonitor test | grep '== '
== Metrics for the connector exporter:
== Metrics for the manager exporter:
== Metrics for the mysql exporter:
== Metrics for the node exporter:
== Metrics for the replicator exporter:

shell> tmonitor test | less
```

Example: Stop all exporters:

```
shell> tmonitor stop
All exporters stopped
```

Example: View the status of all exporters filtered by scope:

```
shell> tmonitor status -i
Tungsten Connector exporter running ok on port 8093
Tungsten Manager exporter running ok on port 8092
Tungsten Replicator exporter running ok on port 8091
All 3 internal exporters are running ok (Up: Tungsten Connector, Tungsten Manager, Tungsten Replicator)

shell> tmonitor status -x
MySQL exporter running ok on port 9104
Node exporter running ok on port 9100
All 2 external exporters are running ok (Up: MySQL, Node)
```

Example: Install init.d boot scripts for all external exporters:

```
shell> tmonitor install
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:

sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install


shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install
node_exporter init.d boot script installed and activated

Use either `sudo service node_exporter start` or `tmonitor --node start` now to start the Node exporter.

mysqld_exporter init.d boot script installed and activated

Use either `sudo service mysqld_exporter start` or `tmonitor --mysql start` now to start the MySQL exporter.
```

Example: Install systemd boot scripts for all external exporters:

```
shell> tmonitor install
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:

sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install


shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install
Created symlink from /etc/systemd/system/multi-user.target.wants/node_exporter.service to /etc/systemd/system/node_exporter.service.
node_exporter systemd boot script installed and enabled

Use either `sudo systemctl start node_exporter` or `tmonitor --node start` now to start the Node exporter.

Created symlink from /etc/systemd/system/multi-user.target.wants/mysqld_exporter.service to /etc/systemd/system/mysqld_exporter.service.
mysqld_exporter systemd boot script installed and enabled

Use either `sudo systemctl start mysqld_exporter` or `tmonitor --mysql start` now to start the MySQL exporter.
```

Example: Remove init.d boot scripts for all external exporters:

```
shell> tmonitor remove
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:

sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove

shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove
node_exporter is still running, unable to remove. Please run either `tmonitor --node stop` or `sudo service node_exporter stop`, then retry this operation

shell> tmonitor stop
Node exporter stopped
MySQL exporter stopped

shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove
node_exporter init.d boot script de-activated and removed
mysqld_exporter init.d boot script de-activated and removed
```

Example: Remove systemd boot scripts for all external exporters:

```
shell> tmonitor stop
Node exporter stopped
MySQL exporter stopped

shell> tmonitor remove
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:
```

```
sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove

shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove
node_exporter systemd unit boot script disabled and removed
mysqld_exporter systemd unit boot script disabled and removed
```

The tmonitor command is located in the `$CONTINUENT_ROOT/tungsten/cluster-home/bin` directory.

> **Note**
>
> The tmonitor command will only be available in the `PATH` if the Tungsten software has been installed with the configuration option `profile-script` [507] included.

6.17.4.5. Monitoring Node Status Using the External node_exporter

The node_exporter will respond to requests on port `9100`, path `/metrics`

The tmonitor command is the best way to manage the node_exporter binary.

For example, to test a running node_exporter service:

```
shell> tmonitor --node test | wc -l
869

shell> tmonitor --node test | head -10
=============================================================================================
== Metrics for the node exporter:
=============================================================================================
# HELP go_gc_duration_seconds A summary of the GC invocation durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 1.0214e-05
go_gc_duration_seconds{quantile="0.25"} 1.3524e-05
go_gc_duration_seconds{quantile="0.5"} 2.2065e-05
go_gc_duration_seconds{quantile="0.75"} 4.1943e-05
go_gc_duration_seconds{quantile="1"} 0.003692845
```

To start the node_exporter binary (only), and then get the status:

```
shell> tmonitor start --node
shell> tmonitor status --node
Node exporter running ok
```

> **Warning**
>
> The node_exporter command is not included in the `PATH` unless you add it manually.

The node_exporter binary is located in the `$CONTINUENT_ROOT/tungsten/cluster-home/prometheus` directory.

6.17.4.6. Monitoring MySQL Server Status Using the External mysqld_exporter

The mysqld_exporter will respond to requests on port `9104`, path `/metrics`

The mysqld_exporter command will read MySQL server connection information from the `~/.my.cnf` file. Since the default listener port for the MySQL server is `13306`, the file must contain at least:

```
shell> cat ~/.my.cnf
[client]
port=13306
user={tungsten_database_user_here}
password={tungsten_database_password_here}
```

The tmonitor command is the best way to manage the mysqld_exporter binary.

For example, to test a running mysqld_exporter service:

```
shell> tmonitor --mysqld test | wc -l
1813

shell> tmonitor --mysqld test | head -10
=============================================================================================
== Metrics for the mysql exporter:
=============================================================================================
# HELP go_gc_duration_seconds A summary of the GC invocation durations.
 # TYPE go_gc_duration_seconds summary
 go_gc_duration_seconds{quantile="0"} 1.0051e-05
 go_gc_duration_seconds{quantile="0.25"} 1.4436e-05
```

```
 go_gc_duration_seconds{quantile="0.5"} 3.3925e-05
 go_gc_duration_seconds{quantile="0.75"} 6.3136e-05
 go_gc_duration_seconds{quantile="1"} 0.000551545
```

To start the mysqld_exporter binary (only), and then get the status:

```
shell> tmonitor start --mysqld
shell> tmonitor status --mysqld
mysqld exporter running ok
```

> **Warning**
>
> The mysqld_exporter command is not included in the `PATH` unless you add it manually.

The mysqld_exporter binary is located in the `$CONTINUENT_ROOT/tungsten/cluster-home/prometheus` directory.

6.17.4.7. Monitoring Tungsten Replicator Status Using the Built-In Exporter

The replicator will respond to requests on port `8093`, path `/metrics`

The tmonitor command is the best way to test the replicator exporter.

For example, to test a running replicator exporter service:

```
shell> tmonitor --replicator test | wc -l
148

shell> tmonitor -t -R test
=================================================================================================
== Metrics for the replicator exporter:
=================================================================================================
# HELP tungsten_replicator_version The Tungsten Clustering software version number
# TYPE tungsten_replicator_version gauge
tungsten_replicator_version{version="Tungsten Clustering 7.0.0 build 478",vendor="Continuent",name="Tungsten Replicator",} 1.0
# HELP tungsten_replicator_service Replicator service value
# TYPE tungsten_replicator_service gauge
tungsten_replicator_service{service="east",role="master",state="online",} 1.0
tungsten_replicator_service{service="east_from_west",role="relay",state="online",} 1.0
# HELP tungsten_replicator_seqno Replicator min/max/current sequence number value
# TYPE tungsten_replicator_seqno gauge
tungsten_replicator_seqno{service="east",seqno="minimum",} 0.0
tungsten_replicator_seqno{service="east",seqno="maximum",} 741693.0
tungsten_replicator_seqno{service="east",seqno="current",} 741693.0
tungsten_replicator_seqno{service="east_from_west",seqno="minimum",} 0.0
tungsten_replicator_seqno{service="east_from_west",seqno="maximum",} 638682.0
tungsten_replicator_seqno{service="east_from_west",seqno="current",} 638682.0
# HELP tungsten_replicator_latency Replicator applied/relative latency value
# TYPE tungsten_replicator_latency gauge
tungsten_replicator_latency{service="east",latency="applied",} 0.754
tungsten_replicator_latency{service="east",latency="relative",} 0.762
tungsten_replicator_latency{service="east_from_west",latency="applied",} 0.738
tungsten_replicator_latency{service="east_from_west",latency="relative",} 0.763
```

Metrics for Tungsten Replicator

Metric	Description	
`tungsten_replicator_version`	The Tungsten software version number	
`tungsten_replicator_service`	Replicator service value	
`tungsten_replicator_seqno`	Replicator min/max/current sequence number value	
`tungsten_replicator_latency`	Replicator applied/relative latency value	

6.17.4.8. Monitoring Tungsten Manager Status Using the Built-In Exporter

The manager will respond to requests on port `8091`, path `/metrics`

The tmonitor command is the best way to test the manager exporter.

For example, to test a running manager exporter service:

```
shell> tmonitor --manager test | wc -l
146

shell> tmonitor -t --manager test
=================================================================================================
== Metrics for the manager exporter:
```

```
================================================================================================
# HELP tungsten_manager_version The Tungsten Clustering software version number
# TYPE tungsten_manager_version gauge
tungsten_manager_version{version="Tungsten Clustering 7.0.0 build 478",vendor="Continuent",name="Tungsten Manager",} 1.0
# HELP tungsten_manager_service The service and datasource served by this Tungsten Manager
# TYPE tungsten_manager_service gauge
tungsten_manager_service{service="east",datasource="db1-demo.continuent.com",role="master",state="online",} 1.0
# HELP tungsten_manager_policy The current Tungsten Manager policy mode
# TYPE tungsten_manager_policy gauge
tungsten_manager_policy{policy="MAINTENANCE",} 0.0
# HELP tungsten_manager_connections Connector active/total connections value
# TYPE tungsten_manager_connections gauge
tungsten_manager_connections{connections="active",} 1.0
tungsten_manager_connections{connections="total",} 149.0
```

6.17.4.9. Monitoring Tungsten Connector Status Using the Built-In Exporter

The connector will respond to requests on port `8092`, path `/metrics`

The tmonitor command is the best way to test the connector exporter.

For example, to test a running connector exporter service:

```
shell> tmonitor --connector test | wc -l
136

shell> tmonitor -t -C test
================================================================================================
== Metrics for the connector exporter:
================================================================================================
# HELP tungsten_connector_version The Tungsten Clustering software version number
# TYPE tungsten_connector_version gauge
tungsten_connector_version{version="Tungsten Clustering 7.0.0 build 478",vendor="Continuent",name="Tungsten Connector",} 1.0
# HELP tungsten_connector_connections Connector active/total connections value
# TYPE tungsten_connector_connections gauge
tungsten_connector_connections{dataservice="east",datasource="db1-demo.continuent.com",connections="active",} 0.0
tungsten_connector_connections{dataservice="east",datasource="db1-demo.continuent.com",connections="total",} 0.0
# HELP tungsten_connector_connections Connector active/total connections value
# TYPE tungsten_connector_connections gauge
tungsten_connector_connections{dataservice="east",datasource="db2-demo.continuent.com",connections="active",} 0.0
tungsten_connector_connections{dataservice="east",datasource="db2-demo.continuent.com",connections="total",} 0.0
# HELP tungsten_connector_connections Connector active/total connections value
# TYPE tungsten_connector_connections gauge
tungsten_connector_connections{dataservice="east",datasource="db3-demo.continuent.com",connections="active",} 0.0
tungsten_connector_connections{dataservice="east",datasource="db3-demo.continuent.com",connections="total",} 2.0
# HELP tungsten_connector_connections Connector active/total connections value
# TYPE tungsten_connector_connections gauge
tungsten_connector_connections{dataservice="usa_active_active",datasource="east",connections="active",} 0.0
tungsten_connector_connections{dataservice="usa_active_active",datasource="east",connections="total",} 0.0
# HELP tungsten_connector_connections Connector active/total connections value
# TYPE tungsten_connector_connections gauge
tungsten_connector_connections{dataservice="usa_active_active",datasource="west",connections="active",} 0.0
tungsten_connector_connections{dataservice="usa_active_active",datasource="west",connections="total",} 0.0
# HELP tungsten_connector_connections Connector active/total connections value
# TYPE tungsten_connector_connections gauge
tungsten_connector_connections{dataservice="west",datasource="db4-demo.continuent.com",connections="active",} 0.0
tungsten_connector_connections{dataservice="west",datasource="db4-demo.continuent.com",connections="total",} 0.0
# HELP tungsten_connector_connections Connector active/total connections value
# TYPE tungsten_connector_connections gauge
tungsten_connector_connections{dataservice="west",datasource="db5-demo.continuent.com",connections="active",} 0.0
tungsten_connector_connections{dataservice="west",datasource="db5-demo.continuent.com",connections="total",} 0.0
# HELP tungsten_connector_connections Connector active/total connections value
# TYPE tungsten_connector_connections gauge
tungsten_connector_connections{dataservice="west",datasource="db6-demo.continuent.com",connections="active",} 0.0
tungsten_connector_connections{dataservice="west",datasource="db6-demo.continuent.com",connections="total",} 0.0
```

6.17.4.10. Alerting Using Prometheus Rules

Below are example alerting rules for Prometheus.

These rules cover basic cluster health and operation.

```
- alert: 'MasterReadOnly'
  expr: mysql_global_variables_read_only == 1 and on(hostname)(tungsten_manager_service{role="master"})
  for: 2m
    description: Database is read only on {{ $labels.hostname}}, but the role is master.

- alert: 'TungstenReplicatorDown'
  expr: up == 0 and {job=~"tungsten-exporters.*",instance=~".*8091"}
  for: 10m
    description: 'Tungsten replicator down or unreachable on {{ $labels.hostname }}, please verify that the replicator is running and exporter is returnin
```

```
- alert: 'TungstenManagerDown'
  expr: up == 0 and {job=~"tungsten-exporters.*",instance=~".*8092"}
  for: 10m
    description: 'Tungsten manager down or unreachable on {{ $labels.hostname }}, please verify that the manager is running and exporter is returning metr

- alert: 'TungstenConnectorDown'
  expr: up == 0 and {job=~"tungsten-exporters.*",instance=~".*8093"}
  for: 10m
    description: 'Tungsten connector down or unreachable on {{ $labels.hostname }}, please verify that the connector is running and exporter is returning

- alert: 'TungstenReplicatorOffline'
  expr: tungsten_replicator_service{state!="online"}
  for: 10m
    description: 'Tungsten replicator not online on {{ $labels.hostname }}, please investigate. (shell> trepctl status)'

- alert: 'TungstenManagerOffline'
  expr: tungsten_manager_service{state!="online"}
  for: 10m
    description: 'Tungsten manager not online on {{ $labels.hostname }}, please investigate. (shell> trepctl status) and (shell> echo ls | cctrl)'

- alert: 'TungstenManagerMaintenance'
  expr: tungsten_manager_policy{policy!="AUTOMATIC"}
  for: 6h
    description: 'Tungsten manager policy not AUTOMATIC on {{ $labels.hostname }}, please check if cluster is still under maintenance.'

- alert: 'TungstenTwoReplicatorMasters'
  expr: sum by (vip)(tungsten_replicator_service{role="master",state="online"})!= 1
  for: 10m
    description: 'Tungsten - multiple replicator masters (shell> trepctl services) for {{ $labels.vip }}, cannot serve two masters. Please investigate imm

- alert: 'TungstenHeapSpaceUsage'
  expr: jvm_memory_bytes_used{area="heap"}/jvm_memory_bytes_max{area="heap"}*100 > 90
  for: 20m
    description: 'Tungsten - heap space more than 90% full for more than 20 minutes on {{ $labels.instance}}. (look at tmsvc.log)'

- alert: 'TungstenReplicaStale'
  expr: tungsten_replicator_latency{latency="relative"} > 3600
  for: 10m
    description: 'Tungsten - no updates on replica {{ $labels.hostname}} for 60 minutes. Check if replicas are behind or if there is no DB activity to rep

- alert: 'TungstenReplicaNoProgress'
  expr: rate(tungsten_replicator_seqno{seqno="current"}[10m]) == 0
  for: 10m
    description: 'Tungsten - no updates on replica {{ $labels.hostname}} for over 10 minutes. Check if replicas are behind or if there is no DB activity t

- alert: 'TungstenZeroDatasourceMasters'
  expr: (sum by (vip)(tungsten_manager_service{role="master"}) < 1) and (sum by (vip)(tungsten_manager_service{role="relay"}) < 1)
  for: 10m
    description: 'Tungsten - zero datasource masters/relays (shell> echo ls | cctrl) for {{ $labels.vip }}, cannot function with no datasource masters. Pl

- alert: 'TungstenMasterRoleNotConsistent'
  expr: (sum by (hostname)(tungsten_manager_service{role="master"}) == sum by (hostname)(tungsten_replicator_service{role="master"})) != 1
  for: 10m
    description: 'Tungsten role=master not consistent between manager and replicator on {{ $labels.hostname}}.'

- alert: 'TungstenSlaveRoleNotConsistent'
  expr: (sum by (hostname)(tungsten_manager_service{role="slave"}) == sum by (hostname)(tungsten_replicator_service{role="slave"})) != 1
  for: 10m
    description: 'Tungsten role=slave not consistent between manager and replicator on {{ $labels.hostname}}.'

- alert: 'TungstenRelayRoleNotConsistent'
  expr: (sum by (hostname)(tungsten_manager_service{role="relay"}) == sum by (hostname)(tungsten_replicator_service{role="relay"})) != 1
  for: 10m
    description: 'Tungsten role=relay not consistent between manager and replicator on {{ $labels.hostname}}.'
```

6.18. Managing THL Files

In most cases the default location, size and retention policy of the THL files is sufficient, however from time to time you may need to adjust some of these options. This section outlines the various processes when working with THL files.

To clean up THL files, you can use the thl purge. For details on this, see Section 9.20.6, “thl purge Command”

6.18.1. 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 6.18.1.1, “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 6.18.1.2, “Relocating THL Storage using Configuration Changes”.

6.18.1.1. Relocating THL Storage using Symbolic Links

In an emergency, the directory currently holding the THL information, can be moved using symbolic links to relocate the files to a location with more space.

Moving the THL location requires updating the location for a Replica by temporarily setting the Replica offline, updating the THL location, and re-enabling back into the cluster:

1. Shun the datasource and switch your node into the offline state using cctrl:

```
shell> cctrl -expert
[LOGICAL:EXPERT] /alpha > datasource host1 shun
[LOGICAL:EXPERT] /alpha > replicator host1 offline
```

2. Create a new directory, or attach a new filesystem and location on which the THL content will be located. You can use a directory on another filesystem or connect to a SAN, NFS or other filesystem where the new directory will be located. For example:

```
shell> mkdir /mnt/data/thl
```

3. Copy the existing THL directory to the new directory location. For example:

```
shell> rsync -r /opt/continuent/thl/* /mnt/data/thl/
```

4. Move the existing directory to a temporary location:

```
shell> mv /opt/continuent/thl /opt/continuent/old-thl
```

5. Create a symbolic link from the new directory to the original directory location:

```
shell> ln -s /mnt/data/thl /opt/continuent/thl
```

6. Recover the host back into the cluster :

```
shell> cctrl -expert
[LOGICAL:EXPERT] /alpha > datasource host1 recover
```

To change the THL location on a Primary:

1. Manually promote an existing Replica to be the new Primary:

```
[LOGICAL] /alpha > switch to host2
SELECTED SLAVE: host2@alpha
PURGE REMAINING ACTIVE SESSIONS ON CURRENT MASTER 'host1@alpha'
PURGED A TOTAL OF 0 ACTIVE SESSIONS ON MASTER 'host1@alpha'
FLUSH TRANSACTIONS ON CURRENT MASTER 'host1@alpha'
PUT THE NEW MASTER 'host2@alpha' ONLINE
PUT THE PRIOR MASTER 'host1@alpha' ONLINE AS A SLAVE
RECONFIGURING SLAVE 'host3@alpha' TO POINT TO NEW MASTER 'host2@alpha'
SWITCH TO 'host2@alpha' WAS SUCCESSFUL
```

2. Update the THL location as provided in the previous sequence.

3. Switch the updated Replica back to be the Primary:

```
[LOGICAL] /alpha > switch to host1
SELECTED SLAVE: host1@alpha
PURGE REMAINING ACTIVE SESSIONS ON CURRENT MASTER 'host2@alpha'
PURGED A TOTAL OF 0 ACTIVE SESSIONS ON MASTER 'host2@alpha'
FLUSH TRANSACTIONS ON CURRENT MASTER 'host2@alpha'
PUT THE NEW MASTER 'host1@alpha' ONLINE
PUT THE PRIOR MASTER 'host2@alpha' ONLINE AS A SLAVE
RECONFIGURING SLAVE 'host3@alpha' TO POINT TO NEW MASTER 'host1@alpha'
SWITCH TO 'host1@alpha' WAS SUCCESSFUL
```

6.18.1.2. Relocating THL Storage using Configuration Changes

To permanently change the directory currently holding the THL information can by reconfigured to a new directory location.

To update the location for a Replica by temporarily setting the Replica offline, updating the THL location, and re-enabling back into the cluster:

1. Shun the datasource and switch your node into the offline state using cctrl:

```
shell> cctrl -expert
```

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

2. Create a new directory, or attach a new filesystem and location on which the THL content will be located. You can use a directory on another filesystem or connect to a SAN, NFS or other filesystem where the new directory will be located. For example:

```
shell> mkdir /mnt/data/thl
```

3. Copy the existing THL directory to the new directory location. For example:

```
shell> rsync -r /opt/continuent/thl/* /mnt/data/thl/
```

4. Change the directory location using tpm to update the configuration for a specific host:

```
shell> tpm update --thl-directory=/mnt/data/thl --host=host1
```

5. Recover the host back into the cluster :

```
shell> cctrl -expert
[LOGICAL:EXPERT] /alpha > datasource host1 recover
```

To change the THL location on a Primary:

1. Manually promote an existing Replica to be the new Primary:

```
[LOGICAL] /alpha > switch to host2
SELECTED SLAVE: host2@alpha
PURGE REMAINING ACTIVE SESSIONS ON CURRENT MASTER 'host1@alpha'
PURGED A TOTAL OF 0 ACTIVE SESSIONS ON MASTER 'host1@alpha'
FLUSH TRANSACTIONS ON CURRENT MASTER 'host1@alpha'
PUT THE NEW MASTER 'host2@alpha' ONLINE
PUT THE PRIOR MASTER 'host1@alpha' ONLINE AS A SLAVE
RECONFIGURING SLAVE 'host3@alpha' TO POINT TO NEW MASTER 'host2@alpha'
SWITCH TO 'host2@alpha' WAS SUCCESSFUL
```

2. Update the THL location as provided in the previous sequence.

3. Switch the updated Replica back to be the Primary:

```
[LOGICAL] /alpha > switch to host1
SELECTED SLAVE: host1@alpha
PURGE REMAINING ACTIVE SESSIONS ON CURRENT MASTER 'host2@alpha'
PURGED A TOTAL OF 0 ACTIVE SESSIONS ON MASTER 'host2@alpha'
FLUSH TRANSACTIONS ON CURRENT MASTER 'host2@alpha'
PUT THE NEW MASTER 'host1@alpha' ONLINE
PUT THE PRIOR MASTER 'host2@alpha' ONLINE AS A SLAVE
RECONFIGURING SLAVE 'host3@alpha' TO POINT TO NEW MASTER 'host1@alpha'
SWITCH TO 'host1@alpha' WAS SUCCESSFUL
```

6.18.2. Changing the THL Retention Times

THL files are by default retained for seven days, but the retention period can be adjusted according the to requirements of the service. Longer times retain the logs for longer, increasing disk space usage while allowing access to the THL information for longer. Shorter logs reduce disk space usage while reducing the amount of log data available.

> **Note**
>
> The files are automatically managed by Tungsten Replicator. 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 `thl-log-retention` [520] setting. The replication service will be restarted on each host with updated retention configuration.

6.18.3. Rebuilding THL on the Primary

If THL is lost on a Primary before the events contained within it have been applied to the Replica(s), the THL will need to be rebuilt from the existing MySQL binary logs.

> **Important**
>
> If the MySQL binary logs no longer exist, then recovery of the lost transactions in THL will NOT be possible.

The basic sequence of operation for recovering the THL on both Primary and Replicas is:

1. Gather the failing requested sequence numbers from all Replicas:

```
shell> trepctl status

...
pendingError           : Event extraction failed
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: Client handshake failure: Client response validation failed: Master log »
does not contain requested transaction: master source ID=db1 client source ID=db2 requested »
seqno=22 client epoch number=22 master min seqno=27 master max seqno=27
...
```

In the above example, when Replica db2 comes back online, it requests a copy of the last seqno in local thl (22) from the Primary db1 to compare for data integrity purposes, which the Primary no longer has.

Keep a note of the lowest sequence number and the host that it is on across all Replicas for use in the next step.

2. On the Replica with the lowest failing requested seqno, get the epoch, source-id and event-id (binlog position) from the THL using the command thl list -seqno [350] specifying the sequence number above. This information will be needed on the extractor (Primary) in a later step. You can add the option `-headers` to reduce the amount of output if required. For example:

```
[tungsten@db2]$ thl list -seqno 22 -headers

SEQ# = 22 / FRAG# = 0 (last frag)
- FILE = thl.data.0000000001
- TIME = 2025-02-10 11:04:42.0
- EPOCH# = 22
- EVENTID = mysql-bin.000011:0000000000004723;181
- SOURCEID = db1
```

There are two more ways of getting the same information using the dsctl command, so use the one you are most comfortable with:

```
[tungsten@db2]$ dsctl get
[{"extract_timestamp":"2025-02-10 11:04:42.0","eventid":"mysql-bin.000011:0000000000004723;181",»
"fragno":0,"last_frag":true,"seqno":22,"update_timestamp":"2025-02-10 11:04:47.0",»
"shard_id":"tungsten_alpha","applied_latency":5,"epoch_number":22,"task_id":0,"source_id":"db1"}]
```

```
[tungsten@db2]$ dsctl get -ascmd
dsctl set -seqno 22 -epoch 22 -event-id "mysql-bin.000011:0000000000004723;181" -source-id "db1"
```

Additionally, if you are using v7.0.2 or later, you can also use thl dsctl:

```
[tungsten@db2]$ thl dsctl -seqno 22

dsctl -service alpha set -reset -seqno 22 -epoch 22 -event-id "mysql-bin.000011:0000000000004723;181" -source-id "db1"
```

3. Place the cluster into *MAINTENANCE* and take the primary replicator `OFFLINE`

```
[tungsten@db1]$ cctrl
[LOGICAL] /alpha > set policy maintenance
[LOGICAL] /alpha > replicator db1 offline
```

4. Clear all THL on the Primary since it is no longer needed by any Replicas:

```
[tungsten@db1]$ thl purge
```

5. Use the dsctl command on the Primary with the values we got from the Replica with the lowest seqno to tell the Primary replicator to begin generating THL starting from that event in the MySQL binary logs:

   Note: If you used the dsctl get -ascmd or the thl dsctl earlier, you may use that provided command now. Ensure the `-reset` option is supplied.

```
[tungsten@db1]$ dsctl set -reset -seqno 22 -epoch 22 -event-id "mysql-bin.000011:0000000000004723;181" -source-id "db1"
```

6. Place the cluster back to *AUTOMATIC*. This should also return the replicator to the `ONLINE` state.

```
[tungsten@db1]$ cctrl
[LOGICAL] /alpha > set policy automatic
```

7. Switch the Replicas to online state once the Primary is fully online:

```
[tungsten@db2]$ trepctl online
```

6.18.4. THL Encryption and Compression

The ability to Compress and/or Encrypt THL was introduced in v7.0.0

Encryption is applied to THL on disk, in flight encryption is handled by enabling the various SSL features of the Replicator

Compression can be enabled in-flight by changing the various configuration properties, and compression on disk can be enabled/disabled either dynamically or by changing the various configuration properties.

The following sections explain enabling/disabling these features in more detail.

6.18.4.1. In-Flight Compression

Compression occurs "in-flight" and is requested by the client replicator prior to fetching the THL from the remote THL Server.

Enabling THL Compression

The following property should be added to your configuration:

```
repl-thl-client-serialization=[SERIALIZATION PROTOCOL]
```

For example:

```
repl-thl-client-serialization=PROTOBUF
```

Valid values for the protocol are: `LEGACY, JAVA, PROTOBUF` or `DEFLATE`

The default for this property if not supplied is `LEGACY`. This retains the behavior in versions prior to v7.0.0 and has the same effect as disabling compression.

`DEFLATE` offers the highest level of compression, but at the cost of being slower during the compression and decompression stages

`JAVA` should not be used in production, and is mainly used for testing purposes.

The THL Server can be configured to accept one or more of these protocols. By default, a server will support ALL protocols. This can be adjusted as follows:

```
repl-thl-server-serialization=[COMMA SEPARATED LIST OF PROTOCOLS]
```

For example, the following would disable `DEFLATE`:

```
repl-thl-server-serialization=LEGACY,PROTOBUF
```

If a client asks for a protocol that is not enabled, it will fall back to `LEGACY`

6.18.4.2. Encryption and Compression On-Disk

THL Encryption and Compression On-Disk can be enabled at install time or dynamically.

The following property should be added to your configuration:

```
replicator-store-thl-encrypted=true|false
replicator-store-thl-compressed=true|false
```

By default, both encryption and compression are disabled.

To change these settings dynamically, the service will need to be put offline first. This will force the replicator to rotate to a new THL log file which will use the new settings.

> **Note**
>
> If only enabled/disabled dynamicaly, the settings WILL persist on a replicator restart.

The commands to enable or disable these settings are:

```
shell> trepctl [-service servicename] thl -compression {enable|disable}
shell> trepctl [-service servicename] thl -encryption {enable|disable}
```

The full steps to enable encryption follows:

```
shell> cctrl
cctrl> set policy maintenance
cctrl> exit

shell> trepctl [-service servicename] offline
shell> trepctl [-service servicename] thl -encryption enable
shell> trepctl [-service servicename] online
```

```
shell> cctrl
cctrl> set policy automatic
cctrl> exit
```

As a result, after this command, thl index command will show the newly generated THL log file as encrypted:

```
shell> thl [-service servicename] index
….
LogIndexEntry thl.data.0000000001(0:28)
LogIndexEntry thl.data.0000000002(29:29) - ENCRYPTED (tls)
```

Encryption uses dedicated keystore and truststore (named by default `tungsten_thl_keystore.jks` and `tungsten_thl_truststore.ts`). Losing these files will make encrypted THL log files impossible to be decoded.

To specify your own truststore and keystores instead of the default mentioned above, you can set this in the configuration using the tpm options `java-thl-keystore-path` and `java-thl-truststore-path`.

The password used for all keystore and truststore files is global and can be configured from the default by using the tpm option `java-key-store-password`.

The Replicator can generate new keys to be used. These keys will then be sent through the THL protocol to other nodes.

To generate a new key, the following command needs to be executed on the primary replicator while the service is `online`:

```
shell> trepctl [-service servicename] thl -encryption genkey
```

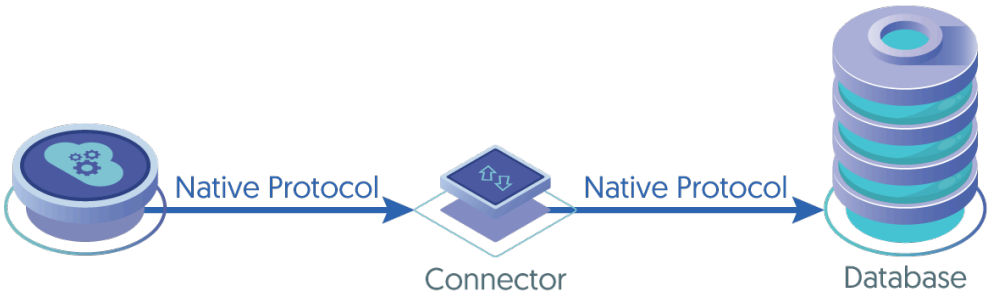
This will result in a new THL log file to be started, using the new generated key, as shown by thl index command:

```
shell> thl [-service servicename] index
….
LogIndexEntry thl.data.0000000001(0:28)
LogIndexEntry thl.data.0000000002(29:30) - ENCRYPTED (tls)
LogIndexEntry thl.data.0000000003(31:33) - ENCRYPTED (tls)
LogIndexEntry thl.data.0000000004(34:34) - ENCRYPTED (thl.alpha.34)
```

Chapter 7. Tungsten Connector

Tungsten Connector acts as a proxy service, sitting between client applications and datasources in order to balance the load and provide high availability (HA) support. The service works by accepting raw packets from clients, and then forwarding them to the datasource. The process is reversed when packets are sent back from the datasource and then redirected back to the clients.

Figure 7.1. Tungsten Connector Basic Architecture



In addition to this basic structure, Tungsten Connector also works with the other components of Tungsten Cluster to handle some specific scenarios and situations:

- The connector works in harmony with the Tungsten Manager as part of Tungsten Cluster and enables the connector to redirect queries between known datasources within a given dataservice. For example, when the manager identifies a failed datasource, queries to that datasource are redirected to an alternative datasource without the application being aware of the change.
- The connector works with the Tungsten Cluster configuration and a number of implied or explicit directives that enable the connector to redirect requests within different datasources within the network. For example, the connector can be configured to automatically forward write requests to a database to the active Primary within the dataservice and reads to active Replicas.

Throughout this process the connector is redirecting the network packets sent by application servers to the appropriate host. The contents and individual statements are not processed or accessed. At all times applications and clients using the connector do not need modification as to them it will appear as a MySQL server.

To start using the connector run the tpm connector command. This will open a connection with the MySQL CLI. See Section 7.3, "Clients and Deployment" for more detail on configuring your application to use the connector.

> **Important**
>
> After installation, if Bridge Mode was disabled, the connector will only work with the `application-user` [474] and `application-password` [473] that were provided during installation. See Section 7.6, "User Authentication" if you need more information on adding users to `user.map`.

```
shell > tpm connector
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 82
Server version: 8.0.40-tungsten MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql >
```

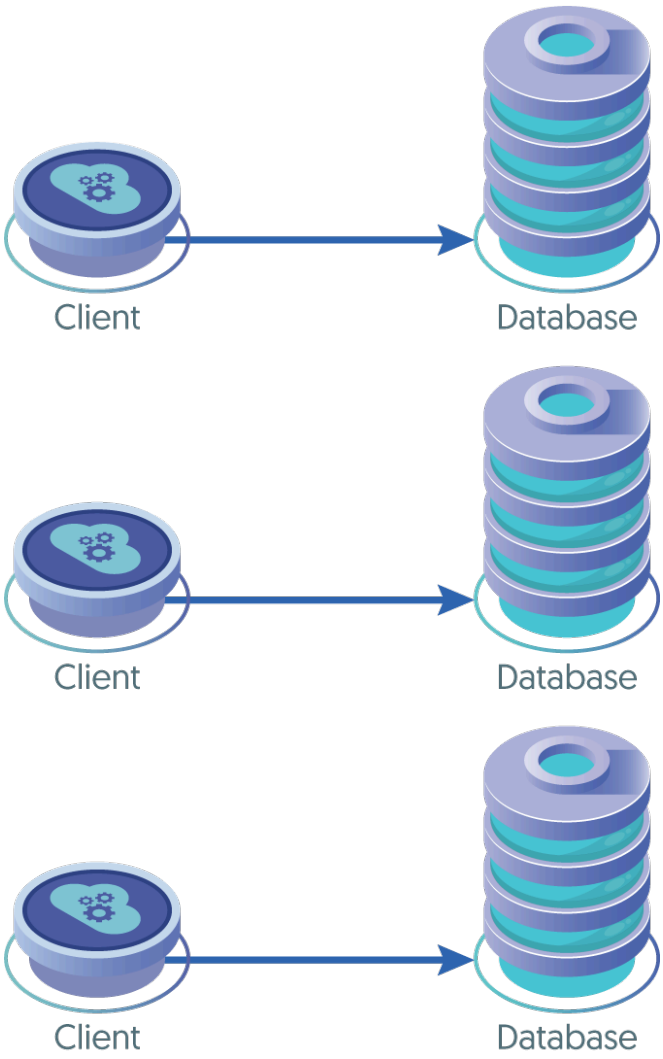
7.1. Connector/Application Basics

Within any typical database deployment there will be two primary considerations:

- High Availability — redirecting requests to alternative servers in the event of a failure.
- Scalability — distributing reads and writes across servers in a replication architecture.

Within a typical basic database deployment clients and application servers will connect directly to databases, as shown in Figure 7.2, "Basic MySQL/Application Connectivity".

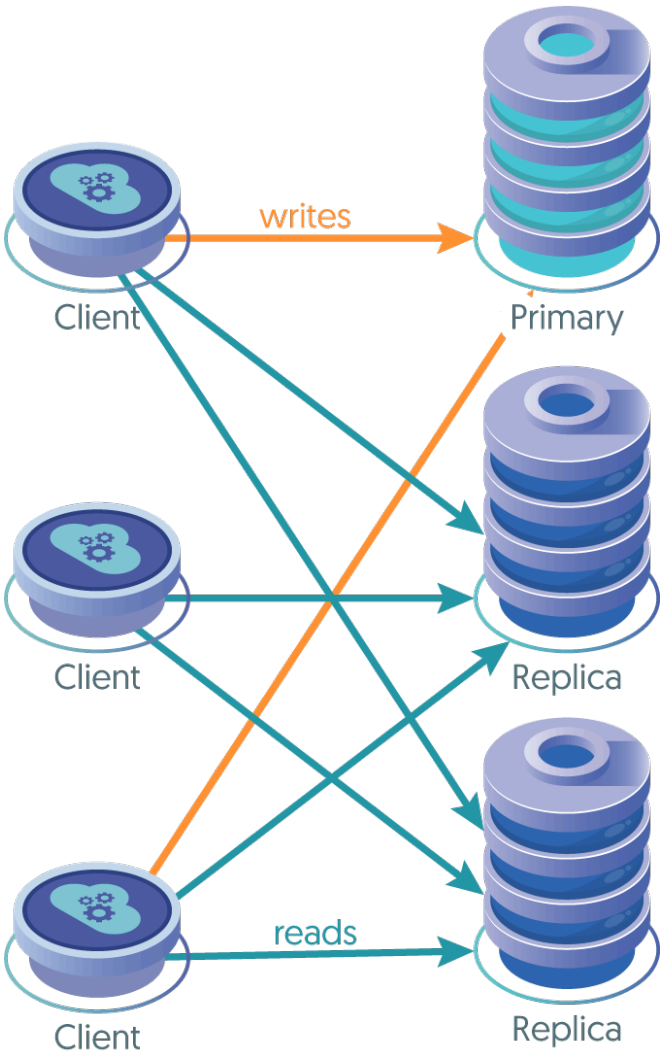
Figure 7.2. Basic MySQL/Application Connectivity



The problem with this deployment model is that it is not able to cope with changes or problems. If one or more of your database servers fails, then the application servers must be reconfigured individually to point to an alternative server. In addition, when considering scalability, there is no provision for redirecting reads and writes to Primaries or Replicas.

In an advanced application deployment, individual servers and/or application may have been configured to connect to Primaries and Replicas within the database infrastructure to handle the scalability offered by using replication (Figure 7.3, "Advanced MySQL/Application Connectivity"). For this to work the application must have been modified and be read/write aware, and it must have been configured to manually connect to the different databases according to the operation being performed.

Figure 7.3. Advanced MySQL/Application Connectivity



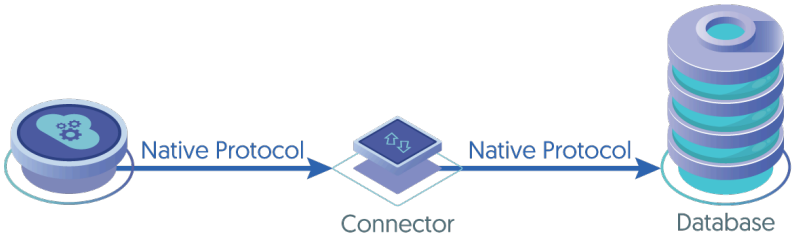
Although this handles the read/write splitting, enabling servers to write to the Primary and read from one or more Replicas, changes to this architecture and structure are not handled. If the Primary fails, application servers must be manually updated to direct their queries to an alternative host.

When deploying Tungsten Cluster the connector takes over the role of primary connection from your application to the database server, and it handles the redirection of requests to the appropriate database server. Depending on your application configuration and architecture, the connector can be used in two ways:

- As a complete solution for redirecting queries between the Primary and Replica hosts within a dataservice, including HA events.
- As a HA solution redirecting queries to the Primary and Replicas within a dataservice, but with application driven Primary/Replica selection.

When deploying your application with Tungsten Cluster through Tungsten Connector, the application server connectivity is through the connector. The connector takes on the role of primary connection for all requests, while routing and distributing those requests to the active datasources within the dataservice.

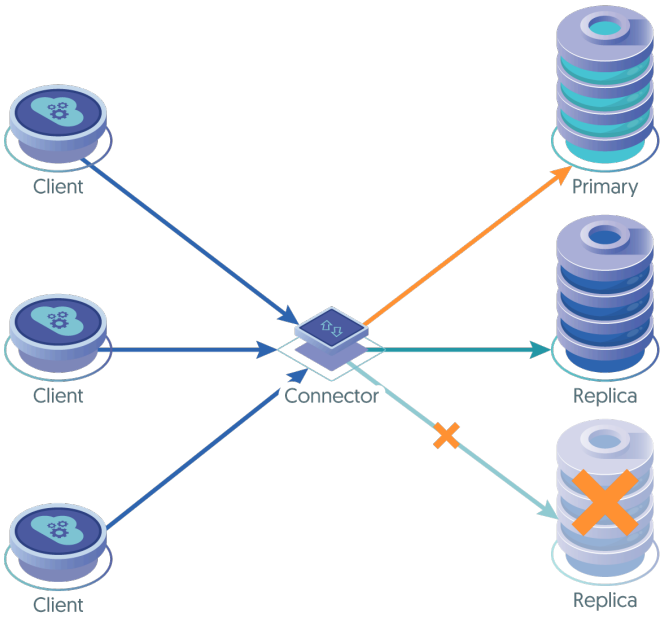
Figure 7.4. Using Tungsten Connector for MySQL/Application Connectivity



The Tungsten Connector is located between the clients and the database servers, providing a single connection point, while routing queries to the database servers. In the event of a failure, the Tungsten Connector can automatically route queries away from the failed server and towards servers that are still operating. During the routing process, Tungsten Connector communicates with the Tungsten Manager to determine which datasources are the most up to date, and their current role so that the packets can be routed properly.

Because the connectivity is between the application service and the Tungsten Connector, the connection to the Tungsten Connector remains constant. Changes to the datasources, including failures, role changes, and expansion or removal of datasources from the dataservice do not require any modification of the application configuration or operation.

Figure 7.5. Tungsten Connector during a failed datasource



For example, in Figure 7.5, “Tungsten Connector during a failed datasource”, the Replica datasource has failed. While this would break the connection between the Tungsten Connector and the datasource, the connection between the application and Tungsten Connector remains available, and Tungsten Connector will re-route queries to an available datasource without reconfiguring the application server connectivity.

7.1.1. Connector Control Flow

The Connector makes a TCP connection to any available Manager, then all command-and-control traffic uses that channel. The Manager never initiates a connection to the Connector.

When there is a state change (i.e. shun, welcome, failover, etc.), the Manager will communicate to the Connector over the existing channel.

The Connector will re-establish a channel to an available Manager if the Manager it is connected to is stopped or lost.

7.2. Basic Connector Configuration

By default, the Tungsten Connector listens on port `9999` so as to avoid conflicting with the default MySQL listener port of `3306`.

The best practice is the change the MySQL port to `13306` and the Tungsten Connector port to `3306`.

For example, modify `my.cnf` to define the port, then restart MySQL:

```
port = 13306
```

For more information, see Section B.4.2, "MySQL Configuration".

Configure the the Tungsten Connector to listen on port `3306`, by adding the following entry to the `/etc/tungsten/tungsten.ini`:

```
application-port=3306
```

> **Important**
>
> This change is especially important if the Tungsten Connector will be installed directly on the database nodes. ALL traffic needs to flow through the Tungsten Connector, so hiding the actual MySQL port prevents mistaken connections directly to the database.

7.3. Clients and Deployment

In order to get the benefits of Tungsten Cluster your application must use the Tungsten Connector. The connector is compatible with MySQL drivers and applications. Use the tpm connector --samples command to see examples of how you can invoke a connection on your own. You may need to adapt these examples to your application and configuration method but the connection details should be the same.

> **Note**
>
> Passwords are obsfucated in the output and will need replacing with the correct password that you configured.

```
shell > tpm connector --samples
Bash -
mysql -hdb1 -P3306 -uapp_user -p'PASSWORD_OBFUSCATED'

Perl::dbi -
$dbh = DBI->connecti('DBI:mysql:host=db1;port=3306', 'app_user', 'PASSWORD_OBFUSCATED')

PHP::mysqli -
$dbh = new mysqli('db1', 'app_user', 'PASSWORD_OBFUSCATED', 'schema', '3306');

PHP::pdo - $
dbh = new PDO('mysql:host=db1;port=3306', 'app_user', 'PASSWORD_OBFUSCATED');

Python::mysql.connector -
dbh = mysql.connector.connect(user='app_user', password='PASSWORD_OBFUSCATED', host='db1', port=3306, database='schema')

Java::DriverManager -
dbh=DriverManager.getConnection("jdbc:mysql://db1:3306/schema", "app_user", 'PASSWORD_OBFUSCATED')
```

If the default Bridge Mode was disabled in favour of Proxy mode, then after installation the connector will only work with the `application-user` [474] and `application-password` [473] options that were provided during installation. See Section 7.6, "User Authentication" if you need more information on adding users to `user.map`.

By default the connection will always be sent to the current Primary. This behavior can be modified by implementing one of the routing methods to send some traffic to Replica datasources.

7.3.1. Connection Pools

Tungsten Connector can work behind a connection pool without any issue.

Upon switch or failover, all connections in the pool will be broken but not closed. The very next time one of these pooled connections is used, it will be transparently reconnected unless autoReconnect has been disabled by the tpm installation flag `connector-autoreconnect=false` [479]

> **Important**
>
> When using a connection pool mechanism in applications, the dynamic connection setting `maxAppliedLatency` should not be used. This setting is only meaningful at connection time; since connection pools open a set of connections, the applied latency will most probably be outdated when the application will actually use the pooled connection

7.4. Routing Methods

Bridge mode is the default connector routing method if no other routing method is explicitly stated. Bridge mode does not use the `user.map` file which reflects other changes to take a more secure default deployment. It will not be enabled in the following cases:

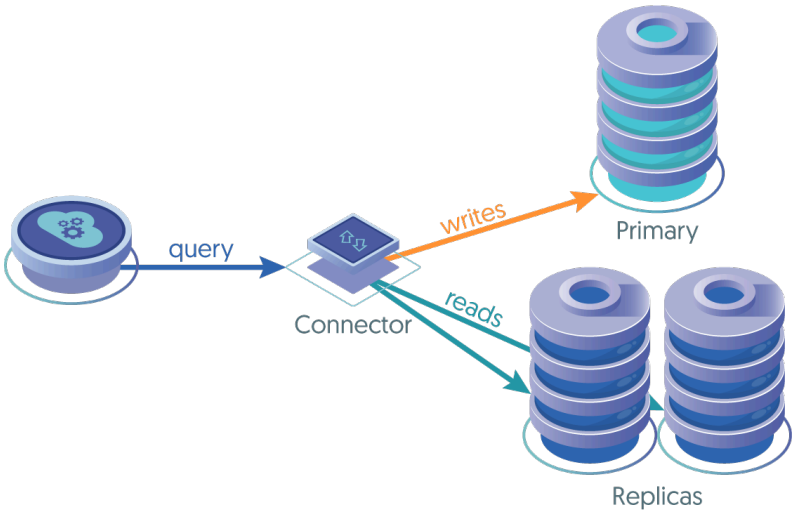
- The `connector-bridge-mode` [480] option is set to `false`.

- The `connector-smartscale` [483] option is set to `true`.
- The `user.map` file contains `@direct` entries.
- The `user.map` file contains `@hostoption` entries.
- The `property=selective.rwsplitting` [443] connector option is set to `true`.

Tungsten Connector routes connections between client connections and datasources using a number of different routing methods. These routing methods affect how client applications and datasources are connected to each other, and control the level of inspection by Tungsten Connector of the connections and statements as they pass through the connector service.

The Tungsten Connector works with Tungsten Manager to automatically route clients connected to the connector to an appropriate server, balancing the load when communicating with Replicas. The different methods are involved in effective read/write splitting, i.e. the ability to correctly route requests to the Primaries or Replicas within the network according to the type of operation being performed by the client. This can be performed automatically, or manually, or through a series of specific configurable routing methods.

Figure 7.6. Tungsten Connector routing architecture



Routing selection is made by the connector based on the availability information using a combination of different settings and parameters. Each level overrides or augments the previous level, and each can be specified in different locations, such as the `user.map`, connecting string, or within individual supplied statements. The settings are processed in the order shown below; later setting override earlier settings.

For example, selecting the SQL routing method defines the default behavior. Specifying the QoS in the `user.map` file supercedes the SQL routing; setting QoS in a comment before the SQL statement supercedes the user and default behavior. Specifying an affinity in the comments overrides both the user and default configuration settings.

- Selected routing method, see Section 7.4.1, “Connector Routing Methods”
- Quality of Service (QoS) specification, see Section 7.4.2, “Primary/Replica Selection”
- Load balancer selection (implied by QoS), see Section 7.4.3, “Connector Load Balancers”
- Replica latency, including optional maximum latency setting, see Section 7.4.4, “Specifying Required Latency”
- Affinity specification, see Section 7.4.5, “Setting Read Affinity and Direct Reads”

These different routing configurations can be selected according to the global configuration, and customization at different points in the communication channel. For example, SQL-based routing configures basic load-balancing, but allows SQL comments to be used to change the default QoS mode and affinity.

	Routing Method	QoS	Latency	Affinity
Global Configuration	Yes	Implied	Yes	Yes
Connection String	Yes	Yes	Yes	Yes
`user.map`	Yes	Yes	Yes	Yes
SQL statement	Yes (with SQL routing enabled)	Yes (with SQL routing enabled)	No	No

At all times, the connector uses the current status of the MySQL servers to make decisions about where queries and connections should be routed. Changes to the Primary, and availability or accessibility of individual dataservers will always be taken into account when routing the queries. For information on what happens if failure occurs during an operation or transaction, see Section 7.8, "Connector Operational States".

The routing methods can either involve direct reads, SmartScale, host-based routing, or SQL inspection-based routing to redirect reads and writes to the appropriate server. In addition to these implied routing methods, clients can also specifically select which host to communicate with through the use of tags and options provided through the connection string.

The selection of a datasource occurs at the point the client connects, and this datasource connection choice remains in effect until the client disconnects, unless a failover or switch occurs.

7.4.1. Connector Routing Methods

The supported routing methods, typical uses and use cases are listed in Table 7.1, "Routing Method Selection".

Table 7.1. Routing Method Selection

Routing Method	Host Selection	Auto R/W Splitting	Replica Latency	Maximum Applied Latency
Smartscale	By Session	Yes (by SQL statement)	Lazy	Yes
Direct Reads	By Content	Yes (by SQL statement)	Lazy	Yes
Host-based	By Hostname	No	Yes	Yes
Port-based	By Network Port	No	No	Yes
SQL-based	By SQL comment	No	No	Yes

Important

Both SmartScale and R/W splitting cannot be enabled at the same time. This is because they are two sides of the basic functionality. R/W splitting and SmartScale both use SQL introspection to determine whether a query should be directed to a Primary or a Replica. SmartScale combines this with an intelligent load-balancer. R/W splitting uses a simpler direct redirection.

In addition to the selection and configuration mechanisms supported, a routing method should be chosen based on your application abilities:

- If the application is replication-aware, and can already direct queries to Primary or Replicas based on the operation type, use Section 7.4.13, "Host-based Routing" or Section 7.4.14, "Port-based Routing"
- If the application has full control of the SQL statements submitted (i.e. not through a third-party tool, or Object-Relational Modeling library), and can already direct queries to Primary or Replicas based on the operation type, use Section 7.4.12, "SQL Routing".
- If the application uses non-auto-commit statements (for example, Hibernate), Section 7.4.13, "Host-based Routing", or Section 7.4.12, "SQL Routing".
- If the application does not fit any of these categories, or is not replication aware use either either Section 7.4.11, "Direct Routing" or Section 7.4.10, "Smartscale Routing".

7.4.2. Primary/Replica Selection

Depending on the chosen routing and authentication method, the selection of the Primary or Replica node can be controlled by the use of the `qos` (Quality Of Service) syntax as outlined below:

- `RO_RELAXED` [240]

  This setting enables the connector to redirect the query as if it were read-only, and therefore prefer a Replica over a Primary, but will choose a Primary if no Replica is available.

- `RW_STRICT` [240]

  This setting indicates that the query is a write and should be directed to a Primary. In Active-Active setups, where multiple Primaries exist, affinity determines in which site the primary should be selected.

Configuring the use of these properties can be controlled in a number of ways, as follows:

- SQL Based Routing (See Section 7.4.12, "SQL Routing")
- Host based Routing (See Section 7.4.13, "Host-based Routing")

- Embedded as part of Schema Name when specifying the database to connect to.

  For example, the following connect string would prefer a Replica for its connection:

```
jdbc:mysql://connector1:3306/mydb@qos=RO_RELAXED?autoreconnect=true
```

Further, connectivity can be influenced by setting a suitable latency (See Section 7.4.4, "Specifying Required Latency") value, or an explicit affinity (See Section 7.4.5, "Setting Read Affinity and Direct Reads".

The rules for selection of whether a connection is made to a Primary or a Replica is therefore controlled by comparing all of these settings and the selected routing mechanism together.

QoS	Primary Selected	Replica Selected	
RW_STRICT [240]	Yes, always	No.	
RO_RELAXED [240]	Only if no Replica available	Yes, if below max applied latency.	

For further reading on how nodes are chosen, see Section 7.4.3, "Connector Load Balancers"

7.4.3. Connector Load Balancers

The load balancing model used, according to the selected *QoS* is defined by a number of different load balancing classes. These are configured automatically when different *QoS* is selected, be explicitly changed by altering the configuration file. The supported load balancers are detailed in the table below:

Load Balancer	Default QoS	Description
DefaultLoadBalancer	*RW_STRICT* [240]	Always selects the Primary data source
MostAdvancedSlaveLoadBalancer	*RO_RELAXED* [240]	Selects the Replica that has replicated the most events, by comparing data sources "high water" marks. If no Replica is available, the Primary will be returned.
LowestLatencySlaveLoadBalancer		Selects the Replica data source that has the lowest replication lag, or *appliedLatency* in ls -l within cctrl output. If no Replica data source is eligible, the Primary data source will be selected.
RoundRobinSlaveLoadBalancer		Selects a Replica in a round robin manner, by iterating through them using internal index. Returns the Primary if no Replica is found online
HighWaterSlaveLoadBalancer	*RW_SESSION*	Given a session high water (usually the high water mark of the update event), selects the first Replica that has higher or equal high water, or the Primary if no Replica is online or has replicated the given session event. This is the default used when SmartScale is enabled.

The default setting is `MostAdvancedSlaveLoadBalancer`.

To change the Connector load balancer, specify the property in the configuration, i.e to use Round Robin, add the following to the `[defaults]` stanza of your `/etc/tungsten/tungsten.ini` and issue tpm update:

```
property=dataSourceLoadBalancer_RO_RELAXED=com.continuent.tungsten.router.resource.loadbalancer.RoundRobinSlaveLoadBalancer [443]
```

or for staging deployments, issue the following from the staging server:

```
shell> ./tools/tpm update alpha \
--property=dataSourceLoadBalancer_RO_RELAXED=com.continuent.tungsten.router.resource.loadbalancer.RoundRobinSlaveLoadBalancer
```

7.4.4. Specifying Required Latency

Depending on the selected routing method, load balancer and QoS setting, a Replica will automatically be chosen when the host connects. The maximum allowed latency can be set to limit the connection to only use a Replica that is within the specified maximum applied latency limit.

This can be specified in the connection string, and enables Replica selection based on the Replica which has a latency within the specified limit. Connector specific URL parameters have to be passed inside the database name, otherwise they might be ignored/stripped by the application driver.

```
jdbc:mysql://connector1:3306/<databasename here>@maxAppliedLatency=5?<mysqlSpeficOptionHere>
```

For example:

```
jdbc:mysql://connector1:3306/mydb@maxAppliedLatency=5?autoreconnect=true
```

The example above will specify that a host with a replication latency of less than 5 seconds should be selected, with autoreconnet enabled.

The option can be set globally by configuring the JDBC options used by the connector by adding the `connector-max-slave-latency` [481] option (value specified in seconds) to the `[defaults]` stanza of your `/etc/tungsten/tungsten.ini` file and issuing tpm update, or for staging deployments, issuing:

```
shell> ./tools/tpm update alpha --connector-max-slave-latency=10
```

Warning

The Connector computes latency by polling the Replicator every 3 seconds for the current replication-view latency.

This gives the Connector an accuracy of +/- 3 seconds, which means that values of 3 or less will not function as expected.

For any queries that have a very low tolerance for replication latency, we strongly suggest you read directly from the Primary database server only. This ensures the latest data is being read.

Important

The `connector-max-slave-latency` [481] flag does not ensure the Replica has applied the latest sequence number, just that its latency at the last commit was under the specified number. This behavior can be adjusted by specifying `use-relative-latency=true` [485] in the configuration.

NOTE: `use-relative-latency=true` [485] is a cluster-wide setting, cctrl and trepctl will also report relative latency based on this setting.

7.4.4.1. Applied and Relative Latency Comparison

- Applied Latency

  The appliedLatency is the latency between the commit time of the source event and the time the last committed transaction reached the end of the corresponding pipeline within the replicator. Within a Primary, this indicates the latency between the transaction commit time and when it was written to the THL. In a Replica, it indicates the latency between the commit time on the Primary database and when the transaction has been committed to the destination database. Clocks must be synchronized across hosts for this information to be accurate. appliedLatency : 0.828 The latency is measured in seconds. Increasing latency may indicate that the destination database is unable to keep up with the transactions from the Primary. In replicators that are operating with parallel apply, appliedLatency indicates the latency of the trailing channel. Because the parallel apply mechanism does not update all channels simultaneously, the figure shown may trail significantly from the actual latency.

  See Section E.2.10, "Terminology: Fields *appliedLatency*" for more information.

- Relative Latency

  The relativeLatency is the latency between now and timestamp of the last event written into the local THL. This information gives an indication of how fresh the incoming THL information is. On a Primary, it indicates whether the Primary is keeping up with transactions generated on the Primary database. On a Replica, it indicates how up to date the THL read from the Primary is. A large value can either indicate that the database is not busy, that a large transaction is currently being read from the source database, or from the Primary replicator, or that the replicator has stalled for some reason. An increasing relativeLatency on the Replica may indicate that the replicator may have stalled and stopped applying changes to the dataserver.

  See Section E.2.72, "Terminology: Fields *relativeLatency*" for more information.

- Comparing Relative and Applied Latencies

  Both relative and applied latency are visible via the trepctl status. Relative indicates the latency since the last time the appliedLastSeqno advanced; for example:

```
shell> trepctl status

Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000211:0000000020094766;0
appliedLastSeqno       : 78022
appliedLatency         : 0.571
...
relativeLatency        : 8.944
Finished status command...
```

  In this example the last transaction had an applied latency (time to write to the Replica DB) of 0.571 seconds from the time it committed on the Primary, and last committed something to the Replica DB 8.944 seconds ago.

  If relative latency increases significantly in a busy system, it may be a sign that replication is stalled. This is a good parameter to check in monitoring scripts.

- For more information,see:

  connector-max-slave-latency [481]

  use-relative-latency [485]

  Section 4.1.6.3, “Relative Latency”

7.4.4.2. Advanced Troubleshooting for Latency-based Routing

To troubleshoot the latency-based routing decisions the connector makes, uncomment the below lines in /opt/continuent/tungsten/tungsten-connector/conf/log4j.properties

```
#log4j.logger.com.continuent.tungsten.router.resource.loadbalancer=debug, stdout
#log4j.additivity.com.continuent.tungsten.router.resource.loadbalancer=false
```

The log will then show and explain what node the connector choose and why.

> **Important**
>
> No connector restart is required to enable logging. If you re-comment the lines, a restart will be required. To disable without a connector restart, replace the word debug with the word info.

7.4.5. Setting Read Affinity and Direct Reads

Affinity enables you to specify at connection time that the connector should forward the connection to a particular host or service for reads, if the service is available. For example, within user.map:

```
user password west east
```

Defines a user that uses the west service, but prefers being routed to the east service for reading from a Replica.

Affinity can also specified within the connection string:

```
jdbc:mysql://connector1:3306/database?affinity=host3&qos=RO_RELAXED
```

Additionally, affinity JDBC options can be set globally via the tpm command option connector-affinity [478].

The read affinity setting supports multiple dataservices with ordering and exclusion.

You may fine-tune the affinity by specifying an ordered list of dataservice names.

- The affinity string is an ordered list of dataservice names, separated by commas, where the first dataservice entry will be the one used by default. If the first dataservice in the list is not available, the connector will use the next one listed, and so forth.
- Dataservices not specified in the list, if any, will be used last and randomly.
- It is also possible to exclude one or more dataservices by adding a hyphen ("-") in front of the dataservice name.

Affinity can also be combined with other node selection, such as QoS. For example, by combining the affinity and *RO_RELAXED* [240], then the specified Replica will be used first, if the load-balancer setting matches, then another Replica within the same service, and finally the Primary. For example, in a dataservice with three nodes, where node1 is the Primary:

```
shell> mysql -h127.0.0.1 -P3306 databasename@qos=RO_RELAXED\&affinity=node2
```

Would use node2 first, then node3, and finally node1 if the others are not available.

> **Note**
>
> Within a composite dataservice, you cannot specify a specific host. You can only specify a physical dataservice within the composite dataservice. For example in a composite service with east and west physical dataservices:
>
> ```
> shell> mysql -h127.0.0.1 -P3306 databasename@qos=RO_RELAXED\&affinity=east
> ```

Additionally, the user.map can be configured to direct specific users to a Replica by using the @direct keyword. For example, the following line in user.map will always direct the user to a Replica, ignoring latency and load balancing settings:

```
@direct readme
```

7.4.6. Setting Read/Write Affinity in Composite Active/Active Environments

Write Affinity for Composite Active/Active environments was introduced in version 6.1.12

Setting write affinity allows write statements to be routed to the remote Active cluster, while reads may follow the affinity for the local cluster. This provides the ability to ensure conflicts do not occurr when writing to multiple active clusters.

This can be configured in a number of different ways, depending upon the configuration of your connectors.

For Proxy Routing:

When running in Proxy mode, you can set the affinity via the `user.map` by specifying the Write Dataservice and Read Dataservice as follows:

```
user password global remoteWriteCluster:localReadCluster
```

Additionally, affinity can be set per connection, for example:

```
mysql -u username -p secret -h connector1 -P 3306 database@affinity=remoteWriteCluster:localReadCluster
```

For Bridge Mode Routing:

You can enable the affinity for Bridge Mode connections by setting the `connector-affinity` [478] tpm property, for example, add the following in your ini file:

```
connector-affinity=remoteWriteCluster:localReadCluster
```

The serviceName specified either side of the : can be a single service, or a comma separated list or services.

The serviceName to the left of the : is the Write affinity, serviceNames to the right of the : apply to Read affinity

For example:

```
user password global remoteWriteCluster,localWriteCluster:localReadCluster,remoteReadCluster
```

7.4.7. Setting Negative Affinity

Negative Affinity can be enabled with both Read and Write affinity but has the opposite affect, namely ensuring Reads or Writes do NOT get routed to a specific service.

This feature can be configured in exactly the same way, just with the addition of - in front of the service name. Below is an examples using `east` and `west` as the example service names.

Allow writes to east but not west, allow reads from both, with preference to east:

```
connector-affinity=east,-west:east
```

A typical use case for this feature would be when you want the flexibility of a Composite Active/Active topology but do not want writes into both clusters. This can be useful when you are unable to avoid potential write conflicts.

7.4.8. Routing using embedded syntax in connect string

Note

This feature will not work when using Bridge mode, this only applies to connectors running in Proxy mode.

It is possible to embed syntax as part of the schema name in your connection string (`-D` switch if using the mysql CLI) to route the connector. Such as using the *`@qos`* syntax to direct the connection to a Primary or a Replica.

Additionally, you can also pass other properties such as affinity and maxAppliedLatency, as shown in the MySQL CLI example below.

```
mysql -hhost -pport -uuser -ppass -Dtest@qos=RO_RELAXED\&affinity=europe\&maxAppliedLatency=15
```

The following shows a java connect string example:

```
connection = DriverManager.getConnection("jdbc:mysql://host:port/schema@qos=RO_RELAXED&affinity=europe&maxAppliedLatency=15", "user", "pass")
```

7.4.9. Connector Datasource Selection in Composite Clusters

Within composite clusters, both Composite Active/Passive and Composite Active/Active, the selection of datasources can be made to be deterministic by specifying the list of datasources within the `user.map` configuration. The configuration is based on an ordered, comma-separated list of services to use which are then selected in order.

The format of the specification is as follows:

```
user pass compositeDS dataservice_by_order_of_priority* [-dataservice_to_exclude]*
```

The specification operates on the following rules:

- List of service names in order

- If the service name has a dash prefix it is always explicitly excluded from the list of available datasources
- If a datasource is not specified, it will always be picked last

For example, in a setup made of three data service, `usa`, `asia` and `europe` using the following configuration:

```
app_user secret usa,asia,-europe
```

Will select data sources in data service `usa`. If `usa` is not available, in `asia`. If `asia` is not availble, then connection will not succeed since `europe` has been negated.

7.4.10. Smartscale Routing

SmartScale allows you to read your data, as much as possible, from a Replica data source.

In this read-write splitting mode, the connector intelligently determines if Replicas are up-to-date with respect to the Primary, and selects them in such a way that read operations are always strictly consistent with the last write of their current session. Sessions are per-connector-instance, in-memory objects that allow different connections to share SmartScale benefits: by providing the same session id, two connections will be able to see each other's write operations consistently.

Possible session ids are:

- `DATABASE` [245]

  Applications will see write operations made to the same database as it is connected to. Reads from other databases might be outdated depending on the Replica latency

- `USER` [245]

  All connections that use the same user will read data consistent with the writes made by the current user. Other users data might be outdated.

- `CONNECTION` [245]

  Only writes made by the current connection are guaranteed to be read consistently. Writes from other connections might be outdated

- `PROVIDED_IN_DBNAME` [245]

  Allows you to specify a variable sessionid in the database connection string . An application, typically PHP, can pass its own session id to make smart scale even more efficient.

Typical use cases

- Applications which can use this level of consistency typically do relatively few writes and many reads. The writes that are performed can be considered to be in a 'silo' of their own, that is, a given application 'session' only writes and reads its own data and is not concerned with the data read/written by other application 'sessions'.
- PHP applications are good candidates for SmartScale since PHP has embedded session IDs that can be passed at connection time.
- Web based applications with user profiles match the scenario where users will update their own profile and want to see their modifications right away, but can accept latency on other users profiles.

Comparison with Direct Reads

- Smart Scale allow session consistency, while Direct Reads always read from Replica, no matter whether data is up-to-date.
- However, the cost for consistency appears at the performance level, since the Connector constantly needs to check Replica progress.
- If your application needs to see the data it just wrote, use SmartScale
- If your application does a lot of small reads that do not need to be up-to-date, use Direct Reads

Limitations

- Prepared Statements - Prepared statements will need to be enclosed between transaction boundaries in order to work correctly with read/write splitting. This way, they will always execute on the Primary. Note that all prepared statements will become invalid upon switches or failover
- Ephemeral objects - Temporary tables, session variables and other objects that are connection specific will not be accessible when reading data using SmartScale. If you need to use these ephemeral object, you should either add a "for update" statement in you selects or avoid using SmartScale
- Read/write functions - Functions that create or modify data in the database should never be called with a simple SELECT statement. Always add "for update" a the end of the select call. Example:

```
SELECT my_function_that_writes('param') FOR UPDATE
```

- Currently, it is not recommended to use SMARTSCALE in conjunction with Parallel Apply. This is due to progress only being measured against the slowest channel.

  It is hoped that this will be resolved in a future release.

7.4.10.1. Specifying the Session ID

While the three first keywords (DATABASE [245], USER [245] and CONNECTION [245]) are connector-wide (a single connector instance will use these session ids for all connections), it is possible to configure the connector to allow a free string session ID. This string will have to be passed through the database name as {db_name}?sessionId={sessionID}. For example, when using a database named "test" and a session ID number 1234, the database name passed to the connector will be:

```
test?sessionId=1234
```

With mysql command line utility, the connection command will look like:

```
mysql -h connectorHost -u user -ppass -P 3306 test?sessionId=1234
```

JDBC clients will have to pass this session ID with a special tag, as follows:

```
jdbc:mysql://connector_host:3306/dbname@sessionId=1234?otherJdbcDriverOption=value
```

In order to use this feature, the special session id PROVIDED_IN_DBNAME [245] needs to be specified at installation time

Also note that a session ID specified in the database name string will override the default provided in the connector configuration file. You can thus have a default session ID set for applications that can't specify it dynamically, and still allow other applications to connect with their own session ID variable.

7.4.10.2. Enabling SmartScale Routing

To enable SmartScale routing, configure the dataservice using the connector-smartscale [483] option. The session ID identification should also be specified by using the connector-smartscale-sessionid [484] option with one of the following values DATABASE [245], USER [245], CONNECTION [245] or PROVIDED_IN_DBNAME [245]:

```
connector-smartscale=true
connector-smartscale-sessionid={DATABASE|USER|CONNECTION|PROVIDED_IN_DBNAME}
```

In this mode, any client application can open a connection to the connector, and queries will automatically be redirected according to the SQL statement content.

In addition, all users that connect to the database must be granted the REPLICATION CLIENT privilege so that the user can compare the current replicator progress for session information. This can be granted using:

```
mysql> GRANT REPLICATION CLIENT ON *.* to app_user@'%'
```

7.4.10.3. Disabling SmartScale Routing

To disable SmartScale routing if it has been previously configured, change the connector-smartscale [483] property to false:

```
connector-smartscale=false
```

7.4.10.4. SmartScale Exploit

When using DATABASE [245] session ID, you can bypass session consistency to read from available Replicas by simply connecting to one database and reading from another. For example:

```
shell> mysql -u... -p... -P3306 db1
mysql> select * from db2.user
```

As long as no update is made on db1 in the meantime, the select will be executed on a Replica (if available)

7.4.11. Direct Routing

Auto Read/Write Splitting	Yes
Primary Selection	Automatically, by SQL examination
Replica Selection	Automatically, by SQL examination
QoS Compatibility	None

SmartScale Compatibility	None

Direct routing is a simplified form of SmartScale that uses a highly-efficient automated read/write splitting system, where of all auto-committed read-only transactions are routed to a pool of read-only Replica datasources. Unlike SmartScale, Direct routing pays no attention to the session state, or replicated data consistency.

This means that performing a write and immediately trying to read the information through a Direct routing connection may fail, because the Connector does not ensure that the written transaction exists on the selected Replica.

Direct routing is therefore ideal in applications where:

- Applications perform few writes, but a high number of reads.
- High proportion of reads on 'old' data. For example, blogs, stores, or machine logging information

Where applications are performing writes, followed by immediate reads of this data, for example conferencing and discussion systems, where reading stale data that has recently been written would create significant application failures, the solution should use should use SmartScale.

Read/Write splitting is supported through examination of the submitted SQL statement:

- If the statement contains `SELECT` and does not contain `FOR UPDATE`, the query is routed to an available Replica.
- If the statement starts `SHOW ...` then it is routed to a Replica.
- All other queries are routed to the Primary.

7.4.11.1. Enabling Direct Routing

To enable direct routing for a specific user, the `user.map` must be modified. Update the file with the `@direct` directive on every host running a connector. The connector will automatically read the changes after the file is saved.For example:

```
@direct sales
```

In this mode, any client application can open a connection to the connector, and queries will automatically be redirected according to the SQL statement content.

7.4.11.2. Limitations of Direct Routing

- Prepared statements must be enclosed within an explicit transaction boundary in order to be correctly routed to a Primary. For example:

```
BEGIN
PREPARE...
EXECUTE...
COMMIT
```

- Ephemeral objects (i.e. anything that is not replicated), will not be available on the Replica. Session variables are an excellent example of this.
- Stored procedures that update data in the database should never be called using a basic `SELECT` statement:

```
mysql> SELECT update_function('data');
```

Instead, add the `FOR UPDATE` keywords to ensure it is routed to the Primary:

```
mysql> SELECT update_function('data') FOR UPDATE;
```

7.4.12. SQL Routing

Auto Read/Write Splitting	No
Primary Selection	Manually, by SQL comments
Replica Selection	Manually, by SQL comments
QoS Compatibility	Supported
SmartScale Compatibility	Yes
Direct Compatibility	Yes

With SQL-based routing, the redirection of queries and operations through the Connector is controlled by hints on the *QoS* provided in the comments of individual statements. Note that this is explicit routing using SQL comments, not the automated read/write splitting supported by Direct or SmartScale routing.

Unless otherwise specified, statements will go to the current Primary to be executed, or whatever Replica is selected by the read-write splitting configuration, if enabled. To specify that a statement can be executed on the Replica, place a comment before the statement:

```
/* TUNGSTEN USE qos=RO_RELAXED */ SELECT * FROM TABLENAME
```

This style of comment indicates to the connector that the specific query that follows should go to a Replica. If unavailable, the query may still be executed on the Primary.

```
-- TUNGSTEN USE qos=RO_RELAXED
```

This style of comment indicates to the connector that all queries that follow should go to a Replica. If unavailable, any query may still be executed on the Primary.

> **Warning**
>
> If you force the Connector to send traffic to a Replica using qos=RO_RELAXED, then any write operations that follow will also go to the Replica until you tell the Connector to go back to the Primary by indicating qos=RW_STRICT. The application is fully responsible for where the traffic is routed to. if care is not taken, the application could send writes to a Replica this way which is unacceptable from a clustering perspective. All writes must go to the Primary or they will be lost to a non-authoritative node, and may corrupt the data.

The below forces all following queries to go to the Primary directly, effectively "turning off" reads from the Replica.

```
-- TUNGSTEN USE qos=RW_STRICT
```

> **Important**
>
> Please note that employing the `--` style will override any `/* */` comments.

7.4.12.1. Enabling SQL Routing

To enable SQL routing, add the following option to the `/etc/tungsten/tungsten.ini` and issue tpm update

```
property=selective.rwsplitting=true
```

Or, if using the staging method deployment, execute:

```
shell> tpm configure alpha \
    --property=selective.rwsplitting=true
```

7.4.12.2. Limitations of SQL Routing

- Read/write splitting must be handled entirely by the client application using the comments to specify which statements are Replica safe. Unless applications explicitly make the decision to write and read to the hosts using the comment system, operations may go to the wrong hosts.
- Prepared statements must be executed against the Primary.
- When testing the operation of the read/write splitting through the mysql client, ensure that command-line client is called using the `-c` option to ensure that comments are preserved:

  ```
  shell> mysql -c -h host -u tungsten -ppassword -P3306 test
  ```

7.4.13. Host-based Routing

Auto Read/Write Splitting	No
Primary Selection	Manually, by hostname/IP address
Replica Selection	Manually, by hostname/IP address
QoS Compatibility	None
SmartScale Compatibility	None

Host-based routing uses specific hostnames to provide the distinction between read and write availability within the connector. Two different hostnames and associated IP addresses need to be created on each connector host. Clients connecting to one host will be routed to the current Primary for writing, and connections to the other host will be redirected to a current Replica using the current load-balancing algorithm.

Once enabled, a client can open a connection directly to a Primary or Replica by connecting to the appropriate IP address or hostname. For example:

```
shell> mysql -hmaster.localhost
```

Will connect to the currently active Primary, while:

```
shell> mysql -hslave.localhost
```

Would connect to any currently available Replica.

7.4.13.1. Enabling Host-based Routing

To enable host-based routing requires both operating system and Connector based configuration changes:

1. The following steps must be made to the operating system configuration for each Connector host that will be configured within the dataservice:

   a. Add a second IP address to the host. This can be achieved either by adding or exposing a second physical ethernet device, or by exposing an alias on an existing hardware interface.

      For example, to add a second IP address to the physical `eth0` interface:

      ```
      shell> sudo ifconfig eth0:1 192.168.2.24
      ```

      To ensure this is retained during a restart, update your network configuration with the additional physical interface and IP address.

   b. Update the `/etc/hosts` file to reflect both addresses and appropriate hostnames. For example:

      ```
      192.168.2.20 host1 Primary.host1
      192.168.2.21 Replica.host1
      ```

   c. When using DNS to resolve addresses, the DNS should also be updated with hostnames to match those configured for each IP interface.

2. Update the `user.map` file on every host running a connector to reflect the the desired *`QoS`* for each hostname. The connector will automatically read the changes after the file is saved.

   ```
   @hostoption Primary.host1 qos=RW_STRICT
   @hostoption Primary.host2 qos=RW_STRICT
   @hostoption Primary.host3 qos=RW_STRICT
   @hostoption Replica.host1 qos=RO_RELAXED
   @hostoption Replica.host2 qos=RO_RELAXED
   @hostoption Replica.host3 qos=RO_RELAXED
   ```

Once configured, client applications must be configured to select the appropriate host based on the operation they are performing.

7.4.13.2. Limitations of Host-based Routing

- Prepared statements must be executed on the Primary.
- Smartscale cannot be enabled at the same time as host-based routing.
- QoS selection will not be honored.

7.4.14. Port-based Routing

Auto Read/Write Splitting	No
Primary Selection	Manually, by network port
Replica Selection	Manually, by network port
QoS Compatibility	None
SmartScale Compatibility	None

Port-based routing configures two independent ports that enable client applications to select whether to connect to a Primary or Replica based on the port they connect to. This method relies on the application choosing the correct port, automatic r/w splitting is not supported. Similar to host-based routing, port-based routing requires the client application to be modified to manually select the appropriate port.

Once enabled, a client can open a connection directly to a Primary or Replica by connecting to the appropriate port. For example:

```
shell> mysql -P3306
```

Will connect to the currently active Primary, while:

```
shell> mysql -P3307
```

will connect to a read-resource, ideally a Replica, but will revert to the Primary if no appropriate Replica is avaialble.

Note

The ports to be used for each connection type are configurable during installation.

7.4.14.1. Enabling Port-based Routing

Enabling port-based routing requires configuring the two ports that will accept queries. One port will be designated as the Primary port, one the read-only port, and queries will be automatically routed accordingly. For example:

```
connector-readonly-listen-port=3307
connector-listen-port=3306
```

Cient applications must be updated to support the two port interfaces and manually direct their queries to the appropriate Primary or Replica.

Using port routing in this way effectively marks all connections to the read-only port as behaving in a similar fashion to setting the connection QoS to *RO_RELAXED* [240].

7.4.15. Read-only Routing

It is possible to deploy a connector that has been configured to provide read-only access to the underlying databases on the standard port. This enforces read-only connectivity through this connector, regardless of any session or connector configuration options. This can be useful for a standalone connector, or a single connector within a dataservice.

This setting places the connector into *RO_RELAXED* [240] mode. The connector will choose the Primary if there is no available Replica. See Section 7.4.2, “Primary/Replica Selection” for more detail on Quality of Service modes.

To enable this functionality, configure the connector using the `connector-readonly` [481] option within you `/etc/tungsten/tungsten.ini` file, and execute tpm update:

```
connector-readonly=true
```

To enable this functionality on specific hosts only, only include this option in the `/etc/tungsten/tungsten.ini` on the hosts in which you want it applying.

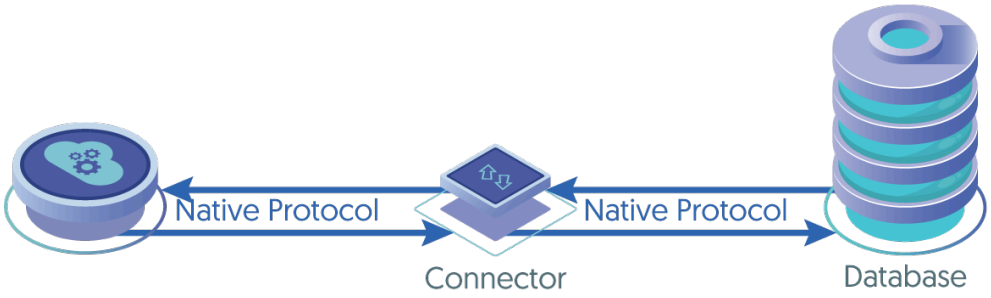
7.5. Using Bridge Mode

Bridge mode is the default connector routing method if no other routing method is explicitly stated. Bridge mode does not use the `user.map` file which reflects other changes to take a more secure default deployment. It will not be enabled in the following cases:

- The `connector-smartscale` [483] option is set to `true`.
- The `user.map` file contains `@direct` entries.
- The `user.map` file contains `@hostoption` entries.
- The `property=selective.rwsplitting` [443] connector option is set to `true`.

Bridge mode eliminates the need to create or define users and passwords within the `user.map` file. Instead, the connector acts as a router connecting the network sockets between client application and MySQL servers.

Figure 7.7. Tungsten Connector Bridge Mode Architecture



Bridge mode provides a simpler method for connecting clients to MySQL, but with reduced facilities, as outlined in the table below:

Feature	Proxy Mode	Bridge Mode
Primary/Replica Selection	Yes	Yes

Feature	Proxy Mode	Bridge Mode
Switch/Failover	Yes	Yes
Automatic Read/Write Splitting	Yes	No
Application-based Read/Write Splitting	Yes	Yes
Seamless Reconnects	Yes	No
Data Source Selection	Current data source is checked to confirm latency and affinity	Pass-through
Session KeepAlive	Yes	No

Bridge mode connections operate as follows:

1. Client opens network connection to Connector
2. Connector allocates a network buffer for the client network connection to the database server
3. Connector opens a network connection to a database server based on the connection parameters (Primary/Replica selection)
4. Connector allocates a network buffer for the database server to the client application
5. Connector directly attaches the network sockets together

Because the network sockets between the two sides are connected directly together, the following behavior applies to bridge mode connections:

- User authentication is handled directly by the database server, rather than through the `user.map` file.
- In the event of a failover or switch of the database servers, all active connections to the affected servers are closed.
- Smartscale and packet inspection to provide read/write splitting are not supported, since the Connector does not access individual packet data.

One key difference is in how Replica latency checking is handled:

- In Bridge mode, the latency is checked at connection time, then you will stick to the Replica for the connection lifetime (which can be shortened if the Replica goes offline).
- In Proxy mode, the latency is re-evaluated before each transaction, which can bring the connection to another Replica if the latency becomes too high during the life of the connection.

If you have long-lasting read-only connections that should not read from stale Replicas, then use Proxy mode.

If your connection lifetime is short (i.e make/break - one transaction then disconnect), or your application is not sensitive to reasonably outdated data for reads, then use Bridge mode and its optional read-only port.

7.5.1. Enabling Bridge Mode

Bridge Mode is enabled by default, however to explicity (re)enable Bridge Mode, the `connector-bridge-mode` [480] option to tpm must be set to `true`:

The default value is `true`, i.e. use the connector in bridge mode.

Bridge Mode can also be used with read-only database servers and affinity if required.

In addition to enabling and disabling Bridge Mode, the size of the buffer can also be set by using the *`bridgeServerToClientBufferSize`* and *`bridgeClientToServerBufferSize`* parameters. This configures the size of the buffer used to hold packet data before the packet is forwarded.

These properties can be set within the `/etc/tungsten/tungsten.ini` using the examples below:

```
property=bridgeServerToClientBufferSize=<Value in Bytes>
property=bridgeClientToServerBufferSize=<Value in Bytes>
```

The default size is 262144 bytes (256KB).

A buffer is opened in each direction for each connection made to the connector when operating in bridge mode.

The total memory allocated can be calculated using the formula:

```
(connections * (bridgeClientToServerBufferSize + bridgeServerToClientBufferSize)
```

For example, with the default settings, 20 simultaneous connections will require 10MB of RAM to service the buffers. With the default connector heap size the Connector should be able to handle up to 500 simultaneous connections.

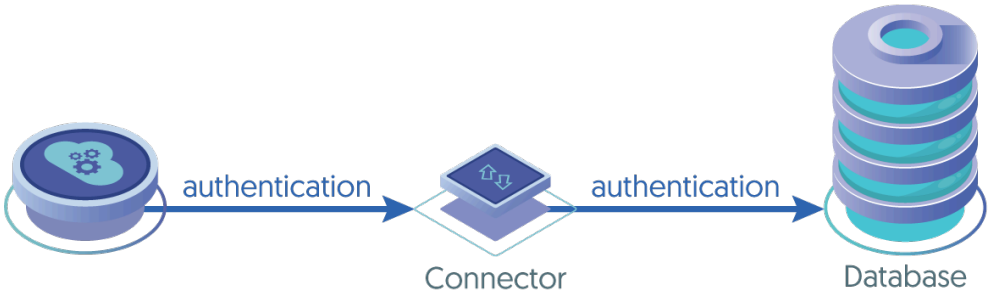
7.6. User Authentication

When configuring Tungsten Connector it is important to ensure that you have a `user.map` in place. The role of `user.map` is to define the usernames and passwords of users that will be connecting to the dataserver.

There is no authentication within the connector. Instead, the connector sends authentication information onto the dataserver. However, the MySQL network protocol exchanges a token between the client and the dataserver in order to authenticate the connection and is designed to prevent 'man in the middle' attacks.

Unfortunately, 'man in the middle' is exactly how Tungsten Connector operates, as the man in the middle to redirect queries to different dataservers as the list of active dataservers changes during the operation of a cluster. The authentication exchange cannot be reinitiated by the dataserver and client, so the Tungsten Connector performs this authentication exchange on behalf of the client using the user and password information from a special file called `user.map`.

Figure 7.8. Tungsten Connector Authentication



To get round this limitation, the connector operates as follows:

- Client opens a connection to the connector and authenticates.
- Connector connects to the datasource using the username supplied by the client, and the corresponding password stored within `user.map`.
- Database server returns the authentication token to the connector.
- Connectors sends the same authentication token back to the client.

This process gives the client application the authentication token required to enable it to communicate with the dataserver and the same token to be used by the connector.

For this system to work, a file, `user.map`, must exist on every connector installation, and it must contain the information for all users that will connect to the datasources from each client. Without this information, connectors will be unable to login on behalf of the client applications.

> **Important**
>
> All the users that require access to your MySQL servers through the Tungsten Connector *must* have an entry in the `user.map`. Without this information, the Tungsten Connector has no way of providing an onward connection to a MySQL server.

The `user.map` file primary role is to operate as the source for authentication information within the connector. However, through the use of additional flags and keywords, the file can also define the routing methods used by different users when connecting to datasources, and different dataservices.

7.6.1. `user.map` File Format

The current `user.map` file is located within the `tungsten-connector/conf` directory within an active installation. The file should be synchronized across all the servers within a dataservice. For more information on methods for keeping the file in sync, see Section 7.6.8, “Synchronizing `user.map` Data”.

The `user.map` file contains the usernames and passwords for each user that connects to the connector and the downstream MySQL server, and these entries are required for authentication. If an entry does not exist within `user.map` users will be unable to connect to MySQL through the connector.

> **Important**
>
> All the users that require access to your MySQL servers through the Tungsten Connector *must* have an entry in the `user.map`. Without this information, the Tungsten Connector has no way of providing an onward connection to a MySQL server.

The rules for the format of the file are as follows:

- Anything after a `#` (hash) symbol are interpreted as comments and ignored. For example:

```
# This line is a comment
```

- The following character cannot be used as the username, password or dataservice values:

```
space
|  # pipe
\t # tab
```

- Using the `-` (hyphen) character as a password indicates that there is an empty or no password ("") for the specified user.

The basic format for user entries within the `user.map` is:

```
username password servicename [affinity]
```

Where:

- *`username`* — the username to be used for authentication.

  The *`username`* also provides hooks into additional options; see `@script`, `@direct`, `@hostoption`.

- *`password`* — the password to be used for authentication.

  - Quotes ' and double quotes " are supported in the user.map password.
  - If there's a space in the password, the password needs to be surrounded with " or ':

    ```
    "password with space"
    ```

  - If there's one or several " or ' in the password without space, the password doesn't need to be surrounded

    ```
    my"pas'w'or"d"
    ```

  - If the password itself starts and ends with the same quote (" or '), it needs to be surrounded by quotes.

    ```
    "'mypassword'" so that the actual password is 'mypassword'.
    ```

  As a general rule, if the password is enclosed in either single or double quotes, these are not included as part of the password during authentication.

- *`servicename`* — the name of the dataservice or composite service to which this username/password apply.

- *`affinity`* — if the servicename is a composite service, the *`affinity`* identifies which service should be preferred for reads.

  The affinity feature routes both reads and writes when using a Composite Active/Active topology.

  The read affinity setting supports multiple dataservices with ordering and exclusion.

  You may fine-tune the affinity by specifying an ordered list of dataservice names.

  - The `affinity` string is an ordered list of dataservice names, separated by commas, where the first dataservice entry will be the one used by default. If the first dataservice in the list is not available, the connector will use the next one listed, and so forth.
  - Dataservices not specified in the list, if any, will be used last and randomly.
  - It is also possible to exclude one or more dataservices by adding a hyphen ("-") in front of the dataservice name.

For example, to configure the user `sales` with the password `secret` to use MySQL servers within the `alpha` dataservice:

```
sales secret alpha
```

To configure a user that has no password:

```
sales - alpha
```

To configure a user within a composite service:

```
sales secret global
```

To configure a user within a composite service, preferring the `east` service for read-only connections:

```
sales secret global east
```

The deployment used in the examples below consists of a composite dataservice `global`, with four member dataservices (i.e. one cluster per site): `east`, `west`, `north` and `south`.

For example, to exclude site `south` from servicing read requests:

```
sales secret global east,west,north,-south
```

The above affinity string "`east,west,north,-south`" will try `east`, then `west`, then `north`. If none of the first three are available, a connection request would not succeed since `south` has been excluded by the negation ("-").

In the following example, dataservices `north` and `south` would be available as random candidates if the first two (`east` and `west`) were unavailable:

```
sales secret global east,west
```

7.6.2. Dual-Paswords and MySQL 8

> **Note**
>
> This feature is ONLY available when using MySQL v8.0.14+ and any tungsten version in bridge mode with no additional configuration.
>
> The use of dual passwords combined with Proxy Mode is only supported with Tungsten version 7.1.0+, by following the procedure outlined below.

Starting from MySQL 8.0.14+ it is possible to configure user accounts to have two passwords. This allows for easy password rotation without downtime by allowing you to set a new password for the user whilst retaining the previous password, either indefinitly or for a period of time. For more information on this feature, view the MySQL documentation here

From version 7.1.0 of Tungsten Connector it is now possible to make use of this feature when using Proxy based authentication (If you are using Bridge mode, then this feature can be used with any Tungsten release)

To make use of this feature, first of all you need to configure the account with the new password, using syntax along the lines of:

```
mysql> ALTER USER app_user@'%' IDENTIFIED BY 'second_pass' RETAIN CURRENT PASSWORD;
mysql> flush privileges;
```

Next, in the connector `user.map` file, you add the additional password for the given user, adding an entry with the following syntax format:

```
@dual <user>[@<host>] <second_pass>
```

Where <user> is the exact username that must have been defined earlier in the `user.map` as a regular entry. (See Section 7.6.1, "`user.map` File Format"

Note that an optional host can be specified next to the user, separated with a @ sign. In this case, the dual password must be defined (and can be different) for each user@host tuple

No connector restart will be needed, the `user.map` will be reloaded automatically after being updated

If/when you then choose to discard the old password, first of all edit the `user.map` and update the original entry swapping out the oldPassword for the newPassword:

```
<user> <second_pass> <service> <affinity>
```

Then issue the syntax in MySQL to discard the old pasword:

```
mysql> ALTER USER app_user@'%' DISCARD OLD PASSWORD;
```

You can then remove the @dual entry from the `user.map` when convenient.

7.6.3. `user.map` Direct Routing

To enable direct reads, as defined within Section 7.4.11, "Direct Routing", the entries for the user within `user.map` must be prefixed using the `@direct`. For example:

```
@direct sales
```

Note that the standard user, password and service must be defined:

```
sales secret alpha
@direct sales
```

For limitations of direct routing, see Section 7.4.11.2, "Limitations of Direct Routing".

7.6.4. `user.map` Host Options

The `user.map` provides a configuration point to enable the connector to support host options that enable you to define qualities of service against specific hosts, as configured according to the guidance within Section 7.4.13, “Host-based Routing”.

When configuring the host options, the hostnames must have previously been defined and be resolvable.

The *QoS* within `user.map` has the following format:

```
@hostoption hostname QoS
```

For example, to enable *RW_STRICT* [240] on one host and *RO_RELAXED* [240] on the other:

```
@hostoption readwrite.master qos=RW_STRICT
@hostoption readonly.master qos=RO_RELAXED
```

7.6.5. `user.map` Updates

When the `user.map` file is updated:

- The Tungsten Connector should automatically identify that the file has been changed and reload the file, updating the user map.
- To manually force the users to be updated, for example, if the `user.map` uses the `@script`, use the tungsten flush privileges command:

  ```
  mysql> tungsten flush privileges;
  +--------------------------------+
  | Message                        |
  +--------------------------------+
  | user.map reloaded successfully |
  +--------------------------------+
  1 row in set (0.00 sec)
  ```

  If you are using the `application-readonly-port` [474] option, this command must be run through both ports. Alternatively, you can trigger a simultaneous flush by running:

  ```
  shell> touch /opt/continuent/tungsten/continuent-connector/conf/user.map
  ```

- When using `@direct` entries in `user.map`, the connector may need to be restarted using connector restart:

  ```
  shell> connector restart
  ```

  This will disconnect all connected clients, but the connector itself should be unavailable only for a short time.

- When the connector installation is updated using tpm, for example during an upgrade, the `user.map` and `dataservices.properties` are automatically copied into the new installation automatically and do not need to be manually copied or update.

  To perform an update without automatically copying the `user.map` file include the `connector-delete-user-map` [480] option in your `/etc/tungsten/tungsten.ini` file.

7.6.6. Generating `user.map` Entries from a Script

The content of the `user.map` file can be generated automatically, for example by automatically extracting information from a separate service, such as LDAP, NIS or others. To specify the script that will generate the information, the `@script` directive can be used within the `user.map`:

```
@script /opt/continuent/share/usermap
```

When using the script method:

- The information must be generated in the same format as for standard entries, i.e.:

  ```
  username password servicename
  ```

- If the script generates multiple entries with the same name, the later output will overwrite the previous entry.
- Multiple `@script` directives can be specified. Each will be processed in turn.
- If a generated list of usernames changes due to the scripts, the connector must be manually forced to reload the usermap using tungsten flush privileges on a connector connection. If you are using the `application-readonly-port` [474] option, this command must be run through both ports. Alternatively, you can trigger a simultaneous flush by running:

  ```
  shell> touch /opt/continuent/tungsten/continuent-connector/conf/user.map
  ```

- If the file is placed into `/opt/continuent/share` then the script will be retained during upgrades through tpm update.
- If a script within the `@script` fails to be executed correctly, or generates no user entries, the connector will fail to start.

The script itself can be relatively simple, the standard output of the command must contain the user entries to be included in `user.map`. Standard error is ignored.

For example:

```
#!/bin/bash

echo 'app_user password dsone'
```

This generates a simple user entry.

7.6.7. Encrypting `user.map` Data

The `user.map` file allows you to use an encryted version of the file by using the `@script` directive. Here is an example of how you can decrypt a file and return the results to `user.map`.

1. Change to a directory outside of the currently installed Tungsten

   Do this to ensure that the OpenSSL key and encrypted file are available after upgrades and other operations.

   ```
   shell> cd /opt/continuent/share
   ```

2. Create an OpenSSL key

   In this example we will use a 1024-bit RSA private key to do the encryption and decryption. There are many options for encrypting and decrypting files but this documentation will not explore those. The same process will work with other encryption techniques. You must ensure that the decryption command runs without user input.

   ```
   shell> openssl genrsa -out usermap.pem 1024
   ```

3. Create the encrypted file of `user.map` entries:

   ```
   tungsten secret nyc_sjc sjc
   tungsten_sjc secret sjc
   tungsten_nyc secret nyc
   ```

   Create an encrypted version of the file:

   ```
   shell> openssl rsautl -encrypt -inkey usermap.pem -in user.map.entries -out user.map.entries.ssl
   ```

4. Test decryption of the encrypted file:

   ```
   shell> openssl rsautl -decrypt -inkey usermap.pem -in user.map.entries.ssl
   ```

   This should return the unencrypted `user.map`:

   ```
   tungsten secret nyc_sjc
   tungsten_sjc secret sjc
   tungsten_nyc secret nyc
   ```

5. Update the installed and configured `tungsten-connector/conf/user.map` file:

   ```
   ...
   # Examples:
   # user tungstenuser has password secret and uses 'sjc_nyc' composite
   # data service, but prefers nyc site for reading:
   #   tungstenuser secret sjc_nyc nyc
   ```

   Now add a `@script` directive to point to the encrypted file and certificate:

   ```
   @script openssl rsautl -decrypt -inkey /opt/continuent/share/usermap.pem -in /opt/continuent/share/user.map.entries.ssl
   ...
   ```

6. Repeat the process on each host. The `user.map` file will be copied to the new version when you upgrade Tungsten so this process must only be completed once per host.

7.6.8. Synchronizing `user.map` Data

Tungsten Cluster does not automatically synchronize information contained within the `user.map` across all the nodes within the cluster. The connector does not identify, track, or update `user.map` content when it sees password changes.

Instead, the file must be updated by hand, through the `@script` directive, and synchronized across multiple hosts either manually or by using a script. For example:

```
#!/bin/bash
```

```
for HOST in host1 host2 host3 host4
do
rsync /opt/continuent/tungsten/tungsten-connector/conf/user.map \
    $HOST:/opt/continuent/tungsten/tungsten-connector/conf/user.map
done
```

If `@script` directives are used, the corresponding scripts must also be included within this synchronization step.

> **Important**
>
> All servers within the cluster must have an identical `user.map` configuration. Failure to have a synchronized configuration may lead to clients being unable to connect to the connector and database servers.

7.6.9. `user.map` Limitations

The `user.map` configuration has the following limitations:

- Users must be defined for each dataservice; if there is a common user that can be used in any of your configured dataservices, there must be an individual line for each dataservice. For example:

  ```
  sales secret alpha
  sales secret beta
  sales secret gamma
  ```

- When using `user.map` with multiple dataservices, additional data services must exist in `dataservices.properties`. Only add the physical data services you would like to work with. Any composite data services will automatically be discovered. The connector must be restarted once the data services have been added.
- Specifying a composite dataservice that has not been defined will raise an error.
- If the `user.map` contains multiple entries for the same user, only the last entry will be used.

7.6.10. Host-based Authentication

In addition to the explicit user/host based authentication support, the connector also includes general host-based authentication that allows client connections only from specific hosts.

Host-based authentication is not enabled in the default installation. To enable it, create a file `authorized_hosts` within the `tungsten/tungsten-connector/conf/` directory of the active installation. The connector will then need to be restarted before host-based authentication is enabled.

> **Important**
>
> The `authorized_hosts` file is not automatically distributed during deployment and updates. The file must be manually copied to other hosts.
>
> If the content of the `authorized_hosts` file is changed, the connector configuration must be reloaded using the connector reconfigure [334] command for changes to take effect.

If the file exists, host-based authentication is enabled. If it is empty, all client connections are denied. The format of the file is that each line defines the host address and netmask in CIDR format. For example:

```
192.168.1.0/24
```

Enables connectivity from all hosts in the range 192.168.1.0-192.168.1.255.

Optional username(s) can be added before the host entries in order to apply restriction to a given (set of) user(s).

When no user is specified, restrictions will be applied on the given addresses for all users.

When the line starts with a username, or a comma separated list of users, then a space, then hostname(s) in CIDR format, only the users connecting from the given IP address(es) will be accepted. Other connection attempts will be rejected with `"ERROR 1044 (42000): Authentication failed"` (when SSL is disabled).

> **Note**
>
> Limitations when SSL to the connector is enabled: When both bridge mode and SSL are enabled, it will NOT be possible to use this new functionality; this limitation is due to the connector not having access to the data (including username) being transferred between client and server, for the very reason it is encrypted. In other configurations including SSL, and due to the way the MySQL protocol works, there is no possibility to cleanly reject SSL connections with the `"Authentication failed"` error.
>
> Denied connections will trigger an error in the SSL handshake phase, leading to error messages on the client application side such as (depends on the client application itself): `"ERROR 2026 (HY000): SSL connection error: er-`

ror:00000001:lib(0):func(0):reason(1)" or "ERROR 2013 (HY000): Lost connection to MySQL server at 'reading authorization packet', system error: 0"

7.7. Testing Connectivity Via the Connector

To test connectivity to the database server through the Tungsten Connector there are a variety of methods to choose from depending on the selected configuration.

7.7.1. Testing Connectivity in Bridge Mode

When using Bridge mode (the default at install), all requests are routed to the Primary by default. To test query routing, run the following query when connected through the Connector:

```
Route to the Primary:
mysql> select @@hostname;
```

In Bridge mode, the only way to verify that reads are being directed to replicas is to establish a read-only port and execute queries through it to force the *QoS RO_RELAXED* [240].

First, ensure that your INI file has the following option, then run tpm update

```
connector-readonly-listen-port=3307
```

To test, ensure you connect to the specified read-only port:

```
Route to a Replica:
shell> mysql -h... -P3307
mysql> select @@hostname;
```

7.7.2. Testing Connectivity in Proxy Mode with No R/W Splitting Enabled

To test Connector query routing in Proxy mode, you may use the URL-based R/W splitting to test query routing:

```
Route to the Primary:
shell> mysql -h... -Dtest@qos=RW_STRICT -e "select @@hostname;"

Route to a Replica:
shell> mysql -h... -Dtest@qos=RO_RELAXED -e "select @@hostname;"
```

7.7.3. Testing Connectivity in Proxy Mode with R/W Splitting Enabled (SmartScale or @direct)

To test Connector query routing in Proxy mode when either SmartScale or @direct read/write splitting has been enabled, you may use the following:

```
Route to the Primary:
mysql> select @@hostname for update;

Route to a Replica:
mysql> select @@hostname;
```

7.8. Connector Operational States

During operation, the connector goes through a number of different states and state transition during specific events. The default mode is the `ONLINE` state, where the connector operates as configured.

During operation, all configured connectors within the dataservice remain in contact with the manager, see Section 7.9, “Connector/Manager Interface” for more information.

Supported states by the Connector are:

- **`ONLINE`** State

  The Connector operates as configured, redirecting connections to the corresponding Primary or Replica.

- **`ON HOLD`** State

  The connector will enter this state as soon as the manager connection is lost if *`delayBeforeOnHoldIfNoManager=0`* (the default), or after the specified delay.

  In this state, new connection requests are paused (client applications trying to connect will see what appears to be a “hang”) while existing connections will continue as normal until the Offline state is reached

This property can be set using the tpm flag `connector-delay-before-onhold`.

- `OFFLINE` State

  The connector enters the offline state *`delayBeforeOfflineIfNoManagerseconds`* after the connection to the manager is lost.

  This property can be set using the tpm flag `connector-delay-before-offline`. The default value is 30, which means going to the offline state after 30 seconds.

  When entering the offline state, the connector terminates all existing connections, and rejects all new connections requests (client applications will receive an error when trying to connect).

The state of a connector can be modified by using the router command within cctrl. This can be used to manually place the connector into online or offline states. For example, to put a connector online the full host and process ID must be used:

```
cctrl> router connector@host1[22476] online
```

Wildcards can be used to enable or disable all the hosts. For example, to place all connectors online:

```
cctrl> router * online
```

While in *`AUTOMATIC`* policy mode, connectors will automatically be placed online if they have entered the `OFFLINE` state automatically as part of a failover. If the routers have been manually placed offline, routers must be manually placed back online.

While in the `ONLINE` state, the connector behaves and alters it's operation according to the following states and events:

7.8.1. Connections During Automatic Failure/Failover

When an automatic failure or failover is identified, for example when the dataservice is in the *`AUTOMATIC`* policy mode, and the Primary is automatically switched to a new host, the following sequence occurs:

1. All connections to the failed datasource are terminated immediately. This ensures that running transactions or operations are terminated by the database server.
2. Connections to clients will remain open and be reconnected transparently, providing they are not within a transaction. For more information, see Section 7.8.3, “Connections During Connection Failures”.

   Only if there is a problem with the connection or an I/O error will the problem be forwarded to the clients.

As with a direct database connection, the client application should handle the reconnection to the Connector, which will be then be redirected to the corresponding Primary or Replica datasource.

7.8.2. Connections During Manual Switch

When a manual switch operation has been initiated, the Connector follows this sequence:

1. New connection attempts to the old datasource are suspended; this gives the impression of a 'hung' connection that must be managed by the client application through the normal timeout procedure.
2. Existing connections to the datasource are terminated under two conditions:
   - As the connections are naturally closed.
   - Open connections are forcibly disconnected after the timeout specified by the `waitForDisconnectTimeout` parameter. By default, this is 5 seconds. To eliminate waiting, the *`waitForDisconnect`* parameter can be set to `false`.

   Once either condition has been met, any remaining connections are closed.
3. New connections (including re-connections) are enabled, and will be routed to the appropriate Primary or Replica.

Client applications should be configured to reconnect to the connector with an interval larger than the disconnect timeout within the connector. This will ensure that the client reconnects when the connector is able to accept the new connection.

7.8.3. Connections During Connection Failures

In the event of a connection failure between a running datasource and the connector, and providing the connection is deemed idle, the connector will transparently reconnect to the failed datasource when the following conditions have been met:

- The connection is not executing any requests.
- The connection is not in the middle of a transaction.

- No temporary tables have been created during this connection.

If all three conditions are met, a new connection will be opened. Connections between the client and the connector will be unaffected.

This option is enabled by default. To disable transparent reconnections, use `connector-autoreconnect=false` [479] option to tpm during installation.

7.8.4. Other Errors

The Connector attempts to emulate and effectively represent any errors raised by the datasource to which the connector has routed the client connection.

- The Tungsten Connector uses the Tanuki Java Service Wrapper to manage the running process. If the Connector process fails, the service wrapper will automatically restart it. If the connector fails repeatedly, attempts to restart will be stopped. The status and reason for these failures can be tracked by examining the `connector.log` log file.

  Connected client applications will be terminated, but should be able to reconnect once the Connector has been restarted.

- Database errors, including invalid statements, operations, or security failures, will be represented identically by the Connector to any clients.

7.8.5. Connector Keepalive

Connections to MySQL servers can automatically time-out according to the `wait_timeout` variable configured within the MySQL server.

To prevent these connections being automatically closed, the connector can be configured to keep the connection alive by submitting a simple `SELECT` statement (actually `SELECT 'KEEP_ALIVE'`) periodically to ensure that the MySQL timeout is not reached and the connection closed.

Two parameters configure the keepalive functionality:

- *`connection.keepAlive.interval`*

  The interval used to check for idle connections. If set to a value of 0, the keep alive check is disabled. Any value greater than zero is the interval check period in seconds.

- *`connection.keepAlive.timeout`*

  The keep-alive statement is submitted if the time since the last activity reaches this timeout value.

The default setting for both parameters is *`autodetect`*.

When set to *`autodetect`* default, the values are automatically calculated by the connector computing suitable values based on the `wait_timeout` value configured in the MySQL server.

```
connection.keepAlive.interval = (int) Math.floor(wait_timeout * 0.10);
connection.keepAlive.timeout = (int) Math.floor(wait_timeout * 0.7);
```

These calculations cannot be modified, but the properties can be explicitly set by using the `property` [443] option to explicitly set the property through tpm, for example, the following can be added to you `/etc/tungsten/tungsten.ini`:

```
property=connection.keepAlive.interval=30
```

Warning

Please note that Connector Keepalive is not compatible with Bridge mode.

In Bridge mode, the client session is directly connected to the MySQL server at the TCP level, literally forwarding the client's packet to the server. This means that closing connections is the responsibility of the MySQL server based on the configured wait_timeout value, not the Connector.

7.8.6. Connector Change User as Ping

When using PHP with connection pooling enabled, the change user command is used to ping a connection within a pool to ensure that the connection is open and active before using it. The Tungsten Connector uses JDBC to connect to MySQL, which does not support the change user protocol option.

To provide an alternative to this for PHP applications communication through the connector, the connector can be configured to respond to the `COM_CHANGE_USER` command from a client application. Rather than performing the change user operation, instead the connector will respond to the client with an acknowledgement, emulating the ping operation.

This operation is disabled by default, and must be explicitly enabled. This can be achieved by setting the correct property value, `treat.com.change.user.as.ping`, to `true` within you `/etc/tungsten/tungsten.ini`:

```
property=treat.com.change.user.as.ping=true
```

7.9. Connector/Manager Interface

The connector remains in constant communication with the Tungsten Manager during operation. This enables the connector to respond to failures and errors, whether automatically identified, or manually triggered. For example, when a manual switch operation occurs, the manager communicates this information to all of the connectors. Each connector then responses according to the rules outlined in Section 7.8, “Connector Operational States”.

The connector remains in communication with one, and only one, manager at a time. If the manager becomes unavailable, the connector tries to communicate with another manager within the dataservice.

Communication from the manager to the connectors are made in parallel using multiple threads, this ensures that all connectors are made aware of a change in the topology of the cluster at the same time, rather than a round-robin or staged distribution. When a change has been requested, the manager waits for a response from the cluster before confirming that switch and operational change has taken place.

Communication between managers and the connectors is handled on ports 11999 (managed by `router-gateway-port` [513]) and 12000 (managed by `mgr-rmi-remote-port` [501]). The connection is used to exchange cluster status and individual datasource availability as identified by the manager so that decisions about active connections can be effectively made by the connector.

In the event that the connection between the connector and the manager is broken, the connector enters a failsafe mode called `ON HOLD`. In this state, connections to and from the connector and datasources will continue as normal until a timeout, configured by the *`delayBeforeOfflineIfNoManager`* property, is reached. By default, this timeout is 600 seconds (10 minutes). Once the timeout has been reached, the connector reaches the `OFFLINE` state.

All of the information about the dataservice, including the other nodes, topology and individual node states and roles are entirely determined by the Connector by requesting this information from the Manager. No on-disk record or description is stored, created, or read by the Connector. When the Connector is first started, it connects to a manager and requests the full cluster configuration.

If the Connector cannot communicate with a manager, the connector remains in the `OFFLINE` state until a manager can be reached.

7.10. Connector Command-Line Interface

The connector command is used for various operations that affect the Tungsten Connector, for example, starting and stopping the Tungsten Connector, getting status, updating and debugging.

When using the connector command-line tool, the following sub-commands are available:

Table 7.2. Connector Command Line Sub-Commands

Option	Description
`client-list` [333]	Return a list of the current client connections through this connector.
`cluster-status` [333]	Return the cluster status, as the connector currently understands it. This is the command-line alternative to the inline cluster status command.
`condrestart` [333]	Restart only if already running
`console` [334]	Launch in the current console (instead of a daemon)
`drain [seconds]` [334]	An alias for graceful-stop. As of v7.0.0
`dump` [334]	Request a Java thread dump (if connector is running)
`graceful-stop [seconds]` [334]	Stops the connector gracefully, allowing outstanding open connections to finish and close before the connector process is stopped. All new connection requests are denied. The Connector will shut down as soon as there are no active connections. [seconds] is an integer specifying the optional time to wait before terminating the connector. Specifying no value for seconds will cause the Connector to wait indefinitely for all connections to finish. Specifying zero (0) seconds will cause the Connector to shut down immediately without waiting for existing connections to complete gracefully. As of v7.0.0, connector drain is available as an alias for connector graceful-stop.
`install` [334]	Install the service to automatically start when the system boots
`mode` [334]	Displays the mode the connector is running in, either "proxy" or "bridge"
`reconfigure` [334]	Reconfigure the connector by forcing the connector to reread the configuration, including the configuration files and `user.map`.

Option	Description
remove [334]	Remove the service from starting during boot
restart [334]	Stop connector if already running and then start
start [334]	Start in the background as a daemon process
status [334]	Query the current status
stop [335]	Stop if running (whether as a daemon or in another console) Optional timeout in seconds can be provided (From release 6.1.19 only)

For more information, please see Section 9.9, "The connector Command".

7.11. Connector Inline Command Interface

When connected to a service through Tungsten Connector, the connection has access to a number of specialized commands that can be executed

> **Important**
>
> When using Bridge Mode, these commands will not be available because you are connected directly to the database server.

Table 7.3. Inline Interface Commands

Option	Description
tungsten cluster status	Displays a detailed view of the information the connector has about the cluster
tungsten connection count	Display the current number of active connection to each datasource
tungsten connection status	Displays information about the connection status for the last statement executed
tungsten flush privileges	Reload the user.map file and update the user credentials
tungsten gc	Executes the connector garbage collector to free memory
tungsten help	Shows help description each statement
tungsten mem info	Display the memory usage information for the connector
tungsten show processlist	List all active queries on this connector instance
tungsten show variables	Display the connector configuration options currently in use

7.11.1. Connector tungsten cluster status Command

Shows the current cluster status, as far as the connector is aware. The output consists of a table showing dataservices and hosts and current status and role information:

```
mysql> tungsten cluster status;
+-----------------+-------+-------+--------+---------+----------------+----------------------+-------------------------+
| dataServiceName | name  | host  | role   | state   | appliedLatency | activeConnectionCount | connectionsCreatedCount |
+-----------------+-------+-------+--------+---------+----------------+----------------------+-------------------------+
| alpha           | host1 | host1 | slave  | SHUNNED | 60.0           | 0                    | 0                       |
| alpha           | host2 | host2 | master | ONLINE  | 0.0            | 1                    | 2                       |
| alpha           | host3 | host3 | slave  | SHUNNED | 61.0           | 0                    | 0                       |
+-----------------+-------+-------+--------+---------+----------------+----------------------+-------------------------+
3 rows in set (0.01 sec)
```

The output fields are as follows:

- *dataServiceName*

  The name of the service. In connectors configured with multiple services, including composite clusters, there will be an entry for each host/servicename combination.

- *name*

  The name of the host within the service.

- *host*

The hostname on which the service is running.

- *role*

  The current role for the host.

- *state*

  The current state for the host within the service.

- *appliedLatency*

  The applied latency of transactions; for Primaries the difference between the commit time and extraction, in a Replica, the difference between commit time in the Primary and commit time in the Replica.

- *activeConnectionCount*

  Count of the current number of active connections.

- *connectionsCreatedCount*

  Count of the number of connections created since the connector has been started.

7.11.1.1. Connector connector cluster status on the Command-line

The tungsten cluster status is also available from the command-line through the connector command. To use this command from the command line, place the cluster-status command on the command-line. For example:

```
shell> connector cluster-status
Executing Tungsten Connector Service --cluster-status ...
+--------------+--------------------+-------------+--------------+--------+--------+-----------------------------------------+----------------------+----
| Data service | Data service state | Data source | Is composite | Role   | State  | High water                              | Last shun reason     | App
+--------------+--------------------+-------------+--------------+--------+--------+-----------------------------------------+----------------------+----
| alpha        | ONLINE             | demo-c11    | false        | master | ONLINE | 11(mysql-bin.000033:0000000000000523;-1) |                      | 752
| alpha        | ONLINE             | demo-c12    | false        | slave  | ONLINE | 11(mysql-bin.000033:0000000000000523;-1) | SHUNNED-FOR-RECOVERY | 752
+--------------+--------------------+-------------+--------------+--------+--------+-----------------------------------------+----------------------+----

Done Tungsten Connector Service --cluster-status
```

The information can also be output in JSON format by adding the `-json`:

```
shell> connector cluster-status -json
Executing Tungsten Connector Service --cluster-status -json...
{
  "alpha" : {
    "demo-c11" : {
      "name" : "demo-c11",
      "dataServiceName" : "alpha",
      "host" : "demo-c11",
      "activeConnectionCount" : 0,
      "alertMessage" : "",
      "alertStatus" : "OK",
      "alertTime" : 1496310617376,
      "appliedLatency" : 7528198.0,
      "available" : true,
      "callableStatementsCreated" : 0,
      "childType" : "UNDEFINED",
      "composite" : false,
      "compositeMember" : null,
      "connectionsCreated" : 0,
      "container" : false,
      "dataServerHost" : "demo-c11",
      "dataSourceRole" : "&mas_lc;",
      "driver" : "org.drizzle.jdbc.DrizzleDriver",
      "enabled" : 0,
      "executable" : false,
      "isAvailable" : true,
      "key" : "demo-c11",
      "lastError" : "",
      "lastShunReason" : "",
      "lastUpdate" : 1511778569541,
      "&mas_lc;" : true,
      "&mas_lc;ConnectUri" : "",
      "precedence" : 99,
      "preparedStatementsCreated" : 0,
      "relativeLatency" : 7528879.0,
      "relay" : false,
      "role" : "&mas_lc;",
      "sequence" : {
```

```
      "wrapAround" : 9223372036854775807,
      "generation" : 0,
      "identity" : "b70818d0-6506-4fd1-8e23-c303ccc2bcba",
      "currentValue" : 0,
      "lastValue" : 0,
      "wrapped" : false
    },
    "&slv_lc;" : false,
    "standby" : false,
    "state" : "ONLINE",
    "statementsCreated" : 0,
    "type" : "DATASOURCE",
    "updateTimestamp" : 1511778569541,
    "url" : "jdbc:mysql:thin://demo-c11:3306/${DBNAME}?jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull&tinyInt1isBit=false&allowMultiQu
    "vendor" : "mysql",
    "vipAddress" : "",
    "vipInterface" : "",
    "vipIsBound" : false,
    "witness" : false
  },
  "demo-c12" : {
    "name" : "demo-c12",
    "dataServiceName" : "alpha",
    "host" : "demo-c12",
    "activeConnectionCount" : 0,
    "alertMessage" : "",
    "alertStatus" : "OK",
    "alertTime" : 1511779000635,
    "appliedLatency" : 7528198.0,
    "available" : true,
    "callableStatementsCreated" : 0,
    "childType" : "UNDEFINED",
    "composite" : false,
    "compositeMember" : null,
    "connectionsCreated" : 0,
    "container" : false,
    "dataServerHost" : "demo-c12",
    "dataSourceRole" : "&slv_lc;",
    "driver" : "org.drizzle.jdbc.DrizzleDriver",
    "enabled" : 0,
    "executable" : false,
    "isAvailable" : true,
    "key" : "demo-c12",
    "lastError" : "--",
    "lastShunReason" : "SHUNNED-FOR-RECOVERY",
    "lastUpdate" : 1511779000635,
    "&mas_lc;" : false,
    "&mas_lc;ConnectUri" : "",
    "precedence" : 99,
    "preparedStatementsCreated" : 0,
    "relativeLatency" : 7528878.0,
    "relay" : false,
    "role" : "&slv_lc;",
    "sequence" : {
      "wrapAround" : 9223372036854775807,
      "generation" : 0,
      "identity" : "a12e5122-d1c3-4d8b-98fe-d58241b86bf9",
      "currentValue" : 4,
      "lastValue" : 3,
      "wrapped" : false
    },
    "&slv_lc;" : true,
    "standby" : false,
    "state" : "ONLINE",
    "statementsCreated" : 0,
    "type" : "DATASOURCE",
    "updateTimestamp" : 1511779000634,
    "url" : "jdbc:mysql:thin://demo-c12:3306/${DBNAME}?jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull&tinyInt1isBit=false&allowMultiQu
    "vendor" : "mysql",
    "vipAddress" : "",
    "vipInterface" : "",
    "vipIsBound" : false,
    "witness" : false
  }
 }
}
Done Tungsten Connector Service --cluster-status -json
```


7.11.2. Connector tungsten connection count Command

Displays the current list of open connections for all hosts within the cluster.

```
mysql> tungsten connection count;
```

```
+--------------+-------------+-------------------------------------------------------------------+
| Data service | Data source | Active Connections                                                |
+--------------+-------------+-------------------------------------------------------------------+
| alpha        | ct-multi1   | [internal(OPEN) DIRECT TO ct-multi1@alpha(master:ONLINE) STATUS(OK)] |
| alpha        | ct-multi2   | []                                                                |
| alpha        | ct-multi3   | []                                                                |
+--------------+-------------+-------------------------------------------------------------------+
3 rows in set (0.00 sec)
```

7.11.3. Connector tungsten connection status Command

Displays the current connection status, for all connections, including indicating whether the connection is using SSL on either the incoming (client) or outgoing (MySQL) connection:

```
mysql> tungsten connection status;
+-----------------------------------------------------------------------------------+
| Message                                                                           |
+-----------------------------------------------------------------------------------+
| ct-multi1@alpha(master:ONLINE) STATUS(OK), QOS=RW_STRICT SSL.IN=false SSL.OUT=false |
+-----------------------------------------------------------------------------------+
1 row in set (0.05 sec)
```

7.11.4. Connector tungsten flush privileges Command

Forces a reload of the `user.map` file to update the user privileges configured within the file. Only forces an update of the connector to which the client is connected.

```
mysql> tungsten flush privileges;
+-------------------------------+
| Message                       |
+-------------------------------+
| user.map reloaded successfully |
+-------------------------------+
1 row in set (0.00 sec)
```

7.11.5. Connector tungsten gc Command

Forces a Java garbage collection for the connector, recovering memory. An example of the memory used, garbage collection, and resulting memory usage below.

```
mysql> tungsten mem info;
+----------------------+-----------------------------------------------------+
| JVM Memory statistics | Value in bytes                                      |
+----------------------+-----------------------------------------------------+
| Peak Thread Count     | 18                                                  |
| Heap Memory           | init = 67108864(65536K) used = 17437496(17028K)     |
|                       | committed = 64880640(63360K) max = 259522560(253440K) |
| Non-heap Memory       | init = 24313856(23744K) used = 13970024(13642K)     |
|                       | committed = 24313856(23744K) max = 224395264(219136K) |
| Thread Count          | 16                                                  |
+----------------------+-----------------------------------------------------+
4 rows in set (0.05 sec)

mysql> tungsten gc;
+------------------------------+
| Message                      |
+------------------------------+
| Garbage collection successful |
+------------------------------+
1 row in set (0.41 sec)

mysql> tungsten mem info;
+----------------------+-----------------------------------------------------+
| JVM Memory statistics | Value in bytes                                      |
+----------------------+-----------------------------------------------------+
| Peak Thread Count     | 18                                                  |
| Heap Memory           | init = 67108864(65536K) used = 4110088(4013K)       |
|                       | committed = 64946176(63424K) max = 259522560(253440K) |
| Non-heap Memory       | init = 24313856(23744K) used = 13970024(13642K)     |
|                       | committed = 24313856(23744K) max = 224395264(219136K) |
| Thread Count          | 16                                                  |
+----------------------+-----------------------------------------------------+
4 rows in set (0.00 sec)
```

7.11.6. Connector tungsten help Command

Displays the list of currently supported commands within the connector inline interface:

```
mysql> tungsten help;
+-------------------------------------------------------------------------------+
| Message                                                                       |
+-------------------------------------------------------------------------------+
| tungsten connection status:                display information about the |
|                                            connection used for the last  |
|                                            request ran                   |
| tungsten connection count:                 gives the count of current    |
|                                            connections to each one of the|
|                                            cluster datasources           |
| tungsten cluster status:                   prints detailed information   |
|                                            about the cluster view this   |
|                                            connector has                 |
| tungsten show [full] processlist:          list all running queries      |
|                                            handled by this connector     |
|                                            instance                      |
| tungsten show variables [like 'string']:   list connector configuration  |
|                                            options in use. The 'string'  |
|                                            may contain '%' wildcards     |
| tungsten flush privileges:                 reload user.map and refresh   |
|                                            user credentials              |
| tungsten mem info:                         display memory information    |
|                                            about current JVM             |
| tungsten gc:                               calls garbage collector       |
| tungsten help:                             display this help message     |
+-------------------------------------------------------------------------------+
9 rows in set (0.00 sec)
```

7.11.7. Connector tungsten mem info Command

```
mysql> tungsten mem info;
+----------------------+-----------------------------------------------------+
| JVM Memory statistics | Value in bytes                                      |
+----------------------+-----------------------------------------------------+
| Peak Thread Count     | 18                                                  |
| Heap Memory           | init = 67108864(65536K) used = 13469328(13153K)     |
|                       | committed = 64946176(63424K) max = 259522560(253440K) |
| Non-heap Memory       | init = 24313856(23744K) used = 14227336(13893K)     |
|                       | committed = 24313856(23744K) max = 224395264(219136K) |
| Thread Count          | 18                                                  |
+----------------------+-----------------------------------------------------+
4 rows in set (0.05 sec)
```

7.11.8. Connector tungsten show [full] processlist Command

```
mysql> tungsten show processlist;
+------------+--------+----------+---------------+----------------+---------+------+-------+------+
| DataSource | Id     | User     | Host          | db             | Command | Time | State | Info |
+------------+--------+----------+---------------+----------------+---------+------+-------+------+
| host1      | 218886 | tungsten | client1:57739 | tungsten_alpha | Sleep   |  316 |       | NULL |
| host1      | 218925 | tungsten | client2:58552 | tungsten_alpha | Sleep   |  281 |       | NULL |
| host1      | 218932 | tungsten | client1:57765 | tungsten_alpha | Sleep   |  274 |       | NULL |
+------------+--------+----------+---------------+----------------+---------+------+-------+------+
3 rows in set (0.05 sec)
```

7.11.9. Connector show slave status Command

The show slave status command generates a version of the standard MySQL command, with the output replaced with values generated by Tungsten. This can be useful for environments where the replica status need to be checked, but with the Tungsten Replicator state, rather than native replication.

```
mysql> show slave status;
*************************** 1. row ***************************
             Slave_IO_State:
                Master_Host: host1
                Master_User:
                Master_Port: 0
              Connect_Retry: 0
...
...

1 row in set (0.01 sec)
```

```
mysql> show replica status;
*************************** 1. row ***************************
           Replica_IO_State:
                Source_Host: host1
                Source_User:
                Source_Port: 0
```

```
              Connect_Retry: 0
...
...

1 row in set (0.01 sec)
```

Note

If the connector is configured in Proxy mode, then any user executing this statement will require the `SELECT` privilege on the tracking schema table `trep_commit_seqno`. The following DDL can be used as an example:

```
GRANT SELECT ON tungsten_<servicename>.trep_commit_seqno TO '<user>'@'<host>';
```

7.11.10. Connector tungsten show variables Command

The connector will intercept the command `tungsten show variables;` and display its current (loaded) configuration. The `variable_Type` column indicates the file where the configuration item is stored.

Important

This command applies only to proxy mode (ie. not bridge mode).

Just as in the MySQL `show variables` command, Tungsten Connector supports filters through the `"like string"` and its wildcard ('%') sign.

```
mysql> tungsten show variables like '%timeout%';
+--------------------+-----------------------------------+----------+
| Variable_Type      | Variable_name                     | Value    |
+--------------------+-----------------------------------+----------+
| connector.properties | bridgeServerToClientForcedCloseTimeout | 50       |
| connector.properties | connection.close.idle.timeout          | 28800000 |
| connector.properties | connection.keepAlive.timeout           | 20160    |
| connector.properties | server.port.binding.timeout            | 60       |
| router.properties    | gatewayConnectTimeoutMs                | 5000     |
| router.properties    | keepAliveTimeout                       | 30000    |
| router.properties    | readCommandRetryTimeoutMs              | 10000    |
| router.properties    | waitForDisconnectTimeout               | 5        |
| router.properties    | waitIfDisabledTimeout                  | 0        |
| router.properties    | waitIfUnavailableTimeout               | 0        |
+--------------------+-----------------------------------+----------+
10 rows in set (0.01 sec)
```

7.12. Advanced Configuration

7.12.1. Working with Proxy Protocol v1

From release 6.1.13, the Connector is able to work with Proxy Protocol v1, available in newer distributions of MySQL from MariaDB and Percona.

By default, when remote clients connect to a Tungsten Cluster via the Connector, the origin IP shown in MySQL maps to the Connector host the client has connected through.

This can have many disadvantages, especially when relying on host level security restricitons within your MySQL database.

The introduction of Proxy Protocol allows the origin IP of the calling client to be passed through instead.

To enable this feature, first you need to ensure you are using a release of MySQL that supports this. At time of documenting this is limited to the following:

- Percona-5.6.25-73.0 and greater
- MariaDB 10.3 and greater

Note

Currently Proxy Protocol is ONLY available in the releases mentioned above. neither Community edition nor Oracle Enterprise edition yet support this.

Next ensure you enable Proxy Protocol within the `my.cnf` by adding the following property: `proxy-protocol-networks=*`

The `*` indicates allow all, however this can be a comma-separated list of (sub)networks or IP addresses instead, to allow finer granularity.

You can now enable the connector to recognise this by enabling the following tpm property: `connector-enable-proxy-protocol=true`

After applying the property and issuing tpm update (or tools/tpm install if a new installation) you should see the correct origin IP's when querying show processlist and tungsten show processlist

If you enable proxy protocol within the connector, but the underlying database does not support this, you will see the following error:

```
shell> tpm connector
mysql: [Warning] Using a password on the command line interface can be insecure.
ERROR 8 (HY000): Could not authorize the user 'app_user'
Attempt to connect to the server failed.
Could not connect: Got packets out of order
Contact your Tungsten administrator.
```

7.12.2. Using Multiple Dataservices

The connector is able to work with multiple dataservices. It may be a combination of Primary/Replica or composite dataservices. The connector will communicate with managers in each dataservice and provide connectivity.

1. Configure the host as a connector for one of the dataservices. This will be the default dataservice for the connector. Any version upgrades for this cluster will also upgrade the connector. See Section 3.6, "Deploying Tungsten Connector Only" if you want the host to be fully independent.

2. Update the `dataservices.properties` file in `/opt/continuent/tungsten/cluster-home/conf`. Add a line for each new Primary/Replica cluster the connector will connect to. Keep this file updated as you add and remove servers from each cluster.

   Important

   Do not list composite dataservices in this file. The connector will automatically discover those from the managers in each cluster.

3. Restart the connector. Any users connected to this connector will be disconnected at this time.

   ```
   shell > connector restart
   ```

4. Update the `user.map` to list new users for each new dataservice. See Section 7.6, "User Authentication" for more details. Specifically, the `user.map` may not include multiple users with the same name but different dataservices. Create unique users in each dataservice before updating `user.map`.

   The read affinity setting supports multiple dataservices with ordering and exclusion.

   You may fine-tune the affinity by specifying an ordered list of dataservice names.

   - `affinity` is defined as the host (single cluster only) or dataservice (composite clusters) to prefer for reads when *`RO_RELAXED`* [240] is in use.

     The affinity feature routes both reads and writes when using a Composite Active/Active topology.

   - This can be handled per user in `user.map`, globally via the tpm option `connector-affinity` [478], or via the connection string.

   - The `affinity` string is an ordered list of dataservice names, separated by commas, where the first dataservice entry will be the one used by default. If the first dataservice in the list is not available, the connector will use the next one listed, and so forth.

   - Dataservices not specified in the list, if any, will be used last and randomly.

   - It is also possible to exclude one or more dataservices by adding a hyphen ("-") in front of the dataservice name.

   - The deployment used in the examples below consists of a composite dataservice `global`, with four member dataservices (i.e. one cluster per site): `east`, `west`, `north` and `south`.

     Just for reference, this is the user definition syntax in the `user.map` file:

     ```
     {User} {Password} {DataServiceName-or-CompositeDataServiceName} [Affinity]
     ```

     For example, to exclude site `south` from servicing read request in `user.map`:

     ```
     sales secret global east,west,north,-south
     ```

     To do the same globally via the tpm command, use the following configuration setting:

     ```
     connector-affinity=east,west,north,-south
     ```

     The above affinity string "`east,west,north,-south`" will try `east`, then `west`, then `north`. If none of the first three are available, a connection request would not succeed since `south` has been excluded by the negation ("-").

In the following `user.map` example, dataservices `north` and `south` would be available as random candidates if the first two [`east` and `west`] were unavailable:

```
sales secret global east,west
```

To do the same globally via the tpm command, use the following configuration setting:

```
connector-affinity=east,west
```

See Section 7.6.1, "`user.map` File Format" for more details.

In 6.0.4 and later, the effect of affinity during cluster changes has been updated to support reconnection when the correct datasource becomes available.

When a site goes offline, connections to this site will be forced closed. Those connections will reconnect, as long as the site stays offline, they will be connected to remote site.

You can now enable an option so that when the site comes back online, the connector will disconnect all these connections that couldn't get to their preferred site so that they will then reconnect to the expected site with the appropriate affinity.

Note that this only applies to bridge mode. In proxy mode, relevancy of connected data source will be re-evaluated before every transaction.

When not enabled, connections will continue to use the server originally configured until they disconnect through normal attribution. This is the default option.

To enable forced reconnection, use the `connector-reset-when-affinity-back=true` [482] option in your configuration.

The connector is now ready to accept users for each of the new dataservices. Keep the `dataservices.properties` and `user.map` files updated to make sure the connector works properly.

7.12.3. Advanced Listeners

Note

This feature is available from v7.0.3.

The idea behind the Connector Advanced Listeners feature is to allow easy definition of one or more listening addresses and ports via a single JSON file. A binding port/address pair can be mapped to a given set of options that will provide a "way" to choose data sources to connect to, through existing connector configuration elements.

The configuration file is written in JSON format in a file located by default in `tungsten-connector/conf/listeners.json`.

Important

If present, this file will be loaded and will override the main configuration, therefore when defining configuration(s) through listeners, other configurations, for example, `connector-readonly-port` will be ignored.

Important

To reload the configuration after changes to the JSON file, the command connector restart must be used, this will cause any existing connections through the connector to be terminated, so care must be taken if implementing this on an existing production system with active clients.

Below is example content defining two Connector ports, one for reads in a local cluster and one for writes in a remote site:

```
[
    {
        "description": "R/W port",
        "listenPort":"3306",
        "dataService": "global",
        "bridgeMode": true,
        "qos": "RW_STRICT"
    },
    {
        "description": "R/O port",
        "listenPort":"3307",
        "dataService": "global",
        "bridgeMode": true,
        "qos": "RO_RELAXED",
        "affinity": "west"
    },
```

```
    {
        "description": "SmartScale port",
        "listenPort":"3308",
        "dataService": "global",
        "smartScale": true,
        "sessionId": "CONNECTION"
    }
]
```

Field Descriptions

- listenAddress (string) : Can be used to restrict connections to a single address. For example, to listen only to local network connections, the local address "192.168.1.10" (or whichever appropriate) will be specified. Default is "0.0.0.0", which is all configured network addresses.
- listenPort (integer) : The Connector port on which to listen for incoming connections. Default is port 3306.
- serverTargetPort (integer) : The MySQL server port to connect to, typically 13306 or any port configured in the target MySQL server. This setting overrides the port from the current cluster definition and should be used with care.
- bridgeMode (boolean) : If set to `true`, the connection to the data source will be made in Bridge mode using the specified QoS to select primary or replica. When `false` or unspecified, connection will be in Proxy mode
- QoS (predefined string) : Quality of Service, either *RW_STRICT* [240] for read/write connections (routes to a primary node), or *RO_RELAXED* [240] for read-only connections (routes to a replica). Defaults to *RW_STRICT* [240]
- smartScale (boolean) : Enables the smart scale feature to read from up-to-date replicas (See Section 7.4.10, "Smartscale Routing" for more information). Setting sessionID will control the desired level of consistency. Defaults to `false`
- sessionId (String) : Relevant only when smartScale is ON, specifies the session ID to use (See Section 7.4.10, "Smartscale Routing" for more information). Values can be `CONNECTION` [245], `USER` [245], `DATABASE` [245], or a free string. Defaults to `DATABASE` [245]
- dataService (String) : Sets the data service to connect to. Typically, in multi-site environments, the global, composite data service will be given. For specific needs, a local data service name can be specified so that connections will never cross the site. Affinity (see below) can also be specified to prefer one site or one host over the other
- affinity (String) : Data service or data source name to select preferably.
- maxAppliedLatency (Integer) : Maximum replica latency in seconds when selecting a node for reading.

7.12.4. Connector Automatic Reconnect

Automatic reconnect enables the Connector to re-establish a connection in the event of a transient failure. Under specific circumstances, the Connector will also retry the query.

1. Connector automatic reconnect is enabled by default in Proxy and Smartscale modes. Use the tpm command option `connector-autoreconnect=false` [479] to disable automatic reconnect.
2. This feature is not available while running in Bridge Mode. Use the tpm command option `connector-bridge-mode=false` [480] to disable Bridge mode.
3. Automatic reconnect enables retries of statements under the following circumstances:
   - not in bridge mode
   - not inside a transaction
   - no temp table has been created
   - no lock acquired and not released
   - the request is a read

The autoreconnect status can be monitored within the *autoReconnect* parameter output by the tungsten show variables command while connected to the Connector. For example:

```
shell> tpm connector
mysql> tungsten show variables like "autoReconnect";
+----------------------+---------------+-------+
| Variable_Type        | Variable_name | Value |
+----------------------+---------------+-------+
| connector.properties | autoReconnect | false |
+----------------------+---------------+-------+
1 row in set (0.00 sec)
```

The above output indicates that the autoreconnect feature is disabled.

> **Note**
>
> The tungsten show command is not available in Bridge mode.

7.12.5. Using the Connector with HAProxy

Tungsten Connector can be used in combination with an HAProxy installation to provide a high-availability connection to the underlying connectors that then provide an intelligent connection to the datasources within the cluster.

There are multiple ways to monitor MySQL health via HAProxy, two which are recommended and one which is not (mysql-check, because this method will flood the Connector logs with connection failure messages):

- `check` - an HAProxy-native test

  Adding `check` to each server line tells HAProxy to perform regular health checks on that backend server. By default, it will try opening a TCP connection and if the TCP handshake succeeds, HAProxy considers that server "up" and if it fails (e.g., no SYN/ACK), the server is marked "down" until it passes again. Without the check option, HAProxy assumes the server is always available and will continue sending traffic to it even if it's unreachable.

  - BRIDGE MODE - # NOT RECOMMENDED

    When the Connector is in Bridge mode, using the native test `check` will generate errors in the connector logs (as well as in the MySQL error log with MySQL versions 8+).

  - PROXY MODE - # RECOMMENDED, with Limitations

    When the Connector is in Proxy mode, it will be considered as up as long as it is running, even if no database backend is available. This has a very low impact on the cluster as the connection will only affect the connector, but it does not guarantee the application will get a proper connection to the database due to the TCP-only nature of the test.

- External Check Script Via systemd

  # RECOMMENDED

  This script is normally launched via systemd, and allows for custom monitoring of the database health. This is the preferred method when full database visibility is needed.

- External Check Script Via xinetd

  # RECOMMENDED, but Obsolete

  This script is normally launched via xinetd, and allows for custom monitoring of the database health. This was the preferred method when full database visibility is needed.

- `mysql-check` - an HAProxy-native test

  - BRIDGE MODE - # NOT RECOMMENDED

    When the Connector is in Bridge mode, using the native test `mysql-check` will generate errors in the connector logs (as well as in the MySQL error log with MySQL versions 8+).

  - PROXY MODE -

    When the Connector is in Proxy mode, using the native test `mysql-check` can be done in two ways, with a MySQL user name specified (supported), and without a MySQL user name (deprecated).

    With a User Name - # RECOMMENDED - If you specify a username, the check consists of sending two MySQL packets, one Client Authentication packet, and one QUIT packet, to correctly close the MySQL session. HAProxy then parses the MySQL Handshake Initialization packet and/or Error packet. The Tungsten Connector will attempt to connect to a data source in the cluster - if none is available, the check will fail and HAProxy will consider this Tungsten Connector as a dead backend. This makes it a useful test which does not produce errors or aborted connects on the server.

    Without a user Name - # NOT RECOMMENDED - If you do not specify a username (this config is deprecated by HAProxy), the check only consists of parsing the MySQL Handshake Initialization packet or Error packet. The Connector will be considered as up as long as it is running, even if no database backend is available. Not recommended due to deprecation.

At the end is a an example `haproxy.cfg` file for a composite cluster, see: Section 7.12.5.5, "Example HAProxy Configuration File"

Exhaustive documentation can be found directly on the HA Proxy documentation page.

Additional configuration examples are located on the HA Proxy config tutorials page.

7.12.5.1. Configuring HAProxy using the native Basic Check

RECOMMENDED

- BRIDGE MODE - # NOT RECOMMENDED

  When the Connector is in Bridge mode, using the native test `check` will generate errors in the Tungsten Connector logs (as well as in the MySQL error log with MySQL versions 8+).

- PROXY MODE - # RECOMMENDED, with Limitations

  When the Connector is in Proxy mode, it will be considered as up as long as it is running, even if no database backend is available. This has a very low impact on the cluster as the connection will only affect the connector, but it does not guarantee the application will get a proper connection to the database due to the TCP-only nature of the test.

Here is a practical example for deploying the HAProxy's native TCP-based `check` option.

```
#---------------------------------------------------------------------
# backend
#---------------------------------------------------------------------
listen connector
    bind *:3306
    mode tcp
    option tcpka   # enables keep-alive both on client and server side
    balance roundrobin
    server conn1 db1:3306 check inter 5s rise 1 fall 1 weight 3 maxconn 5000
    server conn2 db2:3306 check inter 5s rise 1 fall 1 weight 3 maxconn 5000
    server conn3 db3:3306 check inter 5s rise 1 fall 1 weight 3 maxconn 5000
```

This solution does NOT require adding a user to MySQL.

7.12.5.2. Configuring HAProxy with a Check Script Via systemd

RECOMMENDED

A suitable MySQL check script configuration can be added to a basic HAProxy installation using the following settings:

```
#---------------------------------------------------------------------
# frontend
#---------------------------------------------------------------------
frontend connector
  bind *:3306
  default_backend cluster
#---------------------------------------------------------------------
# backend
#---------------------------------------------------------------------
# HTTP health backend that talks to the systemd socket on 9200
backend conn_health
  mode http
  option httpchk GET /health
  http-check expect status 200
  default-server inter 2s fall 3 rise 2

  server conn1 db1:9200 check
  server conn2 db2:9200 check
  server conn3 db3:9200 check

# Actual MySQL traffic backend (TCP) that reuses the health state
backend cluster
  mode tcp
  # Inherit UP/DOWN from the HTTP backend above; don't also set "check" here.
  server conn1 db1:3306 track conn_health/conn1
  server conn2 db2:3306 track conn_health/conn2
  server conn3 db3:3306 track conn_health/conn3

  default-server on-marked-down shutdown-sessions
  timeout check 2s
```

For correct operation with HAProxy, the external check script needs to be installed on all hosts running the Tungsten Connector.

This external script will respond to check calls by HAProxy to ensure the Connector is available and able to reach the database on that node.

This is typically done via the systemd listener configuration which executes the local check script each time the HAProxy daemon asks for status.

The check script will return the status of that Connector to HAProxy in the form of HTTP return codes.

Based on the `default-server port 9200` entry in the configuration above, HAProxy will check via port 9200 on every defined backend host.

> **Note**
>
> The hostname and port numbers in the above example should be modified to match your cluster configuration.

This is the procedure where `systemd socket mode` replaces `xinetd`.

Create and configure the three needed scripts on each host running Tungsten Connector:

- `/etc/systemd/system/connectorchk.socket`

```
[Unit]
Description=Connector health check (socket-activated)

[Socket]
# Listen on all IPv4 addresses on TCP/9200 (add another ListenStream for IPv6 if desired)
ListenStream=0.0.0.0:9200
# Spawn a service instance per connection, with the socket on stdin/stdout
Accept=yes
NoDelay=true

[Install]
WantedBy=sockets.target
```

- `/etc/systemd/system/connectorchk@.service`

```
[Unit]
Description=Connector health check (per-connection instance)
After=network.target

[Service]
Type=simple
User=tungsten
Group=tungsten
ExecStart=/opt/continuent/share/connectorchk.sh

# Make the accepted TCP socket the script's stdin/stdout (inetd-style)
StandardInput=socket
StandardOutput=socket
StandardError=journal

# Optional: set a working directory if your script expects it
# WorkingDirectory=/opt/continuent/share

# --- Optional hardening (uncomment as your script allows) ---
# NoNewPrivileges=true
# PrivateTmp=true
# ProtectSystem=full
# ProtectHome=true
# RestrictAddressFamilies=AF_INET AF_INET6
# RestrictRealtime=true
# LockPersonality=true
# MemoryDenyWriteExecute=true

[Install]
WantedBy=sockets.target
```

- `/opt/continuent/share/connectorchk.sh`

```
#!/bin/sh
#
# This script checks if a mysql server is healthy running on localhost. It will
# return:
# "HTTP/1.x 200 OK\r" (if connector is running smoothly)
# - OR -
# "HTTP/1.x 503 Service Unavailable\r" (else)
#
# The purpose of this script is make haproxy capable of monitoring mysql properly
#

MYSQL_BIN=`which mysql`
MYSQL_HOST=`hostname`
MYSQL_PORT="3306"
MYSQL_USERNAME="haproxy"
MYSQL_PASSWORD="secret"
MYSQL_OPTS="-N -q -A"

#If you create the following file, the proxy will return mysql down
#routing traffic to another host
FORCE_FAIL="/dev/shm/proxyoff"
```

```
OUT=""
return_ok()
{
printf "HTTP/1.1 200 OK\r\n"
printf "Content-Type: text/plain\r\n"
printf "\r\n"
printf "Connector is running.\r\n"
printf "\r\n"
exit 0
}
return_fail()
{
printf "HTTP/1.1 503 Service Unavailable\r\n"
printf "Content-Type: text/plain\r\n"
printf "\r\n"
printf "Connector is *down*.\r\n"
printf "$OUT\r\n\r\n"
exit 1
}
if [ -f "$FORCE_FAIL" ]; then
    OUT="$FORCE_FAIL found"
    return_fail;
fi

OUT=`$MYSQL_BIN $MYSQL_OPTS --host="$MYSQL_HOST" --port="$MYSQL_PORT" --user="$MYSQL_USERNAME" \
--password="$MYSQL_PASSWORD" -e "select @@hostname;" 2>&1`

if [ $? -ne 0 ]; then
    return_fail;
fi

return_ok;
```

You may want to configure the actual check in the `/opt/continuent/share/connectorchk.sh` script, which is is currently doing a connector /status to see if the Tungsten Connector is running.

- Set the permissions for the check scripts:

```
shell> chown tungsten.tungsten /opt/continuent/share/connectorchk.sh
shell> chmod 700 /opt/continuent/share/connectorchk.sh
shell> chmod +x /opt/continuent/share/connectorchk.sh
```

- Configure systemd to add the service:

```
sudo systemctl daemon-reload
sudo systemctl enable --now connectorchk.socket
```

- To verify that the connector check service is running via xinetd, connect to port 9200 using the telnet command:

```
shell> telnet localhost 9200
```

  You should get a response similar to this:

```
HTTP/1.1 200 OK

Content-Type: text/plain


MySQL is running.
```

7.12.5.3. Configuring HAProxy with a Check Script Via xinetd

RECOMMENDED, but Obsolete

A suitable MySQL check script configuration can be added to a basic HAProxy installation using the following settings:

```
#---------------------------------------------------------------------
# backend
#---------------------------------------------------------------------
listen connector
    bind *:3306
    mode tcp
    option tcpka   # enables keep-alive both on client and server side
    balance roundrobin
    default-server port 9200
    server conn1 db1:3306 check inter 5s rise 1 fall 1 weight 3 maxconn 5000
    server conn2 db2:3306 check inter 5s rise 1 fall 1 weight 3 maxconn 5000
    server conn3 db3:3306 check inter 5s rise 1 fall 1 weight 3 maxconn 5000
```

For correct operation with HAProxy, the external check script needs to be installed on all hosts running the Tungsten Connector.

This external script will respond to check calls by HAProxy to ensure the Connector is available and able to reach the database on that node.

This is typically done via the xinetd listener configuration which executes the local check script each time the HAProxy daemon asks for status.

The check script will return the status of that Connector to HAProxy in the form of HTTP return codes.

Based on the `default-server port 9200` entry in the configuration above, HAProxy will check via port 9200 on every defined backend host.

This solution will work for *CONNECTION*-based session IDs when using SmartScale.

> **Note**
>
> The hostname and port numbers in the above example should be modified to match your cluster configuration.

To install the check script:

1. For the check to work, a mysql user with a password must be created within the cluster which the check script can use. The mysql user needs the permissions to be able to run the SQL contained within the check script:

```
WILDCARD IP ACCEPTABLE:
mysql> CREATE USER 'haproxy'@'%' IDENTIFIED WITH mysql_native_password BY 'secret';
mysql> GRANT USAGE ON *.* TO 'haproxy'@'%';

IP REQUIRED, BRIDGE MODE:
mysql> CREATE USER 'haproxy'@'{ip_of_application_host}' IDENTIFIED WITH mysql_native_password BY 'secret';
mysql> GRANT USAGE ON *.* TO 'haproxy'@'{ip_of_application_host}';

IP REQUIRED, PROXY MODE:
mysql> CREATE USER 'haproxy'@'{ip_of_connector_host}' IDENTIFIED WITH mysql_native_password BY 'secret';
mysql> GRANT USAGE ON *.* TO 'haproxy'@'{ip_of_connector_host}';
```

   If you are running SmartScale the user will also need replication client privilege:

```
WILDCARD IP ACCEPTABLE:
mysql> GRANT USAGE, REPLICATION CLIENT ON *.* TO 'haproxy'@'%';

IP REQUIRED, BRIDGE MODE:
mysql> GRANT USAGE, REPLICATION CLIENT ON *.* TO 'haproxy'@'{ip_of_application_host}';

IP REQUIRED, PROXY MODE:
mysql> GRANT USAGE, REPLICATION CLIENT ON *.* TO 'haproxy'@'{ip_of_connector_host}';
```

2. Add the new user on each connector host by adding the following line to `user.map`:

```
haproxy secret cluster_name_here
```

   In the above example, replace `cluster_name_here` with the name of the actual cluster service.

3. Create and configure a check script on each host running Tungsten Connector. For example, create the file `/opt/continuent/share/connectorchk.sh`:

```
#!/bin/sh
#
# This script checks if a mysql server is healthy running on localhost. It will
# return:
# "HTTP/1.x 200 OK\r" (if mysql is running smoothly)
# - OR -
# "HTTP/1.x 503 Service Unavailable\r" (else)
#
# The purpose of this script is make haproxy capable of monitoring mysql properly
#

MYSQL_HOST=`hostname`
MYSQL_PORT="3306"           #Connector Port
MYSQL_USERNAME="haproxy"
MYSQL_PASSWORD="secret"
MYSQL_OPTS="-N -q -A test"

#If you create the following file, the proxy will return mysql down
#routing traffic to another host
FORCE_FAIL="/dev/shm/proxyoff"
OUT=""
return_ok()
{
echo -e "HTTP/1.1 200 OK\r\n"
echo -e "Content-Type: text/plain\r\n"
echo -e "\r\n"
echo -e "MySQL is running.\r\n"
echo -e "\r\n"
```

```
exit 0
}
return_fail()
{
echo -e "HTTP/1.1 503 Service Unavailable\r\n"
echo -e "Content-Type: text/plain\r\n"
echo -e "\r\n"
echo -e "MySQL is *down*.\r\n"
echo -e "$OUT\r\n\r\n"
exit 1
}
if [ -f "$FORCE_FAIL" ]; then
    OUT="$FORCE_FAIL found"
    return_fail;
fi
OUT=`mysql $MYSQL_OPTS --host=$MYSQL_HOST --port=$MYSQL_PORT --user=$MYSQL_USERNAME \
--password=$MYSQL_PASSWORD -e "select @@hostname;" 2>&1`
if [ $? -ne 0 ]; then
    return_fail;
fi

return_ok;
```

Set the permissions for the check script:

```
shell> chown tungsten.tungsten /opt/continuent/share/connectorchk.sh
shell> chmod 701 /opt/continuent/share/connectorchk.sh
```

4. Install xinetd and add the xinetd service. On RedHat/CentOS:

```
shell> sudo yum -y install  xinetd telnet
```

On Debian/Ubuntu:

```
shell> sudo apt-get install xinetd telnet
```

5. Add an entry for the connector check script to `/etc/services`:

```
shell> echo "connectorchk         9200/tcp" | sudo tee -a /etc/services
```

6. Add a configuration to xinetd by creating the file `/etc/xinetd.d/connectorchk` as the root user with the following content:

```
# default: on
# description:connectorchk
service connectorchk
{
flags           = REUSE
    socket_type     = stream
    port            = 9200
    wait            = no
    user            = tungsten
    server          = /opt/continuent/share/connectorchk.sh
    log_on_failure  += USERID
    disable         = no
#       only_from       = 0.0.0.0/0
# recommended to put the IPs that need
# to connect exclusively (security purposes)
    per_source      = UNLIMITED
}
```

7. Now restart xinetd:

```
shell> sudo service xinetd restart
```

8. To verify that the connector check service is running via xinetd, connect to port 9200 using the telnet command:

```
shell> telnet localhost 9200
```

You should get a response similar to this:

```
HTTP/1.1 200 OK

Content-Type: text/plain


MySQL is running.
```

7.12.5.4. Configuring HAProxy using the native MySQL Check

- BRIDGE MODE - # NOT RECOMMENDED

When the Connector is in Bridge mode, using the native test `mysql-check` will generate errors in the connector logs (as well as in the MySQL error log with MySQL versions 8+).

- PROXY MODE -

  When the Connector is in Proxy mode, using the native test `mysql-check` can be done in two ways, with a MySQL user name specified (supported), and without a MySQL user name (deprecated).

  With a User Name - # RECOMMENDED - If you specify a username, the check consists of sending two MySQL packets, one Client Authentication packet, and one QUIT packet, to correctly close the MySQL session. HAProxy then parses the MySQL Handshake Initialization packet and/or Error packet. The Tungsten Connector will attempt to connect to a data source in the cluster - if none is available, the check will fail and HAProxy will consider this Tungsten Connector as a dead backend. This makes it a useful test which does not produce errors or aborted connects on the server.

  Without a user Name - # NOT RECOMMENDED - If you do not specify a username (this config is deprecated by HAProxy), the check only consists of parsing the MySQL Handshake Initialization packet or Error packet. The Connector will be considered as up as long as it is running, even if no database backend is available. Not recommended due to deprecation.

  Warning

  The mysql-check method (with or without a user specified) does NOT check for database presence nor database consistency. To do this, we must use an external check script (via systemd, see Section 7.12.5.2, “Configuring HAProxy with a Check Script Via systemd”)

Here is a practical example for deploying the HAProxy's native `mysql-check` option.

```
#---------------------------------------------------------------------
# backend
#---------------------------------------------------------------------
listen connector
    bind *:3306
    mode tcp
    option tcpka   # enables keep-alive both on client and server side
    balance roundrobin
    option mysql-check user haproxy post-41
    server conn1 db1:3306 check inter 5s rise 1 fall 1 weight 3 maxconn 5000
    server conn2 db2:3306 check inter 5s rise 1 fall 1 weight 3 maxconn 5000
    server conn3 db3:3306 check inter 5s rise 1 fall 1 weight 3 maxconn 5000
```

Important

This solution requires adding a user to MySQL with an empty password.

An example of using mysql-check without a username is not provided here because this functionality is deprecated by HAProxy and not recommended.

The first example uses a `%` to specify a source IP wildcard during user creation:

```
mysql> CREATE USER 'haproxy'@'%' IDENTIFIED WITH mysql_native_password;
```

For deployments that require MySQL to specify an IP address, which address you choose depends on the Connector mode you are using - Bridge or Proxy.

- Bridge Mode

  In Bridge mode, the IP address seen by the database server will be that of the calling application server (ip_of_application_host).

  The below example shows the user creation for Bridge mode deployments that require MySQL to specify an IP address:

  ```
  mysql> CREATE USER 'haproxy'@'{ip_of_application_host}' IDENTIFIED WITH mysql_native_password;
  ```

- Proxy Mode

  In Proxy mode, the IP address seen by the database server will be that of the Tungsten Connector host (ip_of_connector_host).

  This example shows the user creation for Proxy mode deployments that require MySQL to specify an IP address:

  ```
  mysql> CREATE USER 'haproxy'@'{ip_of_connector_host}' IDENTIFIED WITH mysql_native_password;
  ```

  In proxy Mode, the `user.map` file must also be updated as follows:

  ```
  haproxy - cluster_name_here
  ```

In the above example, replace `cluster_name_here` with the name of the actual cluster service. Note that a hyphen [-] is used to indicate a blank password.

> **Note**
>
> MySQL 8 will let you create a user with an empty password provided you don't have a password-validation plugin rejecting it. By default, MySQL 8 does not install or enforce the validate_password plugin, so an empty-string password is allowed. If you see an error when doing this, it usually means validate_password (or another authentication plugin) has been enabled and is enforcing a minimum length or complexity—disabling or tuning that plugin will then allow you to use '' as the password.

7.12.5.5. Example HAProxy Configuration File

Here is an example `haproxy.cfg` file:

```
global
  log         127.0.0.1 local2

  chroot      /var/lib/haproxy
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  user        haproxy
  group       haproxy
  daemon

  # turn on stats unix socket
  # echo "show stat" | nc -U /var/lib/haproxy/stats
  stats socket /var/lib/haproxy/stats

defaults
  mode                    tcp
  option                  tcpka
  option                  dontlognull
  option                  redispatch
  retries                 3
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout check           10s

frontend connector
    bind *:3306
    default_backend cluster

backend cluster
    balance roundrobin
    option tcplog
    #default-server port 9200
    server  conn1 db1:3306 check inter 5s rise 1 fall 1 weight 6 maxconn 5000
    server  conn2 db2:3306 check inter 5s rise 1 fall 1 weight 6 maxconn 5000
    server  conn3 db3:3306 check inter 5s rise 1 fall 1 weight 6 maxconn 5000
```

This example has the Connectors installed on the three database nodes.

Uncomment the `default-server port 9200` line to use the external check script.

7.12.5.6. Monitoring HAProxy Status

To check the status of the HAProxy service the `stats socket` directive must be defined in the `haproxy.cfg` file:

```
global
  # turn on stats unix socket
  stats socket /var/lib/haproxy/stats
  ...
```

> **Note**
>
> HAProxy must be restarted if you change the `haproxy.cfg` file.

To check the status of the HAProxy service using the nc command:

```
shell> echo "show stat" | sudo nc -U /var/lib/haproxy/stats
```

You should get a response similar to this example which has a frontend and a backend defined with three cluster nodes:

```
# pxname,svname,qcur,qmax,scur,smax,slim,stot,bin,bout,dreq,dresp,ereq,econ,eresp,wretr,wredis,status,weight,act,bck,chkfail,chkdown,lastchg,downtime,qlim
```

```
connector,FRONTEND,,,0,1,3000,2,1261,4062,0,0,0,,,,,OPEN,,,,,,,,,1,2,0,,,,0,0,0,1,,,,,,,,,,,0,0,0,,,0,0,0,0,,,,,,,
connector,conn1,0,0,0,1,5000,1,605,1915,,0,,0,0,0,0,0,UP,3,1,0,0,0,327,0,,1,3,1,,1,,2,0,,1,L4OK,,0,,,,,,0,,,1,0,,,,306,,,0,0,0,4,
connector,conn2,0,0,0,1,5000,1,656,2147,,0,,0,0,0,0,0,UP,3,1,0,0,0,327,0,,1,3,2,,1,,2,0,,1,L4OK,,0,,,,,,0,,,0,1,,,,302,,,0,0,0,138,
connector,conn3,0,0,0,0,5000,0,0,0,,0,,0,0,0,0,0,UP,3,1,0,0,0,327,0,,1,3,3,,0,,2,0,,0,L4OK,,0,,,,,,0,,,0,0,,,,-1,,,0,0,0,0,
connector,BACKEND,0,0,0,1,300,2,1261,4062,0,0,,0,0,0,0,0,UP,18,6,0,,0,327,0,,1,3,0,,2,,1,0,,1,,,,,,,,,,,,,1,1,0,0,0,0,302,,,0,0,0,141,
```

Note

You must have the nc command installed, and the command needs to access the socket as root.

7.12.6. Using Fall-Back Bridge Mode

This feature will allow the Tungsten Connector to fall back to bridge mode if a user cannot be successfully authenticated through `user.map`.

The connector is able to employ a special fall-back bridge mode which allows for a hybrid configuration of both Proxy and Bridge modes. By default, the bridge mode fallback feature is disabled.

When `fallBackBridgeMode` is set to either *`RW_STRICT`* [240] or *`RO_RELAXED`* [240], the Connector will first check the `user.map` file for an entry that matches the user name passed in the connection request. If a match is found in the `user.map`, the Connector will act in Proxy mode so the conversation with the client will be handled locally, and a new connection will be opened from the connector to the database server based on the normal Proxy mode routing rules. If the user name is not found in the `user.map`, then the connector will act in Bridge mode, and the connection will be forwarded directly to the specified database server, either to the Primary [*`RW_STRICT`* [240]] or to the Replica [*`RO_RELAXED`* [240]] for handling with no intercept, just a TCP-layer packet routing. There will be no query interpretation or analysis, and no auto-reconnect, just failover handling.

Support for Fall-Back Bridge Mode and MySQL 8 was introduced in v7.0.3.

For more information, see Section 7.5, “Using Bridge Mode”, and Section 7.6, “User Authentication”.

To enable Fall-Back Bridge Mode to use the Primary, include the following entries to the `/etc/tungsten/tungsten.ini` file prior to installation, or issuing tpm update after installation:

```
property=fallBackBridgeMode=RW_STRICT
connector-bridge-mode=false
```

To enable Fall-Back Bridge Mode to use a replica (if available);, include the following entries to the `/etc/tungsten/tungsten.ini` file prior to installation, or issuing tpm update after installation:

```
property=fallBackBridgeMode=RO_RELAXED
connector-bridge-mode=false
```

Important

Updating these values require a connector restart (via tpm update), if enabled after installation, for the changes to be recognized.

Important

To be consistent, Bridge mode should be disabled when `fallBackBridgeMode` is enabled. The `connector-bridge-mode` [480] option must be set to `false`. A consistency check is performed when starting the connector.

7.12.6.1. Using Fall-Back SSL To Bridge Mode

SSL connections are by design unreadable until the handshake has been exchanged. Because of this, the MySQL user name in the request is not visible to the Connector immediately, and therefore the Connector is unable to check against the `user.map` for `fallBackBridgeMode`.

Due to this situation, another feature was created to address SSL connections while the `fallBackBridgeMode` is enabled called `fallBackSSLToBridge`. When `fallBackSSLToBridge` is set to `true` (default), then all SSL connections will use Bridge mode, while non-SSL connections will use the `fallBackBridgeMode` setting (i.e. *`RW_STRICT`* [240] which routes traffic to the Primary or *`RO_RELAXED`* [240] which routes to the Replicas). When `fallBackSSLToBridge` is set to `false`, then SSL connections will run in non-Bridge mode - if the specified user doesn't exist in the `user.map`, an error will be raised.

Important

The `fallBackSSLToBridge` setting is ONLY available when `fallBackBridgeMode` is enabled, and is ignored when `fallBackBridgeMode` is set to `false`.

Since `fallBackSSLToBridge` is enabled by default when `fallBackBridgeMode` is enabled, you may turn it off with the following property:

```
property=fallBackSSLToBridge=false
```

Warning

Updating these values require a connector restart (via tpm update), if enabled after installation, for the changes to be recognized.

7.12.7. Using the Max Connections Feature

This feature will allow the connector behavior to be changed based on the connection count. The connector is able to mimic MySQL's `max_connections`. Depending on your needs, the connector can be configured to pile up or reject connections above this number. This is served by the following two tpm flags:

`connector-max-connections` [481] - defines the maximum number of connections the connector should allow at any time.

When the `connector-max-connections` [481] is set to a non-zero numeric value, the connector denies access to the client in one of two ways: queue (default) or reject.

`connector-drop-after-max-connections` [480] - defines how the connector should handle new connection requests - queue (default) or reject.

Enabling this option causes the connector to drop new connection requests when `connector-max-connections` [481] is reached by immediately sending a "Too Many Connections" error to the client, just like MySQL would.

Important

When a client connection request arrives at the connector, an object is created to track that client connection which uses a certain amount of memory.

The connector then checks the value of `connector-max-connections` [481] against the current connection count.

If the connection limit has been reached, the connector decides how to behave by checking the value of `connector-drop-after-max-connections` [480].

If `connector-drop-after-max-connections` [480] is false (the default), the connector will queue the connection request, but send nothing back to the client at all. This connection check will repeat after a delay. Once the connection count falls below `connector-max-connections` [481] an attempt to connect to a server is made. In this mode, connections will continue to pile up in memory as new requests are queued, resulting in an Out of Memory error. For this reason, the `connector-drop-after-max-connections` [480] is available to prevent connection queueing when the maximum number of connections has been reached.

If `connector-drop-after-max-connections` [480] is enabled, the connector will return a "Too Many Connections" error to the client and remove the client connection object, freeing memory.

To enable `connector-max-connections` [481] include the following setting in your `/etc/tungsten/tungsten.ini`:

```
connector-max-connections=2500
connector-drop-after-max-connections=true
```

To enable `connector-drop-after-max-connections` [480] you must also set a non-zero value for `connector-max-connections` [481]:

Important

Updating these values require a connector restart (via tpm update), if enabled after installation, for the changes to be recognized.

To select a real-world value for `connector-max-connections` [481], set the value to a value slightly lower than the MySQL value of `max_connections` to prevent the server from ever hitting maximum. You may use the following formula for a more complex calculation:

```
connector-max-connections = ( MySQL Primary max_connections / number of connectors ) * 0.95
```

Important

When `connector-drop-after-max-connections` [480] is enabled, be sure that your load balancers are configured to identify that max connections have been reached and to switch to another connector when that happens.

7.12.8. Adjusting the Client Disconnect Delay During Manual Switch

This feature controls how the connector handles existing connections when a manual switch is invoked.

When a graceful switch is invoked via cctrl, by default the Connector will wait for five seconds to allow in-flight activities to complete before forcibly disconnecting all active connections from the application side, no matter what type of query was in use.

If connections still exist after the timeout interval, they are forced closed, and the application will get back an error.

Important

This setting ONLY applies to a manual switch. During a failover, there is no wait and all connections are force-closed immediately.

This timeout is adjusted via the tpm option `connector-disconnect-timeout` [480].

For example, to change the delay to 10 seconds, include the following setting in your `/etc/tungsten/tungsten.ini` fie::

```
connector-disconnect-timeout=10
```

Warning

If you increase this value, you delay the manual switch! ONLY change this if you accept the fact that the manual switch process will last at least as long as this setting in seconds.

Do not set this value to zero (0) or there will be no attempt to disconnect at all. If you wish to disable the wait entirely, set `--property=waitForDisconnect=false` [443] in your configuration on the connectors and run tpm update.

Important

Updating these values require a connector restart (via tpm update), if enabled after installation, for the changes to be recognized.

This value is reflected in the `waitForDisconnectTimeout` setting located in `cluster-home/conf/router.properties`.

7.12.9. Adjusting the Bridge Mode Forced Client Disconnect Timeout

This feature controls how long the Connector in Bridge mode waits before forcibly disconnecting the server side of the session after a client session ends.

Default: 50ms

When a client application opens a socket and connects to the connector, a second socket/connection to the server is created. The Connector in bridge mode then simply transfers data between these two sockets.

When a client application brutally closes a connection without following the proper disconnection protocol, the server will not know about that disconnect until the connector closes the Connector<>server socket. If the connector closes the Connector<>server connection too soon after a client disconnect, there is a chance that the proper disconnection messages will be missed, if sent late. If the connector does not close this connector<>server connection, it would stay open indefinitely, using memory and resources that would otherwise be reclaimed.

The default of 50ms is very conservative and will fit most environments where client applications disconnect properly. When the volume of connections opened and never closed exceeds a certain level, the timeout must be tuned (lowered) to close idle connections faster, or the available resources will get used up.

Many times this situation is caused by health checks, especially from monitoring scripts and load balancers checking port liveness. Many of these check do not gracefully close the connection, triggering the need for tuning the Connector.

If connections still exist after the timeout interval, they are forced closed, and a warning will be printed in the connector logs.

```
C>S ended. S>Cstreaming did not finish within bridgeServerToClientForcedCloseTimeout=500(ms). Will be closed anyway !
```

Important

This setting ONLY applies to Bridge mode.

This timeout is adjusted via the tpm property `property=bridgeServerToClientForcedCloseTimeout` [443] in milliseconds.

For example, to change the delay to 100 milliseconds, the following property should be set in your `/etc/tungsten/tungsten.ini` file:

```
property=bridgeServerToClientForcedCloseTimeout=100
```

Warning

If you decrease this value, you run the risk of disconnecting valid but slow sessions.

Important

Updating these values require a connector restart (via tpm update), if enabled after installation, for the changes to be recognized.

7.12.10. Adjusting the Connnector Response to Resource Losses

This section describes how to control the Connector responses in the event of the loss of a required Datasource or all Managers.

7.12.10.1. Adjusting the Connnector Response to Datasource Loss

Summary: Whenever no Primary datasource is found, the Connector will reject connection requests.

This feature controls how long the Connector waits for the given type of DataSource to come ONLINE before forcibly disconnecting the client application.

By default, wait indefinitely for a resource to become available.

There are two parameters involved in this decision-making. They are:

- `waitIfUnavailable` (default: `true`)

  If `waitIfUnavailable` is `true`, then the Connector will wait for up to the time period specified by `waitIfUnavailableTimeout` to make a connection for a given QOS. If the timeout expires, the Connector will disconnect the client application (reject connection attempts and close ongoing connections).

  If `waitIfUnavailable` is `false`, the Connector will immediately disconnect the client with an error if a connection for a given QOS cannot be made immediately.

- `waitIfUnavailableTimeout` (default: 0, wait indefinitely)

  If `waitIfUnavailable` is `true`, the Connector will wait for up to `waitIfUnavailableTimeout` number of seconds before disconnecting the client. If `waitIfUnavailable` is `false`, this parameter is ignored. If this parameter is set to zero (0) seconds, the Connector will wait indefinitely (client connection requests will hang forever).

`waitIfUnavailable` is specific to data source availability and will be considered if everything else is online: connector and data service.

This will typically be used during a switch or failover while the primary changes: the client application will request a primary (*`RW_STRICT`* [240] QoS), which at some point is not available since both new and old primaries are offline. With `waitIfUnavailable=true`, the connector will wait for the new one to come online (Upto `waitIfUnavailableTimeout` seconds), allowing seamless failover. If set to false, there will be a period of time during switch/failover where client applications will get errors trying to connect AND reconnect/retry failing requests.

For example, to immediately reject connections upon Datasource loss, set the following property in your `/etc/tungsten/tungsten.ini` file:

```
property=waitIfUnavailable=false
```

Warning

PLEASE NOTE: this will make switch and failover much less transparent to the application since the connections will error until the new Primary is elected and back online.

Important

Updating these values require a connector restart (via tpm update), if enabled after installation, for the changes to be recognized.

Important

These entries will NOT work if placed into `[defaults]`, each service must be handled individually.

7.12.10.2. Adjusting the Connnector Response to Manager Loss

Summary: Whenever the Connector loses sight of the managers for a given data service, it will either suspend or reject new connection requests.

`waitIfDisabled` applies to both:

1. Whole offline data service: client application tries to connect to a composite or physical data service that is offline, e.g. during a full site outage where the client application requests access to a local primary without allowing redirection to remote site.
2. connector offline or onhold: typically when the connector looses connectivity to all managers in the cluster, it will first go `ON HOLD`, then `OFFLINE`. In both cases, waitIfDisabled defines what to do in such case: throw an error to the client application or wait until network is back and access to a manager is possible. For example, when the connector is isolated from the cluster, setting `waitIfDisabled=true` will make new connection requests "hang" until either the connector gets back network access to a manager OR `waitIfDisabledTimeout` is reached

By default, suspend requests indefinitely until Manager communications are re-established.

This feature controls how long the Connector waits during a manager loss event to either suspend or reject the client connection.

Here is the decision chain and associated settings for what happens when the connector loses sight of the managers:

1. Delay for the value of `delayBeforeOnHoldIfNoManager` seconds which is 0/no delay by default.
2. Change state to `ON HOLD` and begin the countdown timer starting from the `delayBeforeOfflineIfNoManager` value.

   In the `ON HOLD` state, the connector will hang all new connections and allow existing connections to continue.
3. When the `delayBeforeOfflineIfNoManager` timer expires (30 seconds by default), change state to `OFFLINE`.

   Once `OFFLINE`, the Connector with break existing connections because there is no authoritative Manager node from the Connector's perspective. Without a Manager link, any change to the cluster configuration will remain invisible to the Connector, potentially leading to writes on a Replica node.

   By default, all new connection requests will hang in the `OFFLINE` state. If `waitIfDisabled` is set to `false`, then the Connector will instead reject all new connections.

There are multiple parameters involved in this decision-making. They are:

- `delayBeforeOnHoldIfNoManager` (in seconds, default: `0`, i.e. no delay)

  When the connector loses sight of the managers, delay before going `ON HOLD` for the value of `delayBeforeOnHoldIfNoManager` seconds, which is 0/no delay by default.
- `delayBeforeOfflineIfNoManager` (in seconds, default: 30)

  Once `ON HOLD`, delay before going `OFFLINE` for the value of `delayBeforeOfflineIfNoManager` seconds, 30 by default.
- `waitIfDisabled` (default: `true`)

  If the Dataservice is `OFFLINE` because it is unable to communicate with any Manager, the `waitIfDisabled` parameter determines whether to suspend connection requests or to reject them. If `waitIfDisabled` is `true` (the default), then the Connector will wait indefinitely for manager communications to be re-established. If `waitIfDisabled` is set to `false`, the Connector will return an error immediately.

  To check for data service state, use the tungsten-connector/bin/connector cluster-status command. For example:

```
shell> connector cluster-status
Executing Tungsten Connector Service --cluster-status ...
+--------------+-------------------+-------------+--------------+--------+--------+-----------------------------------+-----------------+----------
| Data service | Data service state | Data source | Is composite | Role   | State  | High water                        | Last shun reason | Applied l
+--------------+-------------------+-------------+--------------+--------+--------+-----------------------------------+-----------------+----------
| europe       | OFFLINE           | c1          | false        | master | ONLINE | 0(c1-bin.000002:0000000000000510;-1) | MANUALLY-SHUNNED | 0.0
| europe       | OFFLINE           | c2          | false        | slave  | ONLINE | 0(c1-bin.000002:0000000000000510;-1) |                  | 1.0
| europe       | OFFLINE           | c3          | false        | slave  | ONLINE | 0(c1-bin.000002:0000000000000510;-1) |                  | 2.0
+--------------+-------------------+-------------+--------------+--------+--------+-----------------------------------+-----------------+----------
```

For more information, see Connector On-Hold State [258].

For example, to decrease the `ON HOLD` time to 15 seconds, the following property should be added to your `/etc/tungsten/tungsten.ini` file:

```
property=delayBeforeOfflineIfNoManager=15
```

For example, to immediately reject connections upon Manager loss, the following property should be added to your `/etc/tungsten/tungsten.ini` file:

```
property=waitIfDisabled=false
```

Warning

PLEASE NOTE: this will make switch and failover much less transparent to the application since the connections will error until communications with at least one manager has been established and the Connector is back online.

Important

Updating these values require a connector restart (via tpm update), if enabled after installation, for the changes to be recognized.

Important

These entries will NOT work if placed into `[defaults]`, each service must be handled individually.

7.12.11. Connector Logging Configuration

The connector is able to modify what is logged using different configuration options.

Informational messages like those below indicate that a client-side session did not properly close connection to the connector using a regular mysql close() call.

```
WARN [MySQLBridge] - [10.3.1.240:35686] C>S ended. S>C streaming did not finish within
                                        bridgeServerToClientForcedCloseTimeout=500 (ms).
                                        Will be closed anyway !

INFO [MySQLBridge] - [10.3.1.240:35686] Server>Client thread didn't finish cleanly - »
                                        return code was: 1
```

Load balancers (especially physical ones) tend to do this when testing the connector, so these messages are expected when behind a load-balancer. Since the same error could come from buggy application code, the default is to log these circumstances.

If you know where the disconnects are coming from, you can safely disable this feature by setting the tpm option `connector-disable-connection-warnings` [480] to `true`

> **Important**
>
> Updating these values require a connector restart (via tpm update), if enabled after installation, for the changes to be recognized.

7.12.11.1. Connector Logging to Syslog

The Connector is able to send log information to Syslog.

You can control the full connector log by editing the `tungsten-connector/conf/wrapper.conf` file.

Depending on what you want to do, you can:

- Send full connector log to the syslog:

```
wrapper.syslog.loglevel=NONE =>
wrapper.syslog.loglevel=INFO
```

- Stop sending log to `connector.log`:

```
wrapper.logfile.loglevel=INFO =>
wrapper.logfile.loglevel=NONE
```

- Change the log file path:

```
wrapper.logfile=../../tungsten-connector/log/connector.log
```

> **Note**
>
> For logging purposes, our software uses both the Tanuki wrapper and Apache's Log4J tool.
>
> Valid log levels for the Tanuki wrapper include:
>
> - NONE for no output
> - FATAL to only show fatal error messages
> - ERROR to show all error messages
> - WARN to show all warning messages
> - STATUS to show all state changes
> - INFO shows all JVM output and informative messages
> - DEBUG shows detailed debug information
>
> Valid log levels for Apache's log4j include: TRACE, DEBUG, INFO, WARN, ERROR and FATAL
>
> Links to the full documentation for each are below:
>
> - Tanuki wrapper: https://wrapper.tanukisoftware.com/doc/english/prop-syslog-loglevel.html

- Apache log4j: https://logging.apache.org/log4j/2.x/manual/configuration.html

7.12.12. Connector Audit Logging

Audit Logging allows for logging data transferred between client application and MySQL servers to a file, database or socket.

Configuration of audit logging is done directly in the connector's `tungsten-connector/conf/log4j.properties` configuration file by editing the parameters explained below. Changes to this configuration file do NOT need a full restart, the command connector reconfigure [334] will reload the log4j configuration and apply changes immediately.

5 categories of information can be printed:

- `ClientRequests` : client application sends a MySQL request (select, insert, update, ...) to the server. Compatible with both bridge and proxy mode, full packet text will have to be printed in order to see the actual request
- `ServerResponses` : MySQL server returns a result in response to a client request. Compatible with both bridge and proxy mode, full packet text will have to be printed in order to see the actual responses
- `AppConnections` : client application connects or disconnects to/from the Connector. Compatible with both bridge and proxy mode, will print one log entry per connection and one per disconnection
- `Commands` : proxy mode only. Will print each "mysql command" that is sent by the application to the connector. Commands include not only regular `COM_QUERY` and `COM_STMT_xxx` used for executing requests and prepared statements, but also more specific commands such as `COM_PING`, `COM_SLEEP`, or `COM_INIT_DB` and `COM_CHANGE_USER` to switch user or schema. Full list of commands is available here
- `Perf` : proxy mode only. When set to info, this logger will print the round trip time to execute individual MySQL commands (see above). A typical use of this logger is to track individual requests execution time.

Each of these categories logs may be enabled of disabled individually with:

```
logger.{category}.level=[OFF|INFO]
```

For example, all incoming client requests can be printed with the following line:

```
logger.ClientRequests.level=INFO
```

Individual log entries (individual line printed) format can be adjusted in this same `conf/log4j.properties` with the line:

```
appender.audit.layout.pattern=%d - [%X{client-ip}:%X{client-port}]<->%X{datasource} %X{user} %X{schema} %m: %X{packet-text}%n
```

Where %X entries are:

- `client-ip` : client application IP address as seen by the connector
- `client-port` : client application outgoing TCP port. Convenient to identify a given client application since a connected IP/port pair is unique at any given time
- `user` : (not available in bridge mode) user name authenticated to the connector. Since bridge mode doesn't inspect packet internals, it will not be able to display this information
- `schema` : (not available in bridge mode) schema on which the client application is connected. Will be blank if no schema has been selected. Since bridge mode doesn't inspect packet internals, it will not be able to display this information
- `datasource` : cluster data source used to execute the request, composed of `<hostname>@<dataservice>(role:STATE)`
- `packet-header` : MySQL packet header as `Packet #<packet number> size=<full packet length (inc. header)> pos=<internal pointer position>`
- `packet-text` : internal packet bytes printable as characters. This will display a readable form of the requests. Note that the header bytes are not printed here
- `packet-binary` : internal packet bytes as hexadecimal values. Will display the full packet, yet non-human-readable form. Note that the header bytes are also printed

By default, audit logging will be output in `tungsten-connector/log/connector-audit.log` file.

Log4j 2, the logging system used by Tungsten, allows for many other options and ways to log data, including Socket and Database logging. Deeper insight and exhaustive documentation will be found directly on the log4j 2 documentation page

7.12.13. Connector SSL Advertisement Configuration

When SSL is enabled, the Connector automatically advertises the ports and itself as SSL capable. With some clients, this triggers them to use SSL even if SSL has not been configured. This causes the connections to fail and not operate correctly.

You can safely disable SSL advertisement in the Connector by setting the tpm option `connector-ssl-capable` [484] to `false`

> **Important**
>
> Updating these values require a connector restart (via tpm update), if enabled after installation, for the changes to be recognized.

7.12.14. Connector IP Address Configuration

When the Connector starts up, it binds to all local IP addresses on the selected port using the wilcard syntax, for example: `0.0.0.0:3306`.

You can force the Connector to bind to a specific IP address by setting the tpm option `property=server.listen.address={IP_ADDRESS}` [443]

If you wished to bind to the localhost IP address only, perhaps for security reasons, here is an example:

```
property=server.listen.address=127.0.0.1
```

> **Important**
>
> Updating these values require a connector restart (via tpm update), if enabled after installation, for the changes to be recognized.

7.12.15. Deploying a Connector through Docker

> **Note**
>
> Tungsten Connector is available to deploy as a Docker Image from v7.0.3 onwards.

Providing you have a license to use Tungsten Clustering, you will be able to download the docker image from the download portal of the Continuent Website.

The image is available as a gzipped tar ball, and the following steps outline how to import and use the image

Step 1 : Download the files

Once downloaded, you will need to uncompress the file and import it into docker. The following syntax can be used as an example:

```
shell> tar zxvf tungsten-connector-docker-7.0.3-141.tar.gz
```

This will result in a number of files/folders available in the tungsten-connector-docker-7.0.3-141 directry, within the path where you extracted the tar package, for example:

```
shell> ls -lh
total 1048848
-rw-r--r--@ 1 cssi  staff   1.3K May 24 12:16 README.txt
drwxr-xr-x@ 5 cssi  staff   160B May 24 11:45 conf/
-rw-r--r--@ 1 cssi  staff   887B May 24 12:07 docker-compose.yml
drwxr-xr-x@ 3 cssi  staff    96B May 24 11:54 security/
-rw-------@ 1 cssi  staff   499M May 24 12:38 tungsten-connector-docker-image-7.1.3-3.tar.gz
```

You will find a README.txt within the root folder, and also within the security folder, these files contain the same examples and steps as outlined on this page.

Step 2 : Configure the Connector

Before launching the connector, you will need to apply a number of configuration changes so that the Connector can communicate with your cluster and act in the desired mode based on your needs.

Modify the docker-compose.yml file

The file supplied looks like the following:

```
version: "3.9"
services:
  connector:
    ports:
      # connector listen port
      - 3306:3306
      # connector REST API listen port
      - 8096:8096
    environment:
      CONNECTOR_DATASERVICES: "alpha:db1,db2,db3"
      CONNECTOR_NAME: "connector"
```

```
    extra_hosts:
      - "db1:192.168.30.81"
      - "db2:192.168.30.82"
      - "db3:192.168.30.83"
    volumes:
      - ./conf/connector.properties:/opt/continuent/tungsten-connector/conf/connector.properties
      # if bridgeMode=OFF the next line is needed, please uncomment it
      #- ./conf/user.map:/opt/continuent/tungsten-connector/conf/user.map
      # if SSL is configured the following two lines are needed, please uncomment them
      #- ./conf/security.properties:/opt/continuent/cluster-home/conf/security.properties
      #- ./security:/opt/continuent/share
    image: "tungsten-connector:latest"
    restart: always
```

In this file you need to define your cluster nodes and cluster service, along with enabling various connector mode and security options. The following changes are mandatory:

- Change the definition of CONNECTOR_DATASERVICES, listing your existing clusters (See below for examples).
- Change the definition of CONNECTOR_NAME should you wish to rename the connector service (This is the name shown in cctrl)
- Define the hosts/IP addresses with extra_hosts if your docker environment does not resolve the hostnames
- By default, the connector will be running in Bridge Mode, Therefore, if enabling Proxy mode, uncomment the following user.map line within volumes

```
- ./conf/user.map:/opt/continuent/tungsten-connector/conf/user.map
```

- If enabling SSL, uncomment the following two security related lines within volumes

```
- ./conf/security.properties:/opt/continuent/cluster-home/conf/security.properties
- ./security:/opt/continuent/share
```

The CONNECTOR_DATASERVICES property will need defining slightly differently depending upon your topology, the example syntax below can be used

- Single Cluster

```
CONNECTOR_DATASERVICES="alpha:db1,db2,db3"
```

- Composite Active/Passive Cluster

```
CONNECTOR_DATASERVICES="alpha:db1,db2,db3\nbeta:db4,db5,db6"
```

- Composite Active/Active Cluster

```
CONNECTOR_DATASERVICES="alpha:db1,db2,db3\nbeta:db4,db5,db6\nalpha_from_beta:db1,db2,db3\nbeta_from_alpha:db4,db5,db6"
```

Proxy Mode Setup

By default, the connector will run in Bridge mode. If you wish to change this to Proxy mode, follow the following steps:

- Uncomment the following user.map line in the `docker-compose.yml` file:

```
- ./conf/user.map:/opt/continuent/tungsten-connector/conf/user.map
```

- Edit the `conf/user.map` file and add every user required to the file

  Single Cluster Example:

```
app_user secret alpha
```

  Composite Active Passive Example:

```
app_user secret global alpha
```

  Composite Active/Active Example:

```
app_user secret global alpha,beta:beta,alpha
```

- Change the bridgeMode settings in the `conf/connector.properties` to:

```
bridgeMode=OFF
```

Enabling Smartscale

To specifically enable Smartscale, within the `conf/connector.properties` file, change the `useSmartScale` property to true

SSL Setup

By default, SSL will be disabled. To enable it, follow these steps:

- Uncomment the following lines in the `docker-compose.yml` file:

```
- ./conf/security.properties:/opt/continuent/cluster-home/conf/security.properties
- ./security:/opt/continuent/share
```

- In the `conf/connector.properties` change the following properties to true:

```
ssl.capable=true
jdbc.driver.options=useSSL=true
api.useSsl=true
```

- From a running Tungsten Cluster copy the following files from `/opt/continuent/share` folder to the `security/` folder:

```
passwords.store
tungsten_keystore.jks
tungsten_truststore.ts
tungsten_connector_keystore.jks
tungsten_connector_truststore.ts
```

Step 3 : Import the Docker image

Use the following command to import the image into docker:

```
docker load --input tungsten-connector-docker-image-7.1.3-3.tar.gz
```

Step 4 : Launch the connector

Once the configuration has been applied and the image imported, you can then launch the connector using the following syntax:

```
docker compose up -d
```

7.13. Connector General Limitations

The following general limitations exist when using the Connector; these issues do not affect the connector when using bridge mode:

- When using mysqldump within MySQL 5.6 or later, the `--single-transaction` option is not supported when connectivity to the database is made through the connector.
- When using mysqldump, the `--flush-logs` option is not supported when connectivity to the database is made through the connector.

Chapter 8. Tungsten Manager

The Tungsten Manager provides the management and monitoring of the Tungsten Cluster services to ensure that datasources, connectors and other components are running, datasources are replicating to each other, and handles failover and maintenance schedules.

8.1. Tungsten Manager Introduction

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

Primarily, the Tungsten Manager handles the following tasks:

- Monitors the replication status of each datasource within the cluster.
- Communicates and updates Tungsten Connector with information about the status of each datasource. In the event of a change of status, Tungsten Connectors are notified so that queries can be redirected accordingly.
- Manages all the individual components of the system. Using the Java JMX system the manager is able to directly control the different components to change status, control the replication process, and
- Checks to determine the availability of datasources by using either the system ping protocol (default), or using the Echo TCP/IP protocol on port 7 to determine whether a host is available. The configuration of the protocol to be used can be made by adjusting the manager properties. For more information, see Section B.3.3.3, “Host Availability Checks”.
- Includes an advanced rules engine. The rule engine is used to respond to different events within the cluster and perform the necessary operations to keep the dataservice in optimal working state. During any change in status, whether user-selected or automatically triggered due to a failure, the rules are used to make decisions about whether to restart services, swap Primaries, or reconfigure connectors.

In order to be able to avoid split brain, a cluster needs an odd number of members such that if there is a network partition, there's always a chance that a majority of the members are in one of the network partitions. If there is not a majority, it's not possible to establish a quorum and the partition with the Primary, and no majority, will end up with a shunned Primary until such time a quorum is established.

To operate with an even number of database nodes, a witness node is required. For more information see xref linkend="deployment-host-types-witness"/>

8.1.1. How Does Failover Work?

Did you ever wonder just what the Tungsten Manager is thinking when it does an automatic failover or a manual switch in a cluster?

What factors are taken into account by the Tungsten Manager it picks a replica to fail over to?

This page will detail the steps the Tungsten Manager to perform a switch or failover.

This section covers both the process and some possible reasons why that process might not complete, along with best practices and ways to monitor the cluster for each situation.

8.1.1.1. Roles for Nodes and Clusters

When we say “role” in the context of a cluster datasource, we are talking about the view of a database node from the Tungsten Manager's perspective.

These roles apply to the node datasource at the local (physical) cluster level, and to the composite datasource at the composite cluster level.

Possible roles are:

- Primary
  - A database node which is writable, or
  - A composite cluster which is active (contains a writable primary)
- Relay
  - A read-only database node which pulls data from a remote cluster and shares it with downstream replicas in the same cluster
- Replica
  - A read-only database node which pulls data from a local-cluster primary node, or from a local-cluster relay node for passive composite clusters
  - A composite cluster which is passive (contains a relay but NO writable primary)

8.1.1.2. Moving the Primary Role to Another Node or Cluster

One of the great powers of the Tungsten Cluster is that the roles for both cluster nodes and composite cluster datasources can be moved to another node or cluster, either at will via the cctrl switch command, or by having an automatic failover invoked by the Tungsten Manager layer.

Please note that while failovers are normally automatic and triggered by the Tungsten Manager, a failover can be also be invoked manually via the cctrl command if ever needed.

8.1.1.2.1. Switch versus Failover

There are key differences between the manual switch and automatic failover operations:

- Switch
  - Switch attempts to perform the operation as gracefully as possible, so there will be a delay as all of the steps are followed to ensure zero data loss
  - When the switch sub-command is invoked within cctrl, the Manager will cleanly close connections and ensure replication is caught up before moving the Primary role to another node
  - Switch recovers the original Primary to be a Replica
  - Please see Section 6.5.2, “Manual Primary Switch”.
- Failover
  - Failover is immediate, and could possibly result in data loss, even though we do everything we can to get all events moved to the new Primary
  - Failover leaves the original primary in a SHUNNED state
  - Connections are closed immediately
  - Use the cctrl recover command to make the failed Primary into a Replica once it is healthy
  - Please see both Section 6.5.1, “Automatic Primary Failover” and Section 8.2, “Tungsten Manager Failover Behavior”

For even more details, please visit: Section 6.5, “Switching Primary Hosts”

8.1.1.2.2. Which Target Replica Node To Use?

Picking a target replica node from a pool of candidate database replicas involves several checks and decisions.

For switch commands for both physical and composite services, the user has the ability to pass in the name of the physical or composite replica that is to be the target of the switch. If no target is passed in, or if the operation is an automatic failover, then the Manager has logic to identify the 'most up to date' replica which then becomes the target of the switch or failover.

Here are the choices to pick a new primary database node from available replicas, in order:

1. Skip any replica that is either not online or that is not a standby replica.
2. Skip any replica that has its status set to ARCHIVE
3. Skip any replica that does not have an online manager.
4. Skip any replica that does not have a replicator in either online or synchronizing state.
5. Now we have a target datasource prospect...
6. By comparing the last applied sequence number of the current target datasource prospect to any other previously seen prospect, we should eventually end up with a replica that has the highest applied sequence number. We also save the prospect that has the highest stored sequence number.
7. If we find that there is a tie in the highest sequence number that has been applied or stored by any prospect with another prospect, we compare the datasource precedence and if there's a difference in this precedence, we choose the datasource with the lowest precedence number i.e. a precedence of 1 is higher than a precedence of 2. If we have a tie in precedence, select the last replica chosen and discard the replica currently being evaluated.
8. After we have evaluated all of the replicas, we will either have a single winner or we may have a case where we have one replica that has the highest applied sequence number but we have another replica that has the highest stored sequence number i.e. it has gotten the most number of THL records from the primary prior to the switch operation. In this case, and this is particularly important in cases of failover, we choose the replica that has the highest number of stored THL records.

9. Skip any replica that has a latency higher than the configured threshold. If too far behind, do not use that replica. The tpm option property=policy.slave.promotion.latency.threshold=900 controls the check, with 900 seconds as the default value.
10. At this point return to the switch or failover command whatever target replica we have chosen so that the operation can proceed.

After looping over all available replicas, check the selected target Replica's appliedLatency to see if it is higher than the configured threshold (default: 900 seconds). If the appliedLatency is too far behind, do not use that Replica. The tpm option `property=policy.slave.promotion.latency.threshold=900` [443] controls the check.

If no viable Replica is found (or if there is no available Replica to begin with), there will be no switch or failover at this point.

For more details on automatic failover versus manual switch, please visit: Section 8.4.1, "Manual Switch Versus Automatic Failover"

8.1.1.2.3. Switch and Failover Steps for Local Clusters

For more details on switch and failover steps for local clusters, please visit:

- Section 6.5, "Switching Primary Hosts"
- Section 8.4.2, "Switch and Failover Steps for Local Clusters"

8.1.1.2.4. Switch and Failover Steps for Composite Services

For more details on switch and failover steps for composite services, please visit:

- Section 6.7, "Composite Cluster Switching, Failover and Recovery"
- Section 8.4.3, "Switch and Failover Steps for Composite Services"

8.1.2. Best Practices for Proper Cluster Failovers

What are the best practices for ensuring the cluster always behaves as expected? Are there any reasons for a cluster NOT to fail over? If so, what are they?

Here are three common reasons that a cluster might not failover properly:

- Policy Not Automatic
  - BEST PRACTICE: Ensure the cluster policy is *`AUTOMATIC`* unless you specifically need it to be otherwise
  - SOLUTION: Use the check_tungsten_policy command to verify the policy status
- Complete Network Partition
  - If the nodes are unable to communicate cluster-wide, then all nodes will go into a `FAILSAFE-SHUN` mode to protect the data from a split-brain situation.
  - BEST PRACTICE: Ensure that all nodes are able to see each other via the required network ports
  - SOLUTION: Verify that all required ports are open between all nodes local and remote - see Section B.3.3.1, "Network Ports"
  - SOLUTION: Use the check_tungsten_online command to check the DataSource State on each node
- No Available Replica
  - BEST PRACTICE: Ensure there is at least one `ONLINE` node that is not in *`STANDBY`* or *`ARCHIVE`* mode

    SOLUTION: Use the check_tungsten_online command to check the DataSource State on each node
  - BEST PRACTICE: Ensure that the Manager is running on all nodes

    SOLUTION: Use the check_tungsten_services command to verify that the Tungsten processes are running on each node
  - BEST PRACTICE: Ensure all Replicators are either `ONLINE` or `GOING ONLINE:SYNCHRONIZING`

    SOLUTION: Use the check_tungsten_online command to verify that the Replicator (and Manager) is ONLINE on each node
  - BEST PRACTICE: Ensure the replication applied latency is under the threshold, default 900 seconds

    SOLUTION: Use the check_tungsten_latency command to check the latency on each node

8.1.3. Command-Line Monitoring Tools

Below are examples of all the health-check tools listed above:

- check_tungsten_services

```
shell> check_tungsten_services -c -r
CRITICAL: Connector, Manager, Replicator are not running

shell> startall
Starting Replicator normally
Starting Tungsten Replicator Service...
Waiting for Tungsten Replicator Service.......
running: PID:14628
Starting Tungsten Manager Service...
Waiting for Tungsten Manager Service.........
running: PID:15143
Starting Tungsten Connector Service...
Waiting for Tungsten Connector Service.......
running: PID:15513

shell> check_tungsten_services -c -r
OK: All services (Connector, Manager, Replicator) are running
```

- check_tungsten_policy

```
shell> check_tungsten_policy
CRITICAL: Manager is not running

shell> manager start

shell> check_tungsten_policy
CRITICAL: Policy is MAINTENANCE

shell> cctrl
cctrl> set policy automatic
cctrl> exit

shell> check_tungsten_policy
OK: Policy is AUTOMATIC
```

- check_tungsten_latency

```
shell> check_tungsten_latency -w 100 -c 200
CRITICAL: Manager is not running

shell> manager start

shell> check_tungsten_latency -w 100 -c 200
CRITICAL: db8=65107.901s, db9 is missing latency information

shell> cctrl
cctrl> cluster heartbeat
cctrl> exit

shell> check_tungsten_latency -w 100 -c 200
WARNING: db9 is missing latency information

shell> cctrl
cctrl> set policy automatic
cctrl> exit

shell> check_tungsten_latency -w 100 -c 200
OK: All replicas are running normally (max_latency=4.511)
```

- check_tungsten_online

```
shell> check_tungsten_online
CRITICAL: Manager is not running

shell> manager start

shell> check_tungsten_online
CRITICAL: Replicator is not running

shell> replicator start

shell> check_tungsten_online
CRITICAL: db9 REPLICATION SERVICE north is not ONLINE

shell> trepctl online

shell> check_tungsten_online
CRITICAL: db9 REPLICATION SERVICE north is not ONLINE
```

Related Pages:

- Section 8.2, “Tungsten Manager Failover Behavior”
- Section 8.3, “Tungsten Manager Failover Tuning”
- Section 8.4, “Tungsten Manager Failover Internals”
- Section B.3.3.1, “Network Ports”

8.2. Tungsten Manager Failover Behavior

The Tungsten Cluster default position is to protect the integrity of the data. Sometimes this means that a failover may be delayed. It is more important to have all the data than for the write Primary to be available. This behavior is configurable.

When the Manager quorum invokes a failover automatically, there are various states that each replication stream may be in that must be taken into account by the Manager layer as it coordinates the failover.

This is due to the loosely-coupled, asynchronous nature of Tungsten Replicator.

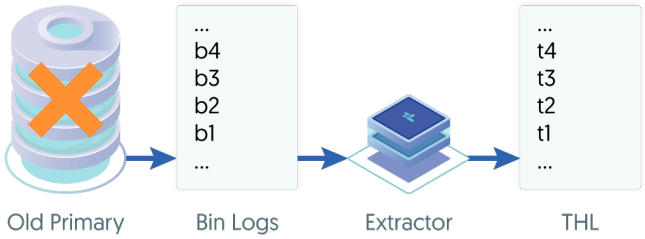
8.2.1. Failover Replication State Scenarios

The multiple scenarios regarding the state of the data are described below, along with how to locate any associated events:

- First, the normal situation - all events in the binary logs have been extracted into the Primary THL, all Primary THL has been transferred to all Replica nodes, and all THL on the Replica nodes has been fully applied to all Replica databases.
- Scenario 1 - There are orphaned MySQL binary log events on the old Primary that were not extracted into THL before a failover occurred.

  This could occur when `stoponDBerror=true` or when the MySQL binlogs are not available (i.e. due to a host crash). The scenario would be reported upon issuing the cctrl recover command.

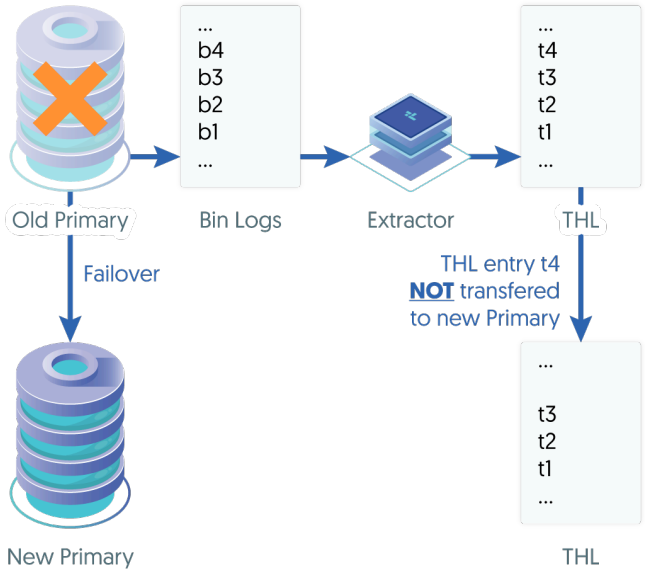
  Figure 8.1. Failover Scenario 1

  The tungsten_find_orphaned script can help locate orphaned binary log events that did not make it to the THL on the old Primary before a failover. For more information, please see Scenario 1 in Section 9.25, “The tungsten_find_orphaned Command”.

- Scenario 2 - There is orphaned THL left on the old Primary that did not make it to the new Primary

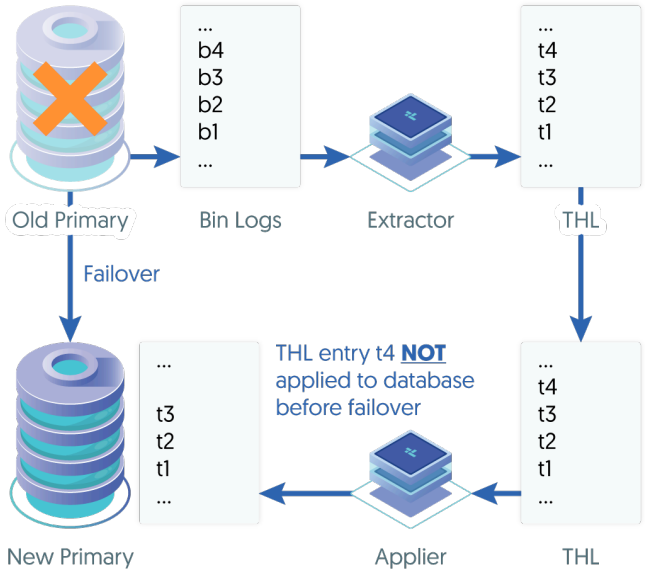
Figure 8.2. Failover Scenario 2



The tungsten_find_orphaned script can help locate orphaned Primary THL left on the old Primary that did not make it to the new Primary before a failover. For more information, please see Scenario 2 in Section 9.25, "The tungsten_find_orphaned Command".

- Scenario 3 - There is orphaned Replica THL on the new Primary node that did not get applied into the database before getting promoted to Primary role

Figure 8.3. Failover Scenario 3



The tungsten_find_orphaned script can help locate orphaned Replica THL on the new Primary node that did not get applied into the database before getting promoted to Primary role during a failover. For more information, please see Scenario 3 in Section 9.25, "The tungsten_find_orphaned Command".

8.2.2. Recovery Behavior After Failover

The Tungsten Manager is able to detect un-extracted, desirable binary log events when recovering the old Primary via cctrl after a failover.

The cctrl recover command will fail if:

- any unextracted binlog events exist on the old Primary that we are trying to recover (Scenario 1)
- the old Primary THL contains more events than the Replica that will be promoted to a new Primary (Scenario 2)

In this case, the cctrl recover command will display text similar to the following:

```
Recovery failed because the failed master has unextracted events in
the binlog. Please run the tungsten_find_orphaned script to inspect
this events. Provided you have a recent backup available, you can
try to restore the data source by issuing the following command:
              datasource {hostname} restore
Please consult the user manual at:
https://docs.continuent.com/tungsten-clustering-6.1/operations-restore.html
```

The tungsten_find_orphaned script is designed to locate orphaned events. For more information, please see Section 9.25, “The tungsten_find_orphaned Command”.

8.2.3. Failover Response when MySQL Server Fails

The Manager and Replicator behave in concert when MySQL dies on the Primary node. When this happens, the replicator is unable to update the `trep_commit_seqno` table any longer, and therefore must either abort extraction or continue extracting without recording the extracted position into the database.

By default, the Manager will delay failover until all remaining events have been extracted from the binary logs on the failing Primary node as a way to protect data integrity.

This behavior is configurable via the following setting, shown with the default of `false`:

```
property=replicator.store.thl.stopOnDBError=false
```

Failover will only continue once:

- all available events are completely read from the binary logs on the Primary node
- all events have reached the Replicas

When `property=replicator.store.thl.stopOnDBError=true` [443], then the Replicator will stop extracting once it is unable to update the `trep_commit_seqno` table in MySQL, and the Manager will perform the failover without waiting, at the risk of possible data loss due to leaving binlog events behind. All such situations are logged.

For use cases where failover speed is more important than data accuracy, those NOT willing to wait for long failover can set `property=replicator.store.thl.stopOnDBError=true` [443] and still use tungsten_find_orphaned to manually analyze and perform the data recovery. For more information, please see Section 9.25, “The tungsten_find_orphaned Command”.

8.2.4. Failover Response when Replica Applier is Latent

During a failover, the manager will wait until the Replica that is the candidate for promotion to Primary has applied all stored THL events before promoting that node to Primary.

This wait time can be configured via the `property=manager.failover.thl.apply.wait.timeout=0` property.

The default value is 0, which means "wait indefinitely until all stored THL events are applied".

Any value other than zero invites data loss due to the fact that once the Replica is promoted to Primary, any unapplied stored events in the THL will be ignored, and therefore lost.

Whenever a failover occurs, the Replica with most events stored in the local THL is selected so that when the events are eventually applied, the data is as close to the original Primary as possible with the least number of events missed.

That is usually, but not always, the most up-to-date Replica, which is the one with the most events applied.

There should be a good balance between the value for `property=manager.failover.thl.apply.wait.timeout` and the value for `property=policy.slave.promotion.latency.threshold=900`, which is the "maximum Replica latency" - this means the number of seconds to which a Replica must be current with the Primary in order to qualify as a candidate for failover. The default is 15 minutes (900 seconds).

8.2.5. Understanding the `FAILSAFE-SHUN` state.

A Tungsten Cluster requires an odd number of database nodes to establish a voting quorum using group communications—typically, this means a three-node cluster. This quorum mechanism ensures that, in the event of a primary database failure, the system can automatically and correctly failover to a healthy node.

Because Tungsten Cluster's quorum relies on jGroups protocols for communication, a highly reliable network is essential for proper functionality. While the cluster is designed to handle intermittent network failures, there may be situations where the Tungsten Manager executes a fail-safe operation to protect data integrity. In such cases, database nodes may isolate themselves from client traffic to prevent a split-brain scenario.

Although the Tungsten Cluster is built to withstand many hardware and software failures, the underlying network reliability ultimately defines the limits of its fault-tolerance capabilities.

This section will cover the basic steps the Tungsten Manager takes when a network failure isolates it from the other nodes—helping you better understand how `FAILSAFE-SHUN` works and why it matters.

8.2.5.1. What causes a `FAILSAFE-SHUN`

In the case of a network failure, on what basis does Tungsten initiate a fail-safe shun? Is it triggered by ICMP unreachability or JGroups unreachability?

Failover only happens if the manager is in a non-primary jGroups partition:

```
2025/02/02 23:12:36 | CHECKING FOR QUORUM: MUST BE AT LEAST 2 DB MEMBERS
2025/02/02 23:12:36 | POTENTIAL QUORUM SET MEMBERS ARE: db1, db3, db2
2025/02/02 23:12:36 | SIMPLE MAJORITY SIZE: 2
2025/02/02 23:12:36 | GC VIEW OF CURRENT DB MEMBERS IS: db1
2025/02/02 23:12:36 | VALIDATED DB MEMBERS ARE: db1
2025/02/02 23:12:36 | REACHABLE DB MEMBERS ARE: db1
2025/02/02 23:12:36 |
=============================================================
2025/02/02 23:12:36 | CONCLUSION: I AM IN A NON-PRIMARY PARTITION OF 1 MEMBERS OUT OF A REQUIRED MAJORITY SIZE OF 2
2025/02/02 23:12:36 | AND THERE ARE 0 REACHABLE WITNESSES OUT OF 0
2025/02/02 23:12:36 | WARN | db1 | WARN [EnterpriseRulesConsequenceDispatcher] - COULD NOT ESTABLISH A QUORUM. RESTARTING SAFE…
```

A partition (a group) is formed by JGroups. When a manager first starts, it looks for a group to join. If a group is not available, then the Manager will create a group. The other Managers that start after the first manager will join the group. If a Manager is shut down, then it will leave the group. The logs show that the db1 manager was alone in the group and restarted in failsafe-shun mode.

But how did we arrive here? Every 3 seconds a heartbeat is sent to the other nodes. If there is no reply for 10 seconds, then a heartbeat gap detection rule will fire:

```
2025/02/02 23:12:14 | INFO | db1 | INFO [EnterpriseRulesConsequenceDispatcher] - HEARTBEAT GAP DETECTED FOR MEMBER 'db2'
2025/02/02 23:12:16 | INFO | db1 | INFO [EnterpriseRulesConsequenceDispatcher] - HEARTBEAT GAP DETECTED FOR MEMBER 'db3'
2025/02/02 23:12:16 | NETWORK CONNECTIVITY: PING TIMEOUT=5
2025/02/02 23:12:16 | NETWORK CONNECTIVITY: CHECKING MY OWN ('db1') CONNECTIVITY
2025/02/02 23:12:16 | HOST db1/10.88.75.104: ALIVE
2025/02/02 23:12:16 | (ping) result: true, duration: 0.01s, notes: ping -c 1 -w 5 10.88.75.104
2025/02/02 23:12:16 | NETWORK CONNECTIVITY: CHECKING CLUSTER MEMBER 'db2'
2025/02/02 23:12:21 | HOST db2/10.88.75.105: NOT REACHABLE
2025/02/02 23:12:21 | (ping) result: false, duration: 5.01s, notes: ping -c 1 -w 5 10.88.75.105
2025/02/02 23:12:21 | NETWORK CONNECTIVITY: CHECKING CLUSTER MEMBER 'db3'
2025/02/02 23:12:26 | HOST db3/10.88.75.106: NOT REACHABLE
2025/02/02 23:12:26 | (ping) result: false, duration: 5.00s, notes: ping -c 1 -w 5 10.88.75.106
```

In this case, as seen below, db1 viewed db2 and db3 as down or not reachable because the network was down. This situation will then trigger a `MembershipInvalidAlarm`. The alarm will check if the other managers are reachable through Group communication or not. If they are not reachable via jGroups, the manager will increment the dispatch number and after 10 seconds will check again. When the dispatch number equals the max dispatch AND the node is still seen as alone in the cluster, the manager will restart in `FAILSAFE-SHUN` mode.

```
2025/02/02 23:12:36 | INFO | db1 | INFO [Rule_0550$u58$_INVESTIGATE$u58$_TIME_KEEPER_FOR_INVALID_MEMBERSHIP_ALARM71605603] - TIMER EXPIRED, INCREMENTED AL
2025/02/02 23:12:36 | INFO | db1 | INFO [Rule_0530$u58$_INVESTIGATE_MEMBERSHIP_VALIDITY1008640238] - CONSEQUENCE
2025/02/02 23:12:36 | NETWORK CONNECTIVITY: PING TIMEOUT=5
2025/02/02 23:12:36 | NETWORK CONNECTIVITY: CHECKING MY OWN ('db1') CONNECTIVITY
2025/02/02 23:12:36 | HOST db1/10.88.75.104: ALIVE
2025/02/02 23:12:36 | (ping) result: true, duration: 0.00s, notes: ping -c 1 -w 2 10.88.75.104
2025/02/02 23:12:36 | NETWORK CONNECTIVITY: CHECKING CLUSTER MEMBER 'db2'
2025/02/02 23:12:36 | HOST db2/10.88.75.105: ALIVE
2025/02/02 23:12:36 | (ping) result: true, duration: 0.00s, notes: ping -c 1 -w 5 10.88.75.105
2025/02/02 23:12:36 |
=============================================================
2025/02/02 23:12:36 | CHECKING FOR QUORUM: MUST BE AT LEAST 2 DB MEMBERS
2025/02/02 23:12:36 | POTENTIAL QUORUM SET MEMBERS ARE: db1, db3, db2
2025/02/02 23:12:36 | SIMPLE MAJORITY SIZE: 2
2025/02/02 23:12:36 | GC VIEW OF CURRENT DB MEMBERS IS: db1
2025/02/02 23:12:36 | VALIDATED DB MEMBERS ARE: db1
2025/02/02 23:12:36 | REACHABLE DB MEMBERS ARE: db1
2025/02/02 23:12:36 |
=============================================================
2025/02/02 23:12:36 | CONCLUSION: I AM IN A NON-PRIMARY PARTITION OF 1 MEMBERS OUT OF A REQUIRED MAJORITY SIZE OF 2
2025/02/02 23:12:36 | AND THERE ARE 0 REACHABLE WITNESSES OUT OF 0
```

```
2025/02/02 23:12:36 | WARN | db1 | WARN [EnterpriseRulesConsequenceDispatcher] - COULD NOT ESTABLISH A QUORUM. RESTARTING SAFE…
```

The Tungsten Manager initiates a `FAILSAFE-SHUN` when a `MemberHearbeatGap` invokes a `MembershipInvalidAlarm`. Only a `MembershipInvalidAlarm` can cause a `FAILSAFE-SHUN`, and only if the host is alone in the group.

Note that the values above are `MAX DISPATCH=3` and `DISPATCH=3`. The max dispatch value is controlled by the `policy.invalid.membership.retry.threshold` property.

In this case the network was online for 20 seconds between db1 and db2 (but not for db3) as we can see from the logs (at 23:12:33):

```
2025/02/02 23:12:33 | NETWORK CONNECTIVITY: PING TIMEOUT=5
2025/02/02 23:12:33 | NETWORK CONNECTIVITY: CHECKING MY OWN ('db1') CONNECTIVITY
2025/02/02 23:12:33 | HOST db1/10.88.75.104: ALIVE
2025/02/02 23:12:33 | (ping) result: true, duration: 0.00s, notes: ping -c 1 -w 2 10.88.75.104
2025/02/02 23:12:33 | NETWORK CONNECTIVITY: CHECKING CLUSTER MEMBER 'db2'
2025/02/02 23:12:33 | HOST db2/10.88.75.105: ALIVE
2025/02/02 23:12:33 | (ping) result: true, duration: 0.00s, notes: ping -c 1 -w 5 10.88.75.105
```

At this point it was too late for the group communications to re-establish the quorum and form a group, so the Manager restarted at 23:12:36.

8.2.5.2. How to tune the delay before `FAILSAFE-SHUN`

To increase the delay before a `FAILSAFE-SHUN` is invoked, set the `policy.invalid.membership.retry.threshold` property in your `/etc/tungsten/tungsten.ini` and run tpm update. For example, setting this value to 6 would result in the Managers being able to survive a network blip that lasts a maximum of 50 seconds:

```
property=policy.invalid.membership.retry.threshold=6
```

The formula works out to: `6 x 10 = 60 - 10` (safety period for jGroups to re-form the group) = 50-second delay before fail-safe shun occurs.

8.2.5.3. FAQ

The following sections provide the questions and answers to questions often asked by customers.

8.2.5.3.1 Does an ICMP packet drop relate to an invalid membership issue in Tungsten?

No. Only a `MemberHeartbeatGap` (not getting a heartbeat reply for 10 seconds) will trigger a `MembershipInvalidAlarm`.

8.2.5.3.2 Does the coordinator decide to initiate a `FAILSAFE-SHUN` across the cluster, or does each host's Manager independently invoke the fail-safe script?

Each Manager independently invokes the fail-safe script if they are alone in the quorum.

8.2.5.3.3 We observed that DB1 initiated a `FAILSAFE-SHUN` at 23:12:36, DB2 at 23:12:33, and DB3 at 23:12:35. Based on the current settings, can you analyze the logs to determine the exact duration of the network blip before the `FAILSAFE-SHUN` was triggered across all three database nodes?

This can be seen only from your network monitoring logs. In the Manager logs we can see only the response to the `"ping -c 1 -w 5 hostname"` result during the membership validity check when we increment the timer, but during this 10-second period the network can be flapping (up-down-up-down), which is NOT visible from the Manager logs. What we can see during the membership validity check is that the network at that exact time was either up or down.

8.2.5.3.4 If we change the property `policy.invalid.membership.retry.threshold=6`, what would be the potential impact on failover behavior or data loss?

Failover can happen for the following reasons:

1. if the primary MySQL server goes down. This has nothing to do with our case.
2. if the host with the primary MySQL server goes down. This has nothing to do with our case.
3. If the host with the primary MySQL server gets isolated AND the other two hosts can still form a majority of the quorum. This has nothing to do with our case.
4. If the host with the primary MySQL server gets isolated AND also the other hosts get isolated from each other. In this case failover does not happen (we have only singular managers isolated from each other) and the cluster will go to failsafe shun. This is our case.

This means that setting the threshold to 6 will only result in keeping the primary online to resist the network blip. Data loss after a failover can happen ONLY if the operator of the cluster issues a recover command in force mode without paying any attention to the

error displayed. Error is displayed when there are transactions left in the primary binlog that were not replicated to the new Primary. In this case the recover command will issue an error and will not proceed. The operator needs to manually apply the transactions to the new Primary. If the operator forces the recover, data loss will occur.

8.2.5.3.5 Apart from the heartbeat gap detection invalid alarm, where can we find logs in Tungsten that indicate non-validated database members?

The message `"Non-validated database members"` in the Manager's `tmsvc.log` means that there wasn't a Manager on that host up and running or because of the network breakage that Manager is not reachable.

8.2.5.3.6 What is the difference between a `FAILSAFE-SHUN` invocation and an automatic failover initiation when the first node detects a network issue? If the affected node is the primary, will it trigger an automatic failover, or a `FAILSAFE-SHUN`? If the affected node is a secondary, will it only trigger a `FAILSAFE-SHUN`? Here are the associated logs:

From DB1:

```
2025/02/02 23:12:33 | GC VIEW OF CURRENT DB MEMBERS IS: db1, db2, db3
2025/02/02 23:12:33 | VALIDATED DB MEMBERS ARE: db1
2025/02/02 23:12:33 | REACHABLE DB MEMBERS ARE: db1, db3, db2
2025/02/02 23:12:36 | GC VIEW OF CURRENT DB MEMBERS IS: db1, db2, db3
2025/02/02 23:12:36 | VALIDATED DB MEMBERS ARE: db1, db3
2025/02/02 23:12:36 | REACHABLE DB MEMBERS ARE: db1, db3, db2
2025/02/02 23:12:36 | GC VIEW OF CURRENT DB MEMBERS IS: db1
2025/02/02 23:12:36 | VALIDATED DB MEMBERS ARE: db1
2025/02/02 23:12:36 | REACHABLE DB MEMBERS ARE: db1
2025/02/02 23:12:36 | FATAL | db1 | FATAL [ClusterManagementHelper] - Fail-safe invoked. Forcing a failsafe exit followed by a restart.
```

From DB2:

```
2025/02/02 23:12:28 | GC VIEW OF CURRENT DB MEMBERS IS: db2
2025/02/02 23:12:28 | VALIDATED DB MEMBERS ARE: db2
2025/02/02 23:12:28 | REACHABLE DB MEMBERS ARE: db2
2025/02/02 23:12:33 | GC VIEW OF CURRENT DB MEMBERS IS: db2
2025/02/02 23:12:33 | VALIDATED DB MEMBERS ARE: db2
2025/02/02 23:12:33 | REACHABLE DB MEMBERS ARE: db2
2025/02/02 23:12:33 | FATAL | db2 | FATAL [ClusterManagementHelper] - Fail-safe invoked. Forcing a failsafe exit followed by a restart.
```

From DB3:

```
2025/02/02 23:12:35 | GC VIEW OF CURRENT DB MEMBERS IS: db1, db2, db3
2025/02/02 23:12:35 | VALIDATED DB MEMBERS ARE: db1, db3
2025/02/02 23:12:35 | REACHABLE DB MEMBERS ARE: db1, db3, db2
2025/02/02 23:12:35 | GC VIEW OF CURRENT DB MEMBERS IS: db3
2025/02/02 23:12:35 | VALIDATED DB MEMBERS ARE: db3
2025/02/02 23:12:35 | REACHABLE DB MEMBERS ARE: db3
2025/02/02 23:12:35 | FATAL | db3 | FATAL [ClusterManagementHelper] - Fail-safe invoked. Forcing a failsafe exit followed by a restart.
```

Automatic failover is initiated by the coordinator. For a coordinator to be able to failover, it must be in a partition that has a majority of nodes. If every Manager is in its own single partition, no failover can be expected. Only a `FAILSAFE-SHUN` can happen at this point.

8.2.5.3.7 Does an ICMP packet drop cause a `FAILSAFE-SHUN` to initiate?

No. A `FAILSAFE-SHUN` will only happen if the Manager is alone in the JGroups partition.

8.2.5.3.8 In the case of a network failure, on what basis does Tungsten initiate the initial `FAILSAFE-SHUN`? Is it triggered by ICMP unreachability or JGroups unreachability?

Tungsten initiates a `FAILSAFE-SHUN` when a `MemberHearbeatGap` invokes a `MembershipInvalidAlarm`. Only a `MembershipInvalidAlarm` can cause a `FAILSAFE-SHUN`, and only if the host is alone in the group.

8.2.5.3.9 In some cases, we did not observe a ping failure in the Tungsten logs, yet Tungsten still initiated a `FAILSAFE-SHUN`. If this is due to JGroups, how can we trace it through the tungsten logs beyond generic messages like `"Quorum lost"` and similar entries?

`FAILSAFE-SHUN` can happen even if there wasn't a ping failure (e.g. the host is up and running) but the other Managers are down (stopped, restarted).

8.2.5.3.10 What are your recommendations for moving from ICMP to a TCP-based network health check in Tungsten? What are the pros and cons of this transition?

Not recommended - ICMP is the best practice as seen over years of experience in the field.

When ICMP is failing, TCP will fail too. If a network is unstable, using TCP to somehow mask the instability is not the correct approach to checking network health for a database cluster.

Our implementation of JGroups uses TCP to communicate with the group members and the ping utility uses ICMP.

8.2.5.3.11. Packet drops and overrun errors do not include timestamps, and other services remained unaffected while only Tungsten was impacted. How can we analyse Tungsten logs to determine whether the issue was caused by a network problem or network blip , especially given the lack of detailed JGroups logging?

The correct way to analyze the Manager logs is to follow the provided example above. Start by locating the triggered events and then follow the included explanations.

Grep for `HEARTBEAT GAP DETECTED FOR MEMBER` and look for which events trigger other events.

Also, JGroups logging is disabled by default. The best practice is to keep JGroups logging disabled because it will rapidly fill the logs and is very difficult to interpret.

8.2.5.3.12. Which protocol does the Heartbeat mechanism use to send signals in the Tungsten system (TCP, ICMP, or UDP)?

Heartbeat events are sent via TCP

8.2.5.3.13. Which protocol does JGroups use by default (TCP or UDP)?

Our jGroups implementation uses TCP to communicate.

8.2.5.3.14. What is the difference between the following entries in Tungsten logs? `GC VIEW OF CURRENT DB MEMBERS`, `VALIDATED DB MEMBERS`, `REACHABLE DB MEMBERS`

- `GC VIEW OF CURRENT DB MEMBERS IS: db1, db2, db3` - this is how the group communication (JGroups GC) sees members
- `VALIDATED DB MEMBERS ARE: db1, db3` - managers that are up and running and respond to the pingMember command (TCP)
- `REACHABLE DB MEMBERS ARE: db1, db3, db2` - host reachable by ping (host ping utility ICMP)

8.3. Tungsten Manager Failover Tuning

There are currently three discrete faults that can cause a failover of a Primary:

- Database server failure - failover will occur 20 seconds after the initial detection.

  `property=policy.liveness.dbping.fail.threshold=1` [443]

  The Tungsten Manager is unable to connect to the database server and gets an i/o error. If the database cannot respond to a tcp connect request after the configured number of attempts, the database server is flagged as `STOPPED` which initiates the failover.

  This would mean, literally, that the process for the database server is gone and cannot respond to a tcp connect request. In this case, by default, the manager will try two more times, once every 10 seconds, after the initial i/o error is detected and after the then 30 second interval has elapsed, will flag the database server as being in the STOPPED state and this, in turn, initiates the failover.

- Host failure - failover will occur 30 seconds after the initial detection

  `property=policy.liveness.hostPing.fail.threshold=2` [443]

  The host on which the Primary database server is running is 'gone'. The first indication that the Primary host is gone could be because the manager on that host no longer appears in the group of managers, one of which runs on each database server host. It could also be that the managers on the hosts besides the Primary do not see a 'heartbeat' message from the Primary manager. In a variety of circumstances like this, both of the managers will, over a 60 second interval of time, once every 10 seconds, attempt to establish, definitively, that the Primary host is indeed either gone or completely unreachable via the network. If this is established, the remaining managers in the group will establish a quorum and the coordinator of that group will initiate failover.

- A replicator failure, if `property=policy.fence.masterReplicator` [443] is set to `true`, will cause a failover 70 seconds after initial detection

  `property=policy.fence.masterReplicator.threshold=6` [443]

  Depending on how you have the manager configured, a Primary replicator failure can also start a process of initiating a failover. There's a specific manager property (`property=policy.fence.masterReplicator=true` [443]) that tells a manager to 'fence' a Primary replicator that goes into either a failed or stopped state. The manager will then try to recover the Primary replicator to an online state and, again, after an interval of 60 seconds, if the Primary replicator does not recover, a failover will be initiated. By default, this behavior is turnd OFF. Most customers prefer to keep a fully functional Primary running, even if replication fails, rather than have a failover occur.

  > **Important**
  >
  > The interval of time from the first detection of a fault until a failover occurs is configurable over 10 second intervals. The formula for determining the listed default failover intervals is based on the value of 'threshold' in the properties file (`tungsten-manager/conf/manager.properties`): interval = (threshold + 1) * 10 seconds

Additionally, there are multiple ways to influence the behavior of the cluster AFTER a failover has been invoked. Below are some of the key variables:

- Behavior when MySQL is not available but the binary logs are - wait for the Replicator to finish extracting the binary logs or not?

  `property=replicator.store.thl.stopOnDBError=false` [443]

  The Manager and Replicator behave in concert when MySQL dies on the Primary node. When this happens, the replicator is unable to update the `trep_commit_seqno` table any longer, and therefore must either abort extraction or continue extracting without recording the extracted position into the database.

  The default of **`false`** means that the Manager will delay failover until all remaining events have been extracted from the binary logs on the failing Primary node as a way to protect data integrity.

  Failover will only continue once:

  - all available events are completely read from the binary logs on the Primary node
  - all events have reached the Replicas

  When `property=replicator.store.thl.stopOnDBError=true` [443], then the Replicator will stop extracting once it is unable to update the `trep_commit_seqno` table in MySQL, and the Manager will perform the failover without waiting, at the risk of possible data loss due to leaving binlog events behind. All such situations are logged.

  For use cases where failover speed is more important than data accuracy, those NOT willing to wait for long failover can set `property=replicator.store.thl.stopOnDBError=true` and still use tungsten_find_orphaned to manually analyze and perform the data recovery. For more information, please see Section 9.25, “The tungsten_find_orphaned Command”.

- Replica THL apply wait time before failover - how long to wait term of seconds for a Replica to finish applying all stored THL to the database before failing over to it.

  `property=manager.failover.thl.apply.wait.timeout=0` [443]

  During a failover, the manager will wait until the Replica that is the candidate for promotion to Primary has applied all stored THL events before promoting that node to Primary.

  The default value is 0, which means "wait indefinitely until all stored THL events are applied".

  > **Warning**
  >
  > Any value other than zero (0) invites data loss due to the fact that once the Replica is promoted to Primary, any unapplied stored events in the THL will be ignored, and therefore lost.

  Whenever a failover occurs, the Replica with most events stored in the local THL is selected so that when the events are eventually applied, the data is as close to the original Primary as possible with the least number of events missed.

  That is usually, but not always, the most up-to-date Replica, which is the one with the most events applied.

- Replica latency check - how far behind in term of seconds is each Replica? If too far behind, do not use for failover.

  `property=policy.slave.promotion.latency.threshold=900` [443]

  The `property=policy.slave.promotion.latency.threshold=900` option is the "maximum Replica latency" - this means the number of seconds to which a Replica must be current with the Primary in order to qualify as a candidate for failover. The default is 15 minutes (900 seconds).

8.4. Tungsten Manager Failover Internals

The process of replica selection during either a manual switch or an automatic failover involves a number of decisions about the health of the candidate replica node(s).

For the list of discrete faults that can cause a Primary failover, please visit: Section 8.3, “Tungsten Manager Failover Tuning”.

8.4.1. Manual Switch Versus Automatic Failover

This section describes the main differences between the manual and automatic scenarios.

Manual Switch

1. The entry point for manual switch/failover is the cctrl command line tool.

2. For switch commands for both physical and composite services, the user has the ability to pass in the name of the physical or composite replica that is to be the target of the switch.

   If no target is passed in, then the Manager has logic to identify the 'most up to date' replica which then becomes the target of the switch. Here are the steps that are taken in that logic:

   a. Skip any replica that is either not `ONLINE` or that is NOT a *`STANDBY`* replica.

   b. Skip any replica that has its status set to *`ARCHIVE`*

   c. Skip any replica that does not have an online manager.

   d. Skip any replica that does not have a replicator in either online or synchronizing state.

   e. Now we have a target datasource prospect.

   f. By comparing the last applied sequence number of the current target datasource prospect to any other previously seen prospect, we should eventually end up with a replica that has the highest applied sequence number. We also save the prospect that has the highest stored sequence number.

   g. If we find that there is a tie in the highest sequence number that has been applied or stored by any prospect with another prospect, we compare the datasource precedence and if there's a difference in this precedence, we choose the datasource with the lowest precedence number i.e. a precedence of 1 is higher than a precedence of 2. If we have a tie in precedence, select the last replica chosen and discard the replica currently being evaluated.

   h. After we have evaluated all of the replicas, we will either have a single winner or we may have a case where we have one replica that has the highest applied sequence number but we have another replica that has the highest stored sequence number i.e. it has gotten the most number of THL records from the primary prior to the switch operation. In this case, and this is particularly important in cases of failover, we choose the replica that has the highest number of stored THL records.

   i. At this point we return whatever target replica we have determined to the switch or failover command so that it can continue.

3. After looping over all available replicas, check the selected target replica's appliedLatency to see if it is higher than the configured threshold (default: 900 seconds). If the appliedLatency is too far behind, do not use that replica. The tpm option `property=policy.slave.promotion.latency.threshold=900` [443] controls the check.

4. If no viable replica is found (or if there is no available replica to begin with), there will be no switch or failover at this point.

   Once the target replica has been identified and validated, the switch/failover continues and follows the steps outlined in the Switch and Failover Steps section.

Automatic Failover

> **Important**
>
> NOTE: Automatic failover only applies to local (physical) primary or relay datasources.

1. The entry point for automatic failover (there is no such thing as an automatic switch) can be found in two rules:

   - `0200a: RECOVER PRIMARY DATASOURCE BY FAILING OVER - NON-REPLICATOR FAULT`
   - `0200b: RECOVER PRIMARY DATASOURCE BY FAILING OVER - REPLICATOR FAULT`

2. The main difference between these two rules is that `0200b` is qualified by having the existence of an expired ReplicatorFaultAlarm. Such an alarm will only exist if there is a non-default policy configuration that says 'I want to failover my primary if my replicator goes into a fault condition'. This is achieved by having the following properties set to true:

   - `property=policy.fence.masterReplicator=true` [443] tells the manager to set the associated primary or relay datasource to the failed state if the replicator has a fault. This will cause a failover if the cluster is in *`AUTOMATIC`* mode.
   - `property=policy.fence.masterReplicator.threshold=6` [443] tells the manager to do six iterations to try to recover the replicator, one iteration every 10 seconds, after which the fencing occurs.

3. The other difference, in both rules, is that the rules that call this method ensure that there's at least one potential target replica for the failover. So some of the validation that is done in the manual case, inside of the method, is actually done in the body of the rules. The reason for this is that we would prefer not to initiate a failover if it doesn't have a chance to succeed.

8.4.2. Switch and Failover Steps for Local Clusters

The set of steps described below apply to both manually-initiated commands as well as those that are initiated by the Tungsten manager rules as a part of an automated recovery scenario.

Important

A failure of any of the below steps will result in a rollback of the operation to the starting state.

Once either the switch or failover operation for a local cluster is triggered, the following steps are taken:

1. FAILOVER ONLY: The current primary will be marked as FAILED. Tungsten will not allow any connections to a failed primary. Applications will appear to hang.
2. FAILOVER ONLY: Put the cluster into *MAINTENANCE* mode. *AUTOMATIC* policy mode will be restored after the failover operation is complete or if the operation is rolled back.

   SWITCH ONLY: Get the current policy mode and then put the cluster into *MAINTENANCE* mode if it isn't already. The current policy mode will be restored after the switch operation is complete or if the operation is rolled back.
3. Verify that the manager running on target is operational.
4. Verify that the replicator on the target is in the online state.
5. SWITCH ONLY: Verify that the manager on the source is operational.
6. SWITCH ONLY: Verify that the replicator on the source is in the online state.
7. SWITCH ONLY: Set the source datasource to the offline state. This operation does the following steps, and by the time it is done, all managers and connectors have an updated copy of the primary/relay datasource which shows its state as offline.

   - Update the on-disk datasource properties files that are stored in `cluster-home/conf/cluster/<cluster-name>/datasource/<datasource-name>`, simultaneously, on all managers.
   - Update the same datasource on all connectors, simultaneously, by calling the router gateway within each manager. Note that this call is synchronous and will not return until ALL connectors have suspended all new requests to connect to the datasource and have closed all active connections for the datasource. For this reason, the completion of the datasource offline call can be delayed depending on how long the connectors will wait before closing active connections.
8. SWITCH ONLY: At this point no more connections to the primary are possible. Applications will appear to hang if Connectors are running in Proxy mode.
9. Set the target datasource to the offline state. This performs the same set of steps as for the source datasource, above.

At this point in the switch operation, we have both the source and the target datasource in the offline state, there should be no active connections to the underlying database servers for these datasources, and no new connections are being allowed to either datasource. At the connector level, if an application requests a new connection to the primary or to a replica, the call will hang until the switch operation is complete. NOTE: the only replica that is put into the offline state is the one that is the target. Any remaining replicas remain available to handle read operations although, because the primary datasource is offline, replication will start to lag.

Now we're going to do the set of steps required to make sure that the target database server has all of the transactions that have been written to the current source database server.

1. SWITCH ONLY: Perform a replicator purge operation on the source. This operation identifies any database server threads that may still be running on the source database server even if the application level connections have been closed and kills those threads. Experience shows that this can sometimes be the case and if these threads were not killed transactions would 'leak' through to the current source and could be lost.
2. SWITCH ONLY: Flush all transactions from the source to the target. This involves doing a replicator flush operation, which will return the sequence number of the flush transaction and then polling with the replicator for the sequence number returned by the flush. This call with block indefinitely, waiting for the flush sequence number.
3. Put the replicator for the target into the offline state.
4. SWITCH ONLY: Put the replicator for the source into the offline state.
5. SWITCH ONLY: Set the source datasource to the replica role.
6. Set the target datasource to the primary or relay role.
7. Set the target replicator to the primary or relay role.
8. Set the source replicator to the replica role.
9. FAILOVER ONLY: Set the source datasource to the shunned state.

10. Put the replicator for the target into the online state.
11. Put the datasource for the target into the online state.
12. At this point, the new primary (or relay) is available to applications.
13. SWITCH ONLY: Put the source datasource, which is now a replica, into the online state.
14. Iterate through any remaining replicas in the cluster, if any, and reconfigure them to point at the new primary/relay.

    NOTE: This step also includes reconfiguring a relay replicator on another site if the service on which the switch or failover occurred is a primary service of a composite cluster.
15. Issue a replicator heartbeat. This writes a transaction on the primary and it will propagate to all replicas. We do not wait for this heartbeat event to propagate.

8.4.3. Switch and Failover Steps for Composite Services

Switch and Failover operations for composite services are handled by a different set of high-level methods than local (physical) dataservice.

Here are the steps taken, in the exact order taken, to execute a switch or failover operation in a composite service:

1. SWITCH ONLY: make sure that there is an online composite primary.
2. FAILOVER ONLY: identify the failed primary. Tungsten will not allow any connections to a failed primary. Applications will appear to hang.
3. If a target datasource is passed in, ensure that it exists and that it is not the current primary.
4. If a target is not passed in, evaluate each composite replica and, for each of those composite replicas, evaluate the relay in the physical service for the composite replica and find the relay with the highest stored sequence number.
5. Get the current policy mode and store it away. Then set the policy mode for the composite service to *MAINTENANCE*. This means that all physical services will also be in *MAINTENANCE* mode.
6. SWITCH ONLY: put the composite primary datasource into the offline state. This has the effect of also putting the physical service's primary datasource into the offline state.
7. SWITCH ONLY: At this point no more connections to the composite/physical primary are possible. Applications will appear to hang.
8. FAILOVER ONLY: shun the composite primary datasource. If the physical service for the failed composite primary is available, this will have the effect of shunning the physical primary datasource as well.
9. Put the composite target datasource into the offline state. This has the effect of also putting the physical relay on the target site into the offline state.
10. Purge transactions on the current physical primary if it's available.
11. SWITCH ONLY: Do a flush operation, as in the physical service switch operation, and wait until the flushed transaction is present on the target relay replicator.
12. SWITCH ONLY: At this point all transactions that are available from the failed primary will now be committed on the target.
13. SWITCH ONLY: Put the replicator for the physical source offline.
14. FAILOVER ONLY: When we have a multi-site composite service, there may be a relay in the service that successfully replicated more transactions than the relay that has been chosen to be the new primary. In this case, we'll need to have the new primary catch up from that relay or else when the system goes online, the further ahead relay will fail to go online because it will have a seqno higher than the new primary. So check whether or not we have such a case and catch up from an 'intermediate' replicator if necessary.
15. FAILOVER ONLY: AT THIS POINT ALL TRANSACTIONS THAT ARE AVAILABLE, FROM THE FAILED PRIMARY, WILL NOW BE COMMITTED ON THE TARGET.
16. Put the target composite datasource offline. This has the effect of putting the target's physical relay offline as well.
17. Put the replicator for the target's physical relay offline.
18. Change the role of the target's physical relay replicator to primary.
19. SWITCH ONLY: Change the role of the source's physical primary datasource to relay.

20. Change the role of the target's physical service relay datasource to primary.
21. Put the replicator for the new physical primary online.
22. Put the datasource for the new physical primary online.
23. SWITCH ONLY: put the replicator for the previous primary datasource into the online state as a relay replicator.
24. SWITCH ONLY: put the source physical primary datasource into the offline state.
25. SWITCH ONLY: set the source physical relay datasource as a primary.
26. SWITCH ONLY: put the new physical relay datasource from the source into the online state.
27. SWITCH ONLY: set the source composite datasource role from composite primary to composite replica.
28. Put the target composite datasource offline.
29. Set the target composite datasource from a composite replica to a composite primary.
30. Put the target composite datasource online. This has the effect of also putting the new physical primary into the online state.
31. At this point the composite and physical primary are online and connection requests will be accepted.
32. SWITCH ONLY: put the source composite datasource, which is now a replica, into the online state. This has the effect of setting the physical relay datasource into the online state as well.
33. Reconfigure all of the remaining composite replicas to point at the new primary. This process involves the following steps:
    a. Identify the physical relay for each replica's physical service.
    b. Put the relay replicator offline.
    c. Set the role of the replicator to relay.
    d. Put the relay replicator into the online state.
34. Take the saved policy manager mode and, if it's not `MAINTENANCE`, put the composite cluster into that saved mode.

8.5. Tungsten Manager Fault Detection, Fencing and Recovery

The information contained in this topic is meant to give a relatively comprehensive understanding of how the Tungsten Manager detects faults and the subsequent processing that leads to fencing the fault and possible recovery from faults. The main focus of this topic is the set of business rules, implemented in the Tungsten Manager, which, collectively, perform fault detection, fault fencing, and fault recovery.

8.5.1. Tungsten Manager Definitions

These definitions assume a familiarity with concepts like failover, switch, Primary and Replica datasource etc.

- coordinator — every cluster designates one of the Tungsten managers in the cluster as the coordinator and it is this manager that will be responsible for taking action, if action is required, to recover the cluster's database resources to the most highly available state possible.
- rules — this term specifically refers to a set of 'business rules', implemented in a format required by the JBoss Drools rules engine, and which are used to perform fault detection, fencing and recovery for Tungsten Clustering.
- rule firing/triggering — refers to the action of a rule becoming active due to the coincidence of one or more conditions specified in the rule itself. For example, a rule that detects a potential missing cluster member, called a 'heartbeat gap detection' rule, can fire if there are no other active alarms and if the rule has not 'seen' a cluster member heartbeat within the last 30-45 seconds. If a rule 'fires', further processing, specified in Java, will take effect.
- fault/fault detection — any condition which, if left unresolved, could lead to a lack of availability of database resources or to data inconsistency etc. Faults are detected. An example of a fault detection is to detect that a specific database server has stopped.
- [fault] alarm — the Tungsten Manager uses raises and processes entities that we will refer to as 'alarms' as an initial part of fault detection. A set of manager rules raise alarms under specific circumstances and that alarm stays active or triggers further processing depending on other rules. An alarm, depending on its type, may not necessarily mean that a fault has been definitively detected but that something has been detected that may or may not lead to an actual fault condition. An example of this, as you will see later, is a `HeartbeatGapAlarm` which occurs when the rules on a specific manager detect that a heartbeat event has not been received from one or more of the other managers in the group.

- (fault) alarm retraction — the action taken, by the rules, to remove an alarm from consideration for further action. For example, if the manager raises a 'Heartbeat Gap Alarm' and then, subsequently, detects that the heartbeat from the errant member has resumed, a rule will retract that alarm.
- (fault) fence/fencing — the action which leads to a first-level amelioration of a fault i.e. an action that keeps a fault from causing further harm - applications are isolated from the fault through the action of fencing. Fencing a fault results in the removal of the original fault condition but may also result in further recovery actions. An example of fencing the fault that is detected when a database server has stopped is to set the associated datasource to the FAILED state. This effectively makes the datasource unavailable to applications, immediately, thus isolating the application from the fault state.
- (fault) recover/recovery — the action which may occur after a fault is initially fenced and which leads to a condition of continued availability, data consistency etc. An example of a recovery operation is, for example, the failover that occurs after a stopped Primary database fault has been fenced.
- split-brain — a condition of a cluster such that members of the cluster group have different views of the same set of database resources and, in the most damaging incidence of split-brain, more than different cluster members designate different database resources as the `Primary` resource, resulting in applications being able, for example, to perform database updates on a database resource that should be a Replica. This condition is to be avoided at all costs since the result is data loss or data corruption.

8.5.2. Cluster Monitoring and Notification Events

The key fact that must be understood in order that the operation of the rules can be grasped is that the rules are primarily driven by notification events which are, in turn, generated by the cluster monitoring subsystem or, in some cases, by threads internal to the manager that act as monitors for specific conditions.

As a general rule, notifications are generated in the monitoring subsystem, on each host, independently of the other hosts, and sent, by each manager, to all of the managers in the group, including itself, via group communications and the notifications are then routed to the rules engine in each manager.

These notifications are sent via a messaging protocol that guarantees that the monitoring events are received by all managers in the group in exactly the same order. This ordering is a critical part of correct rules operation for if notifications are received in a different order and used to drive the rules evaluation, the rules on each manager could arrive at a different state/conclusion based on the ordering.

The cluster monitoring subsystem driven by a set of checker threads that are configured from files found in tungsten-manager/conf and are named `checker*.properties`. There are currently four of these checkers, with their corresponding checker configurations stored in the following `tungsten-manager/conf/checker*.properties` files:

- `checker.heartbeat.properties` - runs a thread that generates a manager heartbeat notification for the manager. If a manager goes away from the group, either by crashing, being stopped, or having a network outage, other managers won't 'see' that manager's heartbeat and will use that as a clue that there may be something amiss on that manager's host.
- `checker.instrumentation.properties` - TBD
- `checker.mysqlserver.properties` - polls the local mysql server for liveness/state. This checker attempts to establish a connection with the local MySQL server and to execute the query that is found in `tungsten-manager/conf/mysql_checker_query.sql`. The checker then evaluates the success or failure of the connect attempt and subsequent query execution and establishes the state of the MySQL server based on that evaluation. Because database server state is so critical to the operation of the cluster, particularly when it comes to availability of database resources, this particular checker uses a very fine-grained and configurable process for evaluating the state.
- `checker.tungstenreplicator.properties` - polls the local Tungsten replicator for liveness/state. This checker connects to the local replicator via the JMX interfaces and queries the replicator for its current state. If, in the process of connecting to the replicator, a connection cannot be established because the replicator is not running, the checker simply returns the STOPPED state.

Let's look at the contents of one of these files - `checker.mysqlserver.properties`:

```
####################################
# CHECKER.MYSQLSERVER.PROPERTIES #
####################################
requiresProxy=true
name=mysql_response
class=com.continuent.tungsten.monitor.checkers.JDBCMySQLDatabaseServerChecker
# delay between each monitoring run - default 3000ms
frequency=3000
# connection will be renewed after this period
reconnectAfter=30000
serverName=viveka
host=viveka
vendor=mysql
port=3306
driver=org.drizzle.jdbc.DrizzleDriver
url=jdbc:mysql:thin://viveka:3306/tungsten
```

```
username=tungsten
password=passw0rd
query=select 1
queryTimeout=5
queryFileName=mysql_checker_query.sql
```

The key thing to understand about this configuration file is that the value for the class property indicates which Java class the manager will load and that class, when alive, will then be configured with the additional properties that can be seen in this file.

Every checker will have a property frequency which indicates how often the checker thread will become active.

Then, since this checker is for MySQL server, you can see that the configuration file indicates which jdbc driver to use, which port to use to connect directly to MySQL, username, password etc. for the connection as well as the name of a file that determines what SQL the checker will run to check for the MySQL server liveness.

8.5.3. Rule Organization - Detection, Investigation, Fencing, Recovery

The main focus of manager operations are fault processing business rules, and since these rules are organized into categories based on their function. As you can infer from some of the previous definitions, there are four major categories of rules, with some ancillary 'housekeeping' categories. The four major categories are:

1. Fault Detection — raise alarms for specific or nascent faults.

   The text show below comes directly from the source code for the manager rules and comprise the major faults detected by the manager:

   - 0600: DETECT MEMBER HEARTBEAT GAP"

     This rule fires if there are not already other alarm types pending, for a specific member, and if at least 30-45 seconds has elapsed since the last time a given manager send a ClusterMemberHeartbeat event to the group of managers. The result of this rule firing will be that a MemberHeartbeatGapAlarm is raised as well as a `MembershipInvalidAlarm`. Both of these alarms trigger other rules, explained later, which do further investigations of the current cluster connectivity and membership.

   - 0601: DETECT STOPPED CLUSTER MANAGER"

     This rule fires after a `MemberHeartbeatGapAlarm` has been raised since the reason for a heartbeat gap can be that a manager has stopped. Further processing determines whether or not the manager is, indeed, stopped. This rule generates a `ManagerStoppedAlarm` if it determines that the manager in question is, in fact, stopped.

   - 0602: DETECT DATASERVER FAULT"

     This rule fires whenever monitoring detects that a given database server is not in the ONLINE state. The result of this rule firing is a `DataServerFaultAlarm` which, in turn, results in further investigation via one of the investigation rules described later.

   - 0604: DETECT UNREACHABLE REMOTE SERVICE"

     In the case of a composite cluster, the current coordinator on each site will connect to one manager on the remote site. After establishing this connection, the local manager will poll the remote manager for liveness and generate a `RemoteServiceHeartbeatNotification` that has a status of REACHABLE. If the remote manager is not reachable, the manager that is polling the remote service will generate a `RemoteServiceHeartbeatNotification` that has a resource state of UNREACHABLE in which case this rule triggers and a `RemoteDataServiceUnreachableAlarm` is raised by this rule.

   - 0606: DETECT REPLICATOR FAULT"

     As the name of this rule indicates, it will detect a replicator fault. This rule triggers if it sees a replicator notification that indicates that any of the replicators in the cluster are in a STOPPED, SUSPECT or OFFLINE state. The result of triggering of this rule is at a `ReplicatorFaultAlarm` is raised.

2. Fault Investigation - process alarms by iteratively investigating whether or not the alarm represents a true fault that requires further action.

   - 0525: INVESTIGATE MY LIVENESS"

     In this rule title the word MY refers to the manager that is currently evaluating the rule. This rule triggers when it sees a `MemberHeartbeatGapAlarm` and, as a result of triggering, the manager checks to see if it has both network connectivity as well as visibility of the other cluster members including, if necessary, visibility of a passive witness host. If, during this connectivity check, a manager determines that it is isolated from the rest of the cluster, it will restart itself in a failsafe mode meaning that if will shun all of its database resources an then attempt to join an existing cluster group as a part of a quorum. This rule is particularly important in cases where there are transient or even protracted network outages since it forms a part of the strategy used to avoid split-brain operations. If the manager, after restarting, is able to become part of a cluster quorum group, the process of joining that group will result in shunned resources becoming available again if appropriate.

- 0530: INVESTIGATE MEMBERSHIP VALIDITY"

  This rule is triggered when it sees a MembershipInvalidAlarm which was previously generated as the result of a member heartbeat gap. The previous rule, INVESTIGATE MY LIVENESS

  checks for network connectivity for the current manager. This rule checks to see if the current manager is a part of a cluster quorum group. This type of check implies a connectivity check as well but goes further to see if other managers in the group are alive and operational. This rule is another critical part of split brain avoidance since, depending on what it determines, it will take one of the following actions:

  - If the manager does not have network connectivity after checking, every 10 seconds, for a period of 60 seconds, it will restart itself in two different modes:

    a. If the manager detects that it is the last man standing

       , meaning that it is currently responsible for the Primary datasource and all of the other cluster members had previously stopped, it will restart normally, leaving the Primary datasource available, and will be prepared to be the leader of any new group of managers.

    b. If the manager is not the last man standing

       , it will restart in failsafe mode i.e. will restart with all of its resources shunned and will attempt to join an existing group.

  - If the manager has network connectivity i.e. can see all of the other hosts in the cluster, it then checks to see if it is a part of a primary partition i.e. a cluster quorum group. If, the first time it checks, it determines that it is not a part of a primary partition, it immediately disconnects all existing Tungsten connector connections from itself. This has the effect, on the Tungsten connector side, of immediately suspending all new database connection requests until such time that the manager determines that it is in a primary partition

    . This is, again, a critical part of avoiding split-brain operation since it makes it impossible for connectors to satisfy new connection requests until a valid cluster quorum can be definitively validated.

  - The manager will then keep doing this check for quorum group membership, every 10 seconds, for a period of 60 seconds. If it determines that it is not a member of a quorum group, it will use the same criteria, as mentioned previously in the network connectivity case, to determine how it shall restart i.e.:

    a. If it is the last man standing

       it will, as in the above case, restart normally, leaving the Primary datasource available.

    b. If the manager is not the last man standing

       , it will restart in failsafe mode i.e. will restart with all of its resources shunned and will attempt to join an existing group.

  - If, after all of the previous checks, the manager establishes that it is a part of a quorum group, it will, if necessary because it disconnected Tungsten connectors during quorum validation, it will become available for Tungsten connectors to connect to it again after synchronizing its view of the cluster with the current cluster coordinator, and will then continue normal operations.

- 0550: INVESTIGATE: TIME KEEPER FOR HEARBEAT GAP ALARM"
- 0550: INVESTIGATE: TIME KEEPER FOR INVALID MEMBERSHIP ALARM"
- 0550: INVESTIGATE: TIME KEEPER FOR MANAGER STOPPED ALARM"
- 0550: INVESTIGATE: TIME KEEPER FOR DATASERVER STOPPED ALARM"
- 0551: INVESTIGATE: TIME KEEPER FOR REMOTE SERVICE STOPPED ALARM"
- 0552: INVESTIGATE: TIME KEEPER FOR REPLICATOR FAULT ALARM"

3. Fault Fencing - fences validated faults, rendering them less disruptive/harmful from the standpoint of the application.
   - 0303: FENCE FAILED NODE"
   - 0304: FENCE FAULTED DATASERVER"

- 0305: FENCE UNREACHABLE REMOTE SERVICE"
- 0306: FENCE REPLICATOR FAULT - DIMINISHED DATASOURCE"
- 0306: FENCE REPLICATOR FAULT - EXPIRED ALARM"

4. Fault Recovery - attempts to render the fault completely harmless by taking some action that either corrects the fault or by providing alternative resources to manage the fault.
   - 0200a: RECOVER MASTER DATASOURCE BY FAILING OVER - NON-REPLICATOR FAULT"
   - 0200b: RECOVER MASTER DATASOURCE BY FAILING OVER - REPLICATOR FAULT"
   - 0201: RECOVER COMPOSITE DATASOURCES TO ONLINE"
   - 0201a: RECOVER FAILSAFE PHYSICAL SLAVE WITH ONLINE PRIMARY"
   - 0201: RECOVER FAILSAFE SHUNNED COMPOSITE SLAVE TO ONLINE"
   - 0202: RECOVER OFFLINE PHYSICAL DATASOURCES TO ONLINE"
   - 0203: RECOVER FAILED PHYSICAL DATASOURCES TO ONLINE"
   - 0204: RECOVER MASTER REPLICATORS TO ONLINE"
   - 0205: RECOVER SLAVE REPLICATORS TO ONLINE"
   - 0206: RECOVER FROM DIMINISHED STATE WHEN STOPPED REPLICATOR RESTARTS"
   - 0207: RECOVER MERGED MEMBERS"
   - 0208: RECOVER AND RECONCILE REMOTE DATA SERVICE STATE"
   - 0209: PREVENT MULTIPLE ONLINE MASTERS"
   - 0210: RECOVER WITNESSES TO ONLINE"
   - 0211: RECOVER REMOTE FAILSAFE SHUNNED COMPOSITE MASTER TO ONLINE"
   - 0212: RECOVER NON-READ-ONLY SLAVES TO READ-ONLY"

8.6. Cluster State Savepoints

The Savepoint functionality was created to support clean, consistent rollbacks during aborted switch and failover operations. This functionality works for both physical clusters as well as for composite clusters.

Savepoints are created automatically with every switch and failover command. The savepoint is only used if there is an exception during switch or failover that is actually able to be rolled-back.

Below is an example of the output:

```
[LOGICAL] /alpha > switch
SELECTED SLAVE: db3@alpha
SET POLICY: AUTOMATIC => MAINTENANCE
Savepoint switch_2(cluster=alpha, source=db1, created=2019/06/29 13:51:56 BST) created
```

...

Not all exceptions during switch and failover will cause a rollback.

In particular, if an exception happens during switch or failover AFTER a new primary datasource has been put online (relay or Primary) then the switch or failover operation cannot be rolled back.

The Manager is configured, by default, to hold a maximum of 50 savepoints. When that limit is hit, the Manager resets the current-savepoint-id to 0 and starts to overwrite existing savepoints, starting at 0.

Chapter 9. Command-line Tools

Tungsten Cluster 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 Cluster 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
- tapi — provides easy access to API endpoints

Commands related to managing Tungsten Cluster

- cctrl — cluster control
- cluster_backup—cluster backup automation

Commands related to the core Tungsten Replicator

- trepctl — replicator control
- multi_trepctl — multi-replicator control
- thl — examine Tungsten History Log contents

Commands related to managing Tungsten Replicator deployments

- tprovision — provision or reprovision a Replica from an existing Primary or Replica database
- tungsten_read_master_events — read Primary events to determine the correct log position

Commands related to monitoring

- tmonitor — management and testing of external Prometheus exporters. Introduced from v6.1.4.
- tungsten_monitor — provides a mechanism for monitoring the cluster state
- tungsten_send_diag — assists with diag and file uploads to Continuent support
- tungsten_prep_upgrade — assists with upgrades from one topology to another

9.1. The cctrl Command

The cctrl command provides cluster management for your installed cluster, providing a command-line shell interface to obtain information and manage your cluster and structure.

9.1.1. cctrl Command-line Options

`cctrl [ -admin ] [ -expert ] [ -host ] [ -logical ] [ -multi ] [ -no-history ] [ -physical ] [ -port ] [ -proxy ] [ -service ] [ -timeout ]`

Where:

– `-admin` [310]

Table 9.1. cctrl Command-line Options

Option	`-admin` [310]
Description	Enter admin mode when connecting
Value Type	string

Automatically enters `admin` mode when cctrl connects to the cluster:

```
shell> cctrl -admin
Tungsten Clustering 7.0.3 build 141
alpha: session established, encryption=false, authentication=false
[ADMIN] /alpha >
```

– -expert [311]

Table 9.2. cctrl Command-line Options

Option	-expert [311]
Description	Enter expert mode when connecting
Value Type	string

Automatically enters expert mode when cctrl connects to the cluster:

```
shell> cctrl -expert
Tungsten Clustering 7.0.3 build 141
alpha: session established, encryption=false, authentication=false
LOGICAL:EXPERT] /alpha >
```

– -host [311]

Table 9.3. cctrl Command-line Options

Option	-host [311]
Description	Host name of the service manager to use
Value Type	string
Default	localhost

Allows you to specify the host to connect to when looking for a manager. By default, cctrl will connect to the local host:

```
shell> cctrl -host host1
Tungsten Clustering 7.0.3 build 141
alpha: session established, encryption=false, authentication=false
[LOGICAL] /alpha >
```

– -logical [311]

Table 9.4. cctrl Command-line Options

Option	-logical [311]
Description	Enter logical mode when connecting
Value Type	string

Automatically enters logical mode when cctrl connects to the cluster. This mode is the default when connecting to a typical; using this option forces this mode:

```
shell> cctrl -expert
Tungsten Clustering 7.0.3 build 141
alpha: session established, encryption=false, authentication=false
LOGICAL:EXPERT] /alpha >
```

– -multi [311]

Table 9.5. cctrl Command-line Options

Option	-multi [311]
Description	Allow support for connecting to multiple services
Value Type	string

By default, cctrl connects to the first local standalone service within the configuration. Using the -multi [311] option, cctrl will connect you at the root level.

For example, without -multi [311] option:

```
shell> cctrl
Tungsten Clustering 7.0.3 build 141
alpha: session established, encryption=false, authentication=false
[LOGICAL] /alpha >
```

With the -multi [311] option:

```
shell> cctrl -multi
Tungsten Clustering 7.0.3 build 141
```

```
alpha: session established, encryption=false, authentication=false
[LOGICAL] / >
```

– -no-history [312]

Table 9.6. cctrl Command-line Options

Option	-no-history [312]
Description	Disable command history
Value Type	string

Prevents cctrl from accessing or recording command history during interaction.

– -physical [312]

Table 9.7. cctrl Command-line Options

Option	-physical [312]
Description	Enter physical mode when connecting
Value Type	string

Automatically enters logical mode when cctrl connects to the cluster. This mode is the default when connecting to a typical topology; using this option forces this mode:

```
shell> cctrl -physical
Tungsten Clustering 7.0.3 build 141
alpha: session established, encryption=false, authentication=false
[PHYSICAL] resource://>
```

– -port [312]

Table 9.8. cctrl Command-line Options

Option	-port [312]
Description	Specify the TCP/IP port of the service manager
Value Type	string
Default	9997

Specify the TCP/IP port of the service manager

– -proxy [312]

Table 9.9. cctrl Command-line Options

Option	-proxy [312]
Description	Operate as a proxy service
Value Type	string

Operate as a proxy service

– -service [312]

Table 9.10. cctrl Command-line Options

Option	-service [312]
Description	Connect to a specific service
Value Type	string

Enables the selection of a specific service when first connecting to the cluster. For example:

```
shell > cctrl -service east_from_west
Tungsten Clustering 7.0.3 build 141
east: session established, encryption=false, authentication=false
east_from_west: session established, encryption=false, authentication=false
[LOGICAL] /east_from_west >
```

– -timeout [312]

Table 9.11. cctrl Command-line Options

Option	-timeout [312]
Description	Specify timeout (in seconds) to determine how long to wait before timing out when unable to connect to the manager. Default 30 seconds
Value Type	string

9.1.2. cctrl Modes

Four different modes are available within cctrl.

The default mode is LOGICAL.

- ADMIN Mode
- EXPERT Mode
- LOGICAL Mode
- PHYSICAL Mode

You can specify the mode to enter from the command-line, using the appropriate switch. For example, to start cctrl in EXPERT mode:

```
shell> cctrl -expert
```

You can also change the mode from within cctrl by issuing the appropriate command. For example, to switch to EXPERT mode:

```
[LOGICAL] /alpha > expert

WARNING: This is an expert-level command:
Incorrect use may cause data corruption
or make the cluster unavailable.

Do you want to continue? (y/n)> y
[LOGICAL:EXPERT] /alpha >
```

The current mode is always displayed as part of the command prompt within cctrl.

9.1.3. cctrl Commands

Table 9.12. cctrl Commands

Option	Description
admin	Change to admin mode
cd	Change to a specific site within a multisite service
cluster	Issue a command across the entire cluster
cluster topology validate	Check, validate and report on current cluster topology and health.
cluster validate	Validate the cluster quorum configuration
create composite	Create a composite dataservice
datasource	Issue a command on a single datasource
expert	Change to expert mode
failover	Perform a failover operation from a primary to a replica
help	Display the help information
ls	Show cluster status
members	List the managers of the dataservice
physical	Enter physical mode
ping	Test host availability
quit, exit	Exit cctrl
recover master using	Recover the Primary within a datasource using the specified Primary
replicator	Issue a command on a specific replicator
router	Issue a command on a specific router (connector)

Option	Description
service	Run a service script
set	Set management options
set master	Set the Primary within a datasource
show topology	Shows the currently configured cluster topology
switch	Promote a Replica to a Primary

9.1.3.1. admin Command

The admin command enables admin mode commands and displays. Admin mode is a specialized mode used to examine and repair cluster metadata. It is not recommended for normal use unless advised by Continuent Support.

9.1.3.2. cd Command

> **Note**
>
> The cd command only works when you have connected to cctrl using the `-multi` [311]

The cd command changes the data service being administered. Subsequent commands will only affect the given data service name.

Using cd .. allows to go back to the root element. The given data service name can be either composite or physical

9.1.3.3. cluster Command

The cluster command operates at the level of the full cluster.

9.1.3.3.1. cluster check Command

The cluster check command issues an MD5 consistency check on one or more tables in a database on the Primary data source. The consistency checks then replicate to each Replica, whereupon the Applier replicator repeats the check.

If the check fails, Replicas may go offline or print a log warning depending on how the replicators are configured. The default is to go offline. You can return a replicator to the online state after a failed check by issuing a replicator online command.

The table name can also be a wildcard (*) in which case all tables will be checked. Users may optionally specify a range of rows to check using the -limit option, which takes a starting row option followed by a number of rows to check. Rows are selected in primary key order.

The following example checks all tables in database accounting.

```
[LOGICAL] /alpha > cluster check accounting.*
```

The following command checks only the first 10 rows in a single table.

```
[LOGICAL] /alpha > cluster check accounting.invoices -limit 1,10
```

> **Warning**
>
> Consistency checks can be very lengthy operations for large tables and will lock them while they run. On the Primary this can block applications. On Replicas it blocks replication.

9.1.3.3.2. cluster flush Command

The cluster flush command sends a heartbeat event through the local cluster and returns a flush sequence number that is guaranteed to be equal to or greater than the sequence number of the flush event. Replicas that reach the flush sequence number are guaranteed to have applied the flush event.

This command is commonly used for operations like switch that need to synchronize the position of one or more Primaries or Replicas.

9.1.3.3.3. cluster heartbeat Command

The cluster heartbeat command sends a heartbeat event through the local cluster to demonstrate that all replicators are working. You should see the sequence numbers on all data sources advance by at least 1 if it is successful.

9.1.3.3.4. cluster offline Command

The cluster offline command brings all data services that are not offline into the offline state. It has no effect on services that are already offline.

9.1.3.3.5. cluster online Command

The cluster online command brings all data services that are not online into the online state. It has no effect on services that are already online.

9.1.3.3.6. cluster validate Command

The cluster validate validates the configuration of the cluster with respect to the quorum used for decision making. The number of active managers and active witnesses within the cluster is validated to ensure that there are enough active hosts to make a decision in the event of a failover or other failure event.

When executed, the validation routine checks all the available hosts, including witness hosts, and determines whether there are enough hosts, and whether their membership of the cluster is valid. In the event of deficiency, corrective action will be recommended.

By default, the command checks all hosts within the configured cluster:

```
[LOGICAL] /alpha > cluster validate
HOST host1/192.168.2.20: ALIVE
HOST host2/192.168.2.21: ALIVE
HOST host3/192.168.2.22: ALIVE
CHECKING FOR QUORUM: MUST BE AT LEAST 2 MEMBERS, OR 1 MEMBERS PLUS ALL WITNESSES
QUORUM SET MEMBERS ARE: host2, host1, host3
SIMPLE MAJORITY SIZE: 2
VALIDATED MEMBERS ARE: host2, host1, host3
REACHABLE MEMBERS ARE: host2, host1, host3
WITNESS HOSTS ARE:
REACHABLE WITNESSES ARE:
MEMBERSHIP IS VALID
GC VIEW OF CURRENT MEMBERS IS: host1, host2, host3
VALIDATED CURRENT MEMBERS ARE: host2, host1, host3
CONCLUSION: I AM IN A PRIMARY PARTITION OF 3 MEMBERS OUT OF THE REQUIRED MAJORITY OF 2
VALIDATION STATUS=VALID CLUSTER
ACTION=NONE
```

Additionally, a list of hosts to exclude from the check can be provided to verify the cluster capability when certain hosts have already failed or been shunned from the dataservice during maintenance.

To exclude hosts, add `excluding` and a comma-separated list of hosts to the command. For example:

```
[LOGICAL] /alpha > cluster validate excluding host3,host2
EXCLUDED host3 FROM VIEW
EXCLUDED host2 FROM VIEW
HOST host1/192.168.2.20: ALIVE
CHECKING FOR QUORUM: MUST BE AT LEAST 2 MEMBERS, OR 1 MEMBERS PLUS ALL WITNESSES
QUORUM SET MEMBERS ARE: host2, host1, host3
SIMPLE MAJORITY SIZE: 2
VALIDATED MEMBERS ARE: host1
REACHABLE MEMBERS ARE: host1
WITNESS HOSTS ARE:
REACHABLE WITNESSES ARE:
MEMBERSHIP IS VALID
GC VIEW OF CURRENT MEMBERS IS: host1
VALIDATED CURRENT MEMBERS ARE: host1
CONCLUSION: I AM IN A NON-PRIMARY PARTITION OF 1 MEMBERS OUT OF A REQUIRED MAJORITY SIZE OF 2
AND THERE ARE 0 REACHABLE WITNESSES OUT OF 0
VALIDATION STATUS=NON-PRIMARY PARTITION
ACTION=RESTART SAFE
```

Cluster validation can be used to provide validation only. To improve the support:

- Add active witnesses to the dataservice, see Section 3.7.2, “Adding Active Witnesses to an Existing Deployment”
- Add Replica hosts to the dataservice, see Section 3.7.1, “Adding Datasources to an Existing Deployment”

9.1.3.3.7. cluster topology validate Command

The cluster topology validate will check and validate a cluster topology and, in the process, will report any issues that it finds. The purpose of this command is to provide a fast way to see, immediately, if there are any issues with any components of a cluster.

Here's an example of the command when run in the context of an Composite Active/Active cluster:

```
[LOGICAL] /usa > cluster topology validate
Validating physical cluster 'east' for composite datasource 'east@usa'
Validating datasource 'db4@east'
Validating datasource 'db5@east'
Validating datasource 'db6@east'
Validating physical subservice 'east_from_west' of service 'east'
```

```
Validating datasource 'db4@east_from_west'
Validating datasource 'db5@east_from_west'
Validating datasource 'db6@east_from_west'
Physical cluster 'east_from_west' is VALID
Physical cluster 'east' is VALID
Validating physical cluster 'west' for composite datasource 'west@usa'
Validating datasource 'db1@west'
Validating datasource 'db2@west'
Validating datasource 'db3@west'
Validating physical subservice 'west_from_east' of service 'west'
Validating datasource 'db1@west_from_east'
Validating datasource 'db2@west_from_east'
Validating datasource 'db3@west_from_east'
Physical cluster 'west_from_east' is VALID
Physical cluster 'west' is VALID
Composite cluster 'usa' is VALID
```

Here's an example of the command when run in the context of an Composite Active/Active cluster that does not validate due to underlying issues:

```
[LOGICAL] /usa > cluster topology validate
Validating physical cluster 'east' for composite datasource 'east@usa'
Validating datasource 'db4@east'
Validating datasource 'db5@east'
Validating datasource 'db6@east'
Validating physical subservice 'east_from_west' of service 'east'
Validating datasource 'db4@east_from_west'
Validating datasource 'db5@east_from_west'
Validating datasource 'db6@east_from_west'
Physical cluster 'east_from_west' is VALID
Physical cluster 'east' is VALID
Validating physical cluster 'west' for composite datasource 'west@usa'
Validating datasource 'db1@west'
Validating datasource 'db2@west'
INVALID: db2@west: Primary datasource is 'db1' but replicator points at 'db3'
Validating datasource 'db3@west'
Validating physical subservice 'west_from_east' of service 'west'
Validating datasource 'db1@west_from_east'
Validating datasource 'db2@west_from_east'
Validating datasource 'db3@west_from_east'
INVALID: db3@west_from_east: Primary datasource is 'db1' but replicator points at 'db2'
Physical cluster 'west_from_east' is INVALID
Physical cluster 'west' is INVALID
Composite cluster 'usa' is INVALID
```

In the above case you can see that there are two issues, shown with the word INVALID at the start of the line. In the cluster west, datasource db2's replicator points at db3 but should point at db1. In the cluster west_from_east, db3's replicator points at db2 but should point at db1.

9.1.3.4. create composite Command

The create composite command creates a new composite data source or data service with the given name. Composite data services can only be create in the root directory '/' while composite data sources need to be created from a composite data service location. Composite data source names should be the same as the physical data services Composite data service name should be named after its composite data sources

The following example creates a composite data service named 'sj_nyc'

```
cctrl> create composite dataservice sj_nyc
```

The following example changes to the composite data service sj_nyc, then creates a composite data source named 'sj' in this composite data service

```
cctrl> use sj_nyc
cctrl> create composite datasource sj
```

9.1.3.5. datasource Command

The datasource command affects a single data source.

Table 9.13. cctrl datasource Commands

Option	Description
backup	Backup a datasource
connections	Displays the current number of connections running to the given node through connectors.

Option	Description
drain	Prevents new connection to be made to the given data source, while ongoing connection remain untouched. If a timeout (in seconds) is given, ongoing connections will be severed only after the timeout expires.
fail	Fail a datasource
offline	Put a datasource into the offline state
online	Put a datasource into the online state
recover	Recover a datasource into operation state as Replica
restore	Restore a datasource from a previous backup
shun	Shun a datasource
welcome	Welcome a shunned datasource back to the cluster

`datasource` [host] [command]

For example:

```
datasource db1 shun
```

9.1.3.5.1. datasource backup Command

The datasource backup command invokes a backup on the data source on the named host using the default backup agent and storage agent. Backups taken in this way can be reloaded using the datasource restore command. The following command options are supported:

- *`backupAgent`* — The name of a backup agent.
- *`storageAgent`* — The name of a storage agent.
- *`timeout`* — Number of seconds to wait before the backup command times out.

On success the backup URL will be written to the console.

The following example performs a backup on host saturn using the default backup agent.

```
cctrl> datasource saturn backup
```

The following example performs a backup on host mercury using the xtrabackup agent, which is named explicitly.

```
cctrl> datasource mercury backup xtrabackup
```

9.1.3.5.2. datasource connections Command

> **Note**
>
> This feature is only available from release 7.0.2 onwards

The datasource connections command displays the current number of connections running to the given node through connectors

The optional `-l` flag will add the list of connectors and their number of connections to this node

Example:

```
[LOGICAL] /nyc > datasource db2 connections -l
15
connector@db3[16305] (10)
connector@db2[20304] (5)
```

9.1.3.5.3. datasource drain Command

> **Note**
>
> This feature is only available from release 7.0.2 onwards

The datasource drain command will prevent new connections to the specified data source, while ongoing connections remain untouched.

If a timeout (in seconds) is given, ongoing connections will be severed after the timeout expires.

This command returns immediately, no matter whether a timeout is given or not. The number of remaining connections can be displayed with the datasource connections command.

This command will typically be used for seamless node maintenance, including composite data source (whole site) maintenance.

Under the hood, this command will put the data source into SHUNNED state, with lastShunReason set to DRAIN-CONNECTIONS. Once the maintenance is finished and willing to re-allow connections to it, the command datasource welcome will bring the data source (and its underlying physical nodes if applicable) back to an "online" operational state

Example:

```
[LOGICAL] /nyc > datasource db2 drain
DataSource 'db2@nyc' set to SHUNNED

[LOGICAL] /nyc > ls
COORDINATOR[db1:AUTOMATIC:ONLINE]
ROUTERS:
+---------------------------------------------------------------------------+
|connector@db1[15937](ONLINE, created=58, active=10)                        |
|connector@db2[20304](ONLINE, created=29, active=5)                         |
|connector@db3[16305](ONLINE, created=20, active=10)                        |
|connector@db4[15065](ONLINE, created=0, active=0)                          |
|connector@db5[15172](ONLINE, created=0, active=0)                          |
|connector@db6[15261](ONLINE, created=0, active=0)                          |
+---------------------------------------------------------------------------+

DATASOURCES:
+---------------------------------------------------------------------------+
|db1(master:ONLINE, progress=189240, THL latency=0.492)                     |
|STATUS [OK] [2022/11/17 01:21:10 PM UTC]                                   |
+---------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                     |
| REPLICATOR(role=master, state=ONLINE)                                     |
| DATASERVER(state=ONLINE                                                   |
| CONNECTIONS(created=59, active=10)                                        |
+---------------------------------------------------------------------------+

+---------------------------------------------------------------------------+
|db2(slave:SHUNNED(DRAIN-CONNECTIONS), progress=102621, latency=172.826)    |
|STATUS [SHUNNED] [2022/11/17 02:12:59 PM UTC]                              |
+---------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                     |
| REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
| DATASERVER(state=ONLINE)                                                  |
| CONNECTIONS(created=6, active=0)                                          |
+---------------------------------------------------------------------------+

+---------------------------------------------------------------------------+
|db3(slave:ONLINE, progress=155889, latency=67.533)                         |
|STATUS [OK] [2022/11/17 01:21:39 PM UTC]                                   |
+---------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                     |
| REPLICATOR(role=slave, master=db1, state=ONLINE)                          |
| DATASERVER(state=ONLINE)                                                  |
| CONNECTIONS(created=42, active=10)                                        |
+---------------------------------------------------------------------------+
```

9.1.3.5.4. datasource fail Command

The datasource fail allows you to place a host into the failed state.

To place a node back into an online state, you must issue the datasource recover command.

If the cluster is in *AUTOMATIC* policy mode, the cluster will attempt to recover the host automatically if the replicator and database are online.

In order to maintain the failed state, switch the cluster to *MAINTENANCE* and/or stop the underlying database and replicator.

The following example changes the state of the node venus, to failed:

```
cctrl> datasource venus fail
```

9.1.3.5.5. datasource offline Command

The datasource offline allows you to place a host into an offline state. It has no effect if the datasource is already in an offline state.

To place a node back into an online state, you must issue the datasource online command.

If the cluster is in *AUTOMATIC* policy mode, the cluster will return the host to online automatically.

In order to maintain the offline state, switch the cluster to *MAINTENANCE* and/or stop the underlying database and replicator.

The following example changes the state of the node mercury, to failed:

```
cctrl> datasource mercury offline
```

9.1.3.5.6. datasource online Command

The datasource online allows you to place a host into an online state. It has no effect if the datasource is already in an online state.

To place a node back into an online state, you must issue the datasource online command.

If the node is in a SHUNNED or FAILED state, this command will fail with an error. Instead, you should use the datasource recover command

The following example changes the state of the node mercury, to failed:

```
cctrl> datasource mercury online
```

9.1.3.5.7. datasource recover Command

The datasource recover reconfigures a shunned data source and returns it to the cluster as a Replica. This command can be used with failed Primary as well as Replica data sources.

For Replica data sources, the recover command attempts to restart the DBMS server followed by replication. If successful, the data source joins the cluster as an online Replica.

For Primary data sources, the recover command first reconfigures the Primary as a Replica. It then performs the same recovery process as for a failed Replica.

If datasource recover is unsuccessful, the next step is typically to restore the data source from a backup. This should enable it to rejoin the cluster as a normal Replica.

The following example recovers host mercury following a failure. The command is identical for Primary and Replica data sources.

```
cctrl> datasource mercury recover
```

9.1.3.5.8. datasource restore Command

The datasource restore command reloads a backup generated with the datasource backup command.

The following command options are supported:

- *uri* — The URI of a specific backup to restore
- *timeout* — Number of seconds to wait before the command times out.

To restore a data source you must first put the data source and its associated replicator offline.

The following example restores host saturn from the latest backup. The preceding commands place the datasource and replicator offline. The commands after the restore return the datasource to the cluster and put it online.

```
cctrl> datasource saturn shun
cctrl> datasource saturn offline
cctrl> replicator saturn offline
cctrl> datasource saturn restore
cctrl> datasource saturn welcome
cctrl> cluster online
```

The following example restores host mercury from an existing backup, which is explicitly named. The datasource and replicator must be offline.

```
cctrl> datasource mercury restore storage://file-system/store-0000000004.properties
```

9.1.3.5.9. datasource shun Command

The datasource shun command removes the data source from the cluster and makes it unavailable to applications. It will remain in the shunned state without further changes until you issue a datasource welcome or datasource recover command.

The datasource shun command is most commonly used to perform maintenance on a data source. It allows you to reboot servers and replicators without triggering automated policy mode rules.

The following example shuns the data source on host venus.

```
cctrl> datasource venus shun
```

9.1.3.5.10. datasource welcome Command

When a datasource has been shunned, the datasource can be welcomed back to the dataservice by using the welcome command. The welcome command attempts to enable the datasource in the ONLINE state using the current roles and configuration. If the datasource was operating as a Replica before it was shunned, the welcome command will enable the datasource as a Replica.

For example, the host host3 is a Replica and currently online:

```
+----------------------------------------------------------------------------+
|host3(slave:ONLINE, progress=157454, latency=1.000)                         |
|STATUS [OK] [2013/05/14 05:12:52 PM BST]                                      |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                       |
|  REPLICATOR(role=slave, master=host2, state=ONLINE)                        |
|  DATASERVER(state=ONLINE)                                                    |
|  CONNECTIONS(created=0, active=0)                                            |
+----------------------------------------------------------------------------+
```

```
[LOGICAL:EXPERT] /alpha > datasource host3 shun
DataSource 'host3' set to SHUNNED
```

To switch the datasource back to the online state, the welcome is used:

```
[LOGICAL:EXPERT] /alpha > datasource host3 welcome
DataSource 'host3' is now OFFLINE
```

The welcome command puts the datasource into the OFFLINE state. If the dataservice policy mode is *AUTOMATIC*, the node will be placed into ONLINE mode due to automatic recovery. When in *MAINTENANCE* or *MANUAL* mode, the node must be manually set online.

The welcome command may not always work if there has been a failure or topology change between the moment it was shunned and welcomed back. Using the recover command may be a better alternative to using welcome when bringing a datasource back online. The recover commands ensures that the replicator, connector and operation of the datasource are correct within the current cluster configuration. See Section 9.1.3.14, “recover Command”.

9.1.3.6. expert Command

The expert command enables expert mode in cctrl. This suppresses prompts for commands that can cause damage to data. It is provided as a convenience for fast administration of the system.

> **Warning**
>
> This mode should be used with care, and only be used by experienced operators who fully understand the implications of the subsequent commands issued.
>
> Missuse of this feature may cause irreparable damage to a cluster

9.1.3.7. failover Command

The failover command performs a failover to promote an existing Replica to Primary after the current Primary has failed. The Primary data source must be in a failed state to use failover. If the Primary data source is not failed, you should instead use switch.

If there is no argument the failover command selects the most caught up Replica and promotes it as the Primary. You can also specify a particular host, in which case failover will ensure that the chosen Replica is fully up-to-date and promote it.

Failover ensures that the Replica has applied all transactions present in its log, then promotes the Replica to Primary. It does not attempt to retrieve transactions from the old Primary, as this is by definition already failed. After promoting the chosen Replica to Primary, failover reconfigures other Replicas to point to it and ensures all data sources are online.

To recover a failed Primary you should use the datasource recover command.

Failover to any up-to-date Replica in the cluster. If no Replica is available the operation fails:

```
cctrl> failover
```

Failover from a broken Primary to a specific node:

```
cctrl> failover to mercury
```

9.1.3.8. help Command

The help command provides help text from within the cctrl operation.

With no other arguments, help provides a list of the available commands:

```
[LOGICAL] /alpha > help

--------
Overview
--------
Description: Overview of Tungsten cctrl Commands

Commands
--------
admin                          - Enter admin mode
cd <name>                      - Change to the specified SOR cluster element
cluster <command>              - Issue a command on the entire cluster
create composite <type> <name> - Create SOR cluster components
datasource <host> <cmd>        - Issue a command on a datasource
expert                         - Enter expert mode
failover                       - Failover from failed &mas_lc; to &slv_lc;
help                           - Show help
ls [options]                   - Show generic cluster status
members                        - List all of the managers in the cluster
ping                           - Test host availability
physical                       - Enter physical mode
quit or exit                   - Leave cctrl
replicator <host> <cmd>        - Issue a command on a replicator
service                       - Run a service script
set                            - Set management options
switch                         - Promote a &slv_lc; to &mas_lc;

To get more information about particular commands type help followed by a
command.  Examples: 'help datasource' or 'help create composite'.
```

To get specific information about an individual command or operation, provide the command name to the help command. For example, to get information about the ping command, type help ping at the cctrl prompt.

9.1.3.9. ls Command

The ls command displays the current structure and status of the cluster.

`ls` [-l] [host] [[resources] | [services] | [sessions]]

The ls command operates in a number of different modes, according to the options provided on the command-line, as follows:

- No options

  Generates a list of the current routers, datasources, and the their current status and services.

- `-l`

  Outputs extended information about the current status and configuration. The -l option can be used in both the standard (no option) and host specific output formats to provide more detailed information.

- `host`

  You can also specify an individual component within the cluster on which to obtain information. For example, to get the information only for a single host, issue

  ```
  cctrl> ls host1
  ```

- `resources`

  The resources option generates a list of the configured resources and their current status.

- `services`

  The `services` option generates a list of the configured services known to the manager.

- `sessions`

  The `sessions` outputs statistics for the cluster. Statistics will only be presented when `SMARTSCALE` is enabled for the connectors

Without any further options, the output of ls looks similar to the following:

```
[LOGICAL] /alpha > ls

COORDINATOR[host1:AUTOMATIC:ONLINE]
```

```
ROUTERS:
+----------------------------------------------------------------------------+
|connector@host1[1179](ONLINE, created=0, active=0)                          |
|connector@host2[1532](ONLINE, created=0, active=0)                          |
|connector@host3[17665](ONLINE, created=0, active=0)                         |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|host1(master:ONLINE, progress=60, THL latency=0.498)                        |
|STATUS [OK] [2013/03/22 02:25:00 PM GMT]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+

+----------------------------------------------------------------------------+
|host2(slave:ONLINE, progress=31, latency=0.000)                             |
|STATUS [OK] [2013/03/22 02:25:00 PM GMT]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=host1, state=ONLINE)                        |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+

+----------------------------------------------------------------------------+
|host3(slave:ONLINE, progress=35, latency=9.455)                             |
|STATUS [OK] [2013/03/21 06:47:53 PM GMT]                                    |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=host1, state=ONLINE)                        |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
```

The purpose of the `Alert STATUS` field is to provide standard, datasource-state-specific values for ease of parsing and backwards-compatibility with older versions of the cctrl command.

The `STATUS` field is effectively the same information as the `DataSource State` that appears on the first line after the colon [:], just presented slightly differently.

Here are the possible values for `STATUS`, showing the `DataSource State` first, and the matching `Alert STATUS` second:

Datasource State	Alert STATUS
ONLINE	OK
OFFLINE	WARN (for non-composite datasources)
OFFLINE	DIMINISHED (for composite passive replica)
OFFLINE	CRITICAL (for composite active primary)
FAILED	CRITICAL
SHUNNED	SHUNNED

Any other `DataSource State` sets the `STATUS` to `UNKNOWN`.

9.1.3.10. members Command

The members command outputs a list of the currently identified managers within the dataservice.

For example:

```
[LOGICAL] /alpha > members
alpha/host1(ONLINE)/192.168.1.60:7800
alpha/host2(ONLINE)/192.168.1.61:7800
alpha/host3(ONLINE)/192.168.1.62:7800
```

The command outputs each identified manager service within the current dataservice.

The format of the output information is:

```
DATASERVICE/HOST(STATUS)/IPADDR:PORT
```

Where:

- DATASERVICE

  The name of the dataservice.

- HOST

  The name of the host on which the manager resides.

- STATUS

  The current status of the manager.

- IPADDR

  The IP address of the manager.

- PORT

  The primary TCP/IP port used for contacting the manager service.

The members service can be used as an indicator of the overall status of the dataservice. The information shown for each manager should within a single dataservice should be identical. If different information is shown, or an incomplete number of managers compared to the number of configured managers is provided, then it may indicate a communication or partition problem within the dataservice.

9.1.3.11. physical Command

The physical command enables physical mode commands and displays. This is a specialized mode used to examine interfaces of resources managed by the cluster. It is not recommended for normal administrative use.

9.1.3.12. ping Command

The ping command checks to see whether a host is alive. If the host name is omitted, it tests all hosts in the cluster including witness hosts.

Ping uses the host ping timeout and methods specified in the manager.properties file. By default output is parsimonious.

The following shows an example of the output:

```
[LOGICAL] /nyc > ping
NETWORK CONNECTIVITY: PING TIMEOUT=2
NETWORK CONNECTIVITY: CHECKING MY OWN ('db2') CONNECTIVITY
NETWORK CONNECTIVITY: CHECKING CLUSTER MEMBER 'db1'
NETWORK CONNECTIVITY: CHECKING CLUSTER MEMBER 'db3'
```

9.1.3.13. quit Command

Exits cctrl and returns the user to the shell.

9.1.3.14. recover Command

The recover will attempt to recover and bring online all nodes and services that are not in an ONLINE state.

Any previous failed Primary nodes will be reconfigured as Replicas, and all associated replicator services will be reconciled to connect to the correct Primary

If recovery is unsuccessful, the next step is typically to restore any failed data source from a backup.

9.1.3.15. replicator Command

The replicator command allows for control of the replicators, and include a number of uses.

Table 9.14. cctrl replicator Commands

Option	Description
master	Configure a replicator as a master.
offline	Set replicator offline.
online	Set replicaotr online.
relay	Set the replicator as a relay of the supplied service/host.

Option	Description
restart	Restart the replicator process.
slave	Configure replicator as a slave of the current master.
start	Start the replicator process if it is not running.
status	Show the current replicator status.
stop	Stop the replicator process if it is running.

`replicator` [host] [operation]

To perform operations safely on replicators you should either put the cluster in *MAINTENANCE* mode or shun the data source associated with the replicator.

AUTOMATIC policy mode may interfere with stop/offline operations.

> **Important**
>
> Commands controlling the replicator should be used with care and only be experienced operators, or under the guidance of Continuent Support
>
> Incorrect use of ths option could cause replicators to be misconfigured leading to unepxected replication errors, or even worse, data loss or corruption

9.1.3.16. router Command

The router command allows for control of the connectors.

The command can be used to take a connector offline or online and can be applied to a specific host or all hosts using the * wildcard, for example:

To take all connectors offline, issue:

```
[LOGICAL] /alpha > router * offline
```

To take a specific connector offline, issue:

```
[LOGICAL] /alpha > router connector@db1[5080] offline
```

Note, that in the above example, the host name needs to match the name as listed within the ls output.

Other options are:

- `online` : Bring the connector online is already offline
- `status` : Show the status of the connector including number of connections per connector, per node.

> **Note**
>
> Regardless of the cluster *AUTOMATIC* policy mode, an offline router will not automatically be bought back online.

9.1.3.17. service Command

The service command executes a command on the operating system according to standard Linux/Unix service script conventions. The service command may apply to a single host or may be executed on all hosts using the * operator. This latter form is also known as a broadcast command. You can enter service commands from any manager.

Commonly defined services include the following. User-defined services may also be invoked using the service command provided they are listed in the service configuration files for the cluster.

- connector: Tungsten Connector service
- mysql: MySQL service
- replicator: Tungsten Replicator service

The standard service commands are:

- restart: Stop and then start the service

- start: Start the service if it is not already running
- status: Show the current process status
- stop: Stop the service if it is running
- tail: Show the end of the process log (useful for diagnostics)

To start all mysqld processes in the cluster. This should be done in. *maintenance* mode to avoid triggering a failover.

```
cctrl> service */mysql restart
```

Stop the replicator process on host mercury.

```
cctrl> service mercury/replicator tail
```

Show the end of the log belonging to the connector process on host jupiter.

```
cctrl> service jupiter/connector tail
```

> **Warning**
>
> [Re-]starting Primary DBMS servers can cause failover when operating in *AUTOMATIC* policy mode. Always set policy mode to *MAINTENANCE* before restarting a Primary DBMS server.

9.1.3.18. set Command

The set command sets a management option. The following options are available.

- set policy - Set policy for cluster automation

  See Section 6.4.1, "Setting Policy Modes" for more details on using set policy

- set output - Set logging level in cctrl
- set force `>[true|false]`

> **Warning**
>
> Setting force should NOT be used unless advised by Support. Using this feature without care, can break the cluster and possibly cause data corruption.

9.1.3.19. show topology Command

The show topology command shows the topology for the currently selected cluster, or cluster composite.

For example, below is sample output for an Composite Active/Passive cluster:

```
[LOGICAL] /east > show topology
clustered_master_slave
```

For example, below is sample output for an Composite Active/Active cluster:

```
[LOGICAL] /usa > show topology
clustered_multi_master
```

When selecting a cluster within the composite:

```
[LOGICAL] /usa > use east
[LOGICAL] /east >  show topology
clustered_primary
```

The following values are output according to the cluster selected:

- Active service returns `clustered-primary`
- Passive service returns `clustered-sub-service`
- Composite Active/Passive returns `composite-master-slave`
- Composite Active/Active returns `composite-multi-master`

9.1.3.20. switch Command

The switch command performs a planned failover to promote an existing Replica to Primary and reconfigure the current Primary as a Replica.

The most common reason for a switch operation is to perform maintenance on the Primary.

If there is no argument the switch command selects the most caught up Replica and promotes it as the Primary. You can also specify a particular host, in which case switch will ensure that the chosen Replica is fully up-to-date and promote it.

Switch is a complex operation.

- First, we ensure that all transactions to the Primary through SQL router or connector processes complete before initiating the switch.
- It submits a flush transaction through the replicator to ensure that the chosen Replica is fully caught up with the Primary.
- It then reconfigures the Primary and Replica to reverse their roles.
- Finally, it puts the Primary and Replica back online.

In the event that switch does not complete, We attempt to revert to the old Primary. If a switch fails, you should check the cluster using 'ls' to ensure that things rolled back correctly.

Examples:

Switch to any up-to-date Replica in the cluster. If no Replica is available the operation fails.

```
cctrl> switch
```

Switch the Primary to host mercury.

```
cctrl> switch to mercury
```

The switch command can also be used to switch between the Active and Passive clusters in a Tungsten Cluster+ Active/Passive™ Topology.

In this scenario, the switch will promote the RELAY node in the remote cluster to be the Primary, and revert the Primary in the local cluster to be the Relay

To initiate the switch in a composite cluster, issue the command from the composite cluster, for example if you have cluster service east and west in a composite cluster called global, and east is the current Active site:

```
cctrl> use global
cctrl> switch
```

9.1.3.21. system summary Command

The system summary command outputs useful information and statistics regarding the manager on host on which the command was executed.

For example, if your cctrl was started on `db1` then the information shown would be related to that host only. An example of the kind of output displayed is as follows:

```
[LOGICAL] /west > system summary

+-------------------------------------------------------------------------------+
|db1 general info                                                               |
+-------------------------------------------------------------------------------+
|                  coordinator:    db3                                          |
|               dataServiceName:    west                                        |
|                dataSourceName:    db1                                         |
|                 isCoordinator:    false                                       |
|                     isWitness:    false                                       |
|          jgroupsUseEncryption:    true                                        |
|          jmxUseAuthentication:    true                                        |
|              jmxUseEncryption:    true                                        |
|keystoreLocation:    /opt/continuent/share/tungsten_keystore.jks               |
|                    managerPID:    7318                                        |
|                     parentPID:    7299                                        |
|                    policyMode:    AUTOMATIC                                   |
|                     startTime:    2025-02-12T13:39:53.098 UTC                 |
|                         state:    ONLINE                                      |
|trustoreLocation:    /opt/continuent/share/tungsten_truststore.ts              |
|                uptimeSeconds:    39                                          |
+-------------------------------------------------------------------------------+


+-------------------------------------------------------------------------------+
```

```
|db1 memory info                                                                |
+-------------------------------------------------------------------------------+
|                         free:    23553168                                     |
|                          max:    253427712                                    |
|                         peak:    59939696                                     |
|                        total:    83492864                                     |
|                         used:    59939696                                     |
+-------------------------------------------------------------------------------+


+-------------------------------------------------------------------------------+
|db1 thread info                                                                |
+-------------------------------------------------------------------------------+
|           daemon-thread-count:    41                                          |
|             peak-thread-count:    74                                          |
|                  thread-count:    67                                          |
|            total-cpu-percent:    16.809640522875817                          |
|            total-cpu-time-ms:    6584                                        |
|          total-cpu-time-nano:    6584797639                                  |
|    total-started-thread-count:    309                                        |
|           total-user-percent:    15.492238562091504                          |
|           total-user-time-ms:    6068                                        |
|         total-user-time-nano:    6068686286                                  |
|                       uptime:    39168                                       |
+-------------------------------------------------------------------------------+
```

9.2. The check_tungsten_latency Command

The check_tungsten_latency command reports warning or critical status information depending on whether the latency across the nodes in the cluster is above a specific level.

Table 9.15. check_tungsten_latency Options

Option	Description
-c	Report a critical status if the latency is above this level
--perslave-perfdata	Show the latency performance information on a per-Replica basis
--perfdata	Show the latency performance information
-w	Report a warning status if the latency is above this level

The command outputs information in the following format:

```
LEVEL: DETAIL
```

Where DETAIL includes detailed information about the status report, and LEVEL is:

- CRITICAL — latency on at least one node is above the specified threshold level for a critical report. The host reporting the high latency will be included in the DETAIL portion:

  For example:

  ```
  CRITICAL: host2=0.506s
  ```

- WARNING — latency on at least one node is above the specified threshold level for a warning report. The host reporting the high latency will be included in the DETAIL portion:

  For example:

  ```
  WARNING: host2=0.506s
  ```

- OK — status is OK; the highest reported latency will be included in the output.

  For example:

  ```
  OK: All slaves are running normally (max_latency=0.506)
  ```

The -w and -c options must be specified on the command line, and the critical figure must be higher than the warning figure. For example:

```
shell> check_tungsten_latency -w 0.1 -c 0.5
CRITICAL: host2=0.506s
```

Performance information can be included in the output to monitor the status. The format for the output is included in the DETAIL block and separates the maximum latency information for each node with a semicolon, and the detail block with a pipe symbol. For example:

```
shell> check_tungsten_latency -w 1 -c 1 --perfdata
```

```
OK: All slaves are running normally (max_latency=0.506) |  max_latency=0.506;1;1;;
```

Performance information for all the Replicas in the cluster can be output by using the `--perslave-perfdata` option which must be used in conjunction with the `--perfdata` option:

```
shell> check_tungsten_latency -w 0.2 -c 0.5 --perfdata --perslave-perfdata
CRITICAL: host2=0.506s | host1=0.0;0.2;0.5;; host2=0.506;0.2;0.5;;
```

9.3. The check_tungsten_online Command

The check_tungsten_online command checks whether all the services for a given service and host are online and running.

Within a Tungsten Cluster service, the replicator, manager and connector services are checked. All must be online for an OK response.

Table 9.16. check_tungsten_online Options

Option	Description
`-c`	Enable composite dataservice checks (default: disabled)
`-d`	Debug mode (enables verbose mode also)
`-h`	Display the help text
`-a`	Check manager services on all nodes, not just localhost.
`-port`	RMI port for the replicator being checked
`-r`	Disable replicator checks (default: enabled)
`-s`	Specify the service you would like to check (default: all available services)
`-v`	Verbose mode.

By default, the script will check all manager and replication services for the localhost

To also check the manager services on the other cluster nodes, use `-a`

You can also check the cluster-wide composite status using option `-c`

> **Warning**
>
> Using `-a` or `-c` on multiple nodes will alert on all monitored nodes for the same offline service

The command outputs information in the following format:

```
LEVEL: DETAIL
```

Where DETAIL includes detailed information about the status report, and LEVEL is:

- CRITICAL — status is critical and requires immediate attention. This indicates that more than one service is not running.

  For example:

  ```
  CRITICAL: Replicator is not running
  ```

- WARNING — status requires attention. This indicates that one service within the system is not online.
- OK — status is OK.

  For example:

  ```
  OK: All services are online
  ```

This output is easily parseable by various monitoring tools, including Nagios NRPE, and can be used to monitor the status of your services quickly without resorting to using the full trepctl output.

For example:

```
shell> check_tungsten_online
OK: All services are online
```

If you have multiple services installed, use the `-s` to specify the service:

```
shell> check_tungsten_online -s alpha
```

```
OK: All services are online
```

9.4. The check_tungsten_policy Command

The check_tungsten_policy command checks whether the policy is in *AUTOMATIC* mode or not.

Table 9.17. check_tungsten_policy Options

Option	Description
-h	Display the help text

This command only needs to be run on one node within the service; the command returns the policy mode for all nodes.

The command outputs information in the following format:

```
LEVEL: DETAIL
```

Where DETAIL includes detailed information about the status report, and LEVEL is:

- CRITICAL — status is critical and requires immediate attention. This indicates that the policy is not *AUTOMATIC*.

  For example:

  ```
  CRITICAL: Policy is MAINTENANCE
  ```

- WARNING — status requires attention. This indicates that one service within the system is not online.
- OK — status is OK.

  For example:

  ```
  OK: Policy is AUTOMATIC
  ```

This output is easily parseable by various monitoring tools, including Nagios NRPE, and can be used to monitor the status of your services quickly without resorting to using the full trepctl output.

For example:

```
shell> check_tungsten_policy
OK: Policy is AUTOMATIC
```

9.5. The check_tungsten_progress Command

The check_tungsten_progress command determines whether the replicator is actually making progress by executing a heartbeat operation and monitoring for this operation to complete within an optional time period (default is 1 second).

Table 9.18. check_tungsten_progress Options

Option	Description
-s	Service name to check (optional)
-t	Give a time period during which progress should be identified. Default: 1

The command outputs information in the following format:

```
LEVEL: DETAIL
```

Where DETAIL includes detailed information about the status report, and LEVEL is:

- CRITICAL — replicator is not making progress and either has not completed the heartbeat operation, or has failed. If failed, the reason will be shown in the DETAIL:

  For example:

  ```
  CRITICAL: Replicator is not ONLINE
  ```

- OK — replicator is making progress.

  For example:

```
OK: Replicator is making progress
```

This output is easily parseable by various monitoring tools, including Nagios NRPE, and can be used to monitor the status of your services quickly without resorting to using the full trepctl output.

The time delay can be added on busy systems to ensure that the replicator is progressing, especially if you see errors like this:

```
CRITICAL: Replicator is not ONLINE
```

For example, to wait 5 seconds to ensure the replicator is progressing:

```
shell> check_tungsten_progress  -t 5
OK: Replicator is making progress
```

Optionally, specify the replication service name using the `-s` option. This is normally only needed for Composite Active/Active deployments where there is no single default replication service.

```
shell> check_tungsten_progress -t 5 -s east_from_west
OK: Replicator is making progress
```

9.6. The check_tungsten_services Command

The check_tungsten_services command provides a simple check to confirm whether configured services are currently running. The command must be executed with a command-line option specifying which services should be checked and confirmed.

Table 9.19. check_tungsten_services Options

Option	Description
`-c`	Check the Connector service status.
`-h`	Display the help text.
`-r`	Check the replication services status.

The command outputs information in the following format:

```
LEVEL: DETAIL
```

Where DETAIL includes detailed information about the status report, and LEVEL is:

- CRITICAL — status is critical and requires immediate attention.

  For example:

  ```
  CRITICAL: Replicator is not running
  ```

- OK — status is OK.

  For example:

  ```
  OK: All services (Replicator) are online
  ```

This output is easily parseable by various monitoring tools, including Nagios NRPE, and can be used to monitor the status of your services quickly without restoring to using the full trepctl output.

> **Note**
>
> The check_tungsten_services only confirms that the services and processes are running; their state is not confirmed. To check state with a similar interface, use the check_tungsten_online command.

To check the services:

- To check the replicator services:

  ```
  shell> check_tungsten_services -r
  OK: All services (Replicator) are online
  ```

- To check the replicator and manager services are executing:

  ```
  shell> check_tungsten_services -r
  OK: All services (Replicator, Manager) are running
  ```

- To check the connector services:

```
shell> check_tungsten_services -c
OK: All services (Replicator) are online
```

9.7. The clean_release_directory Command

The clean_release_directory is located in the `tools` directory and removes older releases of the installed product from the installation directory. Over time, as tpm updates the configuration or new releases of the product, new directories with the full release information are created, but old ones are not removed in case you need to go back to a previous release.

The clean_release_directory command removes all but the five most recent installs and the current release. For example, with the following directory:

```
shell> ls -l /opt/continuent/releases
total 32
drwxr-xr-x. 2 tungsten tungsten   23 Feb 19 15:15 install
drwxr-xr-x. 8 tungsten tungsten 4096 Feb 19 15:12 tungsten-clustering-5.3.6-24_pid42731
drwxr-xr-x. 8 tungsten tungsten 4096 Feb 19 15:12 tungsten-clustering-6.0.0-487_pid44240
drwxr-x---. 7 tungsten tungsten 4096 Feb 19 15:13 tungsten-clustering-6.1.16-20_pid45954
drwxr-x---. 7 tungsten tungsten 4096 Feb 19 15:13 tungsten-clustering-6.1.25-4_pid47834
drwxr-x---. 7 tungsten tungsten 4096 Feb 19 15:13 tungsten-clustering-7.0.0-848_pid49903
drwxr-xr-x. 7 tungsten tungsten 4096 Feb 19 15:14 tungsten-clustering-7.1.2-81_pid52338
drwxr-xr-x. 7 tungsten tungsten 4096 Feb 19 15:15 tungsten-clustering-7.1.4-10_pid54710
drwxr-xr-x. 7 tungsten tungsten 4096 Feb 19 15:15 tungsten-clustering-7.0.3-141_pid57084
```

Warning

The clean_release_directory command removes old releases. Although this does not affect THL, stored data, or your configuration, it may remove working, but old, configurations, releases and versions of Tungsten Cluster.

Running clean_release_directory:

```
shell> ./tools/clean_release_directory
Deleting release directories in /opt/continuent; keeping the last five and current installation
Cleaning old releases from /opt/continuent
Deleting /opt/continuent/releases/tungsten-clustering-6.0.0-487_pid44240
Deleting /opt/continuent/releases/tungsten-clustering-5.3.6-24_pid42731
```

The resulting releases directory now contains a simpler list:

```
shell> ls -l /opt/continuent/releases
total 24
drwxr-xr-x. 2 tungsten tungsten   23 Feb 19 15:15 install
drwxr-x---. 7 tungsten tungsten 4096 Feb 19 15:13 tungsten-clustering-6.1.16-20_pid45954
drwxr-x---. 7 tungsten tungsten 4096 Feb 19 15:13 tungsten-clustering-6.1.25-4_pid47834
drwxr-x---. 7 tungsten tungsten 4096 Feb 19 15:13 tungsten-clustering-7.0.0-848_pid49903
drwxr-xr-x. 7 tungsten tungsten 4096 Feb 19 15:14 tungsten-clustering-7.1.2-81_pid52338
drwxr-xr-x. 7 tungsten tungsten 4096 Feb 19 15:15 tungsten-clustering-7.1.4-10_pid54710
drwxr-xr-x. 7 tungsten tungsten 4096 Feb 19 15:15 tungsten-clustering-7.0.3-141_pid57084
```

9.8. The cluster_backup Command

The cluster_backup command provides a simple mechanism to execute a backup inside of the cluster. It is designed to be called manually or as part of cron. The command should be added to cron on every server. When started, the command will check if the server is the current coordinator for the cluster. If not, the command will exit without an error. This design ensures that the command will only run on one server in the cluster.

The command supports command-line options that allow you to alter how and where the backup is executed.

`cluster_backup [ --agent ] [ --password ] [ --user ] [ --datasource ] [ --directory ] [ --help, -h ] [ --info, -i ] [ --json ] [ --net-ssh-option ] [ --notice, -n ] [ --offline-backup String ] [ --quiet, -q ] [ --require-slave-backup ] [ --require-automatic-mode ] [ --validate ] [ --verbose, -v ]`

Where:

Table 9.20. cluster_backup Command-line Options

Option	Description
`--agent`	The replicator backup agent to use when running the backup, as configured in cctrl
`--password`	API Admin user password (if NOT specified in INI via rest-api-admin-pass
`--user`	API Admin user (if NOT specified in INI via rest-api-admin-user

Option	Description
--datasource	Execute the backup against the specified datasource
--directory	Use this installed Tungsten directory as the base for all operations
--help, -h	Display this message
--info, -i	
--json	Provide return code and logging messages as a JSON object after the script finishes
--net-ssh-option	Set the Net::SSH option for remote system calls
--notice, -n	
--offline-backup String	Put the datasource OFFLINE prior to taking the backup.
--quiet, -q	
--require-slave-backup	Require that the backup is taken of a Replica datasource. If this is enabled and there are no Replicas available, no backup will be taken. Supply true|false. Default: true
--require-automatic-mode	The script will fail if the cluster is not in the AUTOMATIC policy mode. Supply true|false. Default: true
--validate	Only run the script validation
--verbose, -v	

After the command confirms the current server is the coordinator, it will attempt to find a datasource to backup. Unless --require-slave-backup has been disabled, only Replicas that are ONLINE will be eligible. If no datasource can be found, the command will exit with an error. The backup will then be started on the datasource.

The cluster_backup command will wait until the cctrl command has returned before exiting. The cctrl command can return prior to the backup is completed if it takes too long or if there is another error. The tungsten_nagios_backups check or similar should be used to make sure that you always have a recent backup available in the cluster.

> **Note**
>
> If manager-rest-api-authentication=true [498] and you do not have the rest-api-admin-user [512] and rest-api-admin-pass [512] not specificed in your ini, then you will also need to supply the --user and --password options for cluster_backup to work.

The cluster_backup command may also be configured for use in Composite clusters. In this use case, one backup per cluster will be created.

See Section 6.10.2, “Automating Backups” for more information.

For example:

```
shell> crontab -l
00 00 * * * /opt/continuent/tungsten/cluster-home/bin/cluster_backup >>/opt/continuent/service_logs/cluster_backup.log 2>&1
```

All output will be sent to /opt/continuent/service_logs/cluster_backup.log.

9.9. The connector Command

The connector is the wrapper script that handles the execution of the connector service.

Table 9.21. connector Commands

Option	Description
client-list [333]	Return a list of the current client connections through this connector.
cluster-status [333]	Return the cluster status, as the connector currently understands it. This is the command-line alternative to the inline cluster status command.
condrestart [333]	Restart only if already running
console [334]	Launch in the current console (instead of a daemon)
drain [seconds] [334]	An alias for graceful-stop. As of v7.0.0
dump [334]	Request a Java thread dump (if connector is running)

Option	Description
graceful-stop [seconds] [334]	Stops the connector gracefully, allowing outstanding open connections to finish and close before the connector process is stopped. All new connection requests are denied. The Connector will shut down as soon as there are no active connections. [seconds] is an integer specifying the optional time to wait before terminating the connector. Specifying no value for seconds will cause the Connector to wait indefinitely for all connections to finish. Specifying zero (0) seconds will cause the Connector to shut down immediately without waiting for existing connections to complete gracefully. As of v7.0.0, connector drain is available as an alias for connector graceful-stop.
install [334]	Install the service to automatically start when the system boots
mode [334]	Displays the mode the connector is running in, either "proxy" or "bridge"
reconfigure [334]	Reconfigure the connector by forcing the connector to reread the configuration, including the configuration files and user.map.
remove [334]	Remove the service from starting during boot
restart [334]	Stop connector if already running and then start
start [334]	Start in the background as a daemon process
status [334]	Query the current status
stop [335]	Stop if running (whether as a daemon or in another console) Optional timeout in seconds can be provided (From release 6.1.19 only)

These commands and options are described below:

– client-list [333]

Lists all ongoing connections with origin, local and remote (mysql data source) IPs/ports

```
shell> connector client-list
Executing Tungsten Connector Service --client-list ...
+---------------------+--------------------+---------------------+-----------------------+
| Client | Connector Inbound | Connector Outbound | Data Source |
+---------------------+--------------------+---------------------+-----------------------+
| /192.168.20.11:53890 | /192.168.20.11:9999 | /192.168.20.11:51612 | c1/192.168.20.11:13306 |
| /192.168.20.1:56492 | /192.168.20.11:9999 | /192.168.20.11:58556 | c1/192.168.20.11:13306 |
+---------------------+--------------------+---------------------+-----------------------+

Done Tungsten Connector Service --client-list
```

– cluster-status [333]

Returns the cluster status, in the same form as the inline tungsten cluster status command. In addition, the same information can be generated encapsulated in JSON format by adding the -json option. See Section 7.11.1.1, “Connector connector cluster status on the Command-line”, for more information.

```
shell> connector cluster-status
Executing Tungsten Connector Service --cluster-status ...
+-------------+-------------------+-------------+-------------+--------+--------+-----------------------------------------+-----------------+--------
| Data service | Data service state | Data source | Is composite | Role | State | High water | Last shun reason | Applied latency | Relative latency | Act
+-------------+-------------------+-------------+-------------+--------+--------+-----------------------------------------+-----------------+--------
| europe | ONLINE | c1 | false | master | ONLINE | 0(c1-bin.000002:0000000000001219;113912) | | 1.0 | 114.0 | 1 | 4 |
| europe | ONLINE | c2 | false | slave | ONLINE | 0(c1-bin.000002:0000000000001219;113912) | | 1.0 | 113.0 | 0 | 0 |
| europe | ONLINE | c3 | false | slave | ONLINE | 0(c1-bin.000002:0000000000001219;113912) | | 1.0 | 113.0 | 0 | 1 |
+-------------+-------------------+-------------+-------------+--------+--------+-----------------------------------------+-----------------+--------

Done Tungsten Connector Service --cluster-status
```

– condrestart [333]

Restart the connector, only if it is already running. This can be useful to use when changing configuration or performing database management within automated scripts, as the connector will be only be restart if it was previously running.

For example, if the connector is running, connector condrestart [333] operates as connector restart:

```
shell> connector condrestart
Stopping Tungsten Connector Service...
Waiting for Tungsten Connector Service to exit...
Stopped Tungsten Connector Service.
Starting Tungsten Connector Service...
```

```
Waiting for Tungsten Connector Service......
running: PID:26646
```

However, if not already running, the operation does nothing:

```
shell> connector condrestart
Stopping Tungsten Connector Service...
Tungsten Connector Service was not running.
```

– `console` [334]

Launch in the current console (instead of a daemon)

– `drain [seconds]` [334]

An alias for graceful-stop. As of v7.0.0

– `dump` [334]

Request a Java thread dump (if connector is running)

– `graceful-stop [seconds]` [334]

Stops the connector gracefully, allowing outstanding open connections to finish and close before the connector process is stopped. All new connection requests are denied. The Connector will shut down as soon as there are no active connections. [seconds] is an integer specifying the optional time to wait before terminating the connector. Specifying no value for seconds will cause the Connector to wait indefinitely for all connections to finish. Specifying zero (0) seconds will cause the Connector to shut down immediately without waiting for existing connections to complete gracefully. As of v7.0.0, connector drain is available as an alias for connector graceful-stop.

– `install` [334]

Installs the startup scripts for running the connector at boot. For an alternative method of deploying these start-up scripts, see deployall.

– `mode` [334]

– `reconfigure` [334]

Reloads the configuration without touching existing connections. New configuration only applies to new connections. Note that this command won't work for the following settings: connection.keepAlive.interval, connection.keepAlive.timeout, server.listen.address, server.port, server.port.binding.timeout, server.port.binding.retry.delay (when reaching server.max_connections only)

– `remove` [334]

Removes the startup scripts for running the connector at boot. For an alternative method of removing these start-up scripts, see undeployall.

– `restart` [334]

Stops the connector, if it is already running, and then restarts it:

```
shell> connector restart
Stopping Tungsten Connector Service...
Stopped Tungsten Connector Service.
Starting Tungsten Connector Service...
Waiting for Tungsten Connector Service......
running: PID:26248
```

– `start` [334]

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

```
shell> connector start
Starting Tungsten Connector Service...
```

– `status` [334]

Checks the execution status of the connector:

```
shell> connector status
Tungsten Connector Service is running: PID:27015, Wrapper:STARTED, Java:STARTED
```

If the connector is not running:

```
shell> connector status
Tungsten Connector Service is not running.
```

This only provides the execution state of the connector, not the actual state of replication. To get detailed information on the status of replication use trepctl status.

– stop [335]

Stops the connector if it is already running, terminating all connections immediately:

```
shell> connector stop
Stopping Tungsten Connector Service...
Waiting for Tungsten Connector Service to exit...
Stopped Tungsten Connector Service.
```

Optional timeout can be provided, for example:

```
shell> connector stop 30
Stopping Tungsten Connector Service...
Waiting for Tungsten Connector Service to exit...
Stopped Tungsten Connector Service.
```

This would cause the connector to wait 30 seconds to allow connections to terminate cleanly, after the timeout, any remaining connections will be terminated.

9.10. The deploylall Command

The deploylall tool installs the required startup scripts into the correct location so that all required services can be automatically started and stopped during the startup and shutdown of your server.

To use, the tool should be executed with superuser privileges, either directly using sudo, or by logging in as the superuser and running the command directly.

The script will automatically detect the initialization system in use (systemd or initd) and prefer systemd when both are available.

```
shell> sudo deploylall
```

Important

In order for the configuration to persist during future updates, and/or to execute the deploylall script at install time, you should add the install=true [494] tpm option to your configuration

The startup scripts are added to the correct run levels to enable operation during standard startup and shutdown levels.

Note

For **systemd** configurations only:

For continuity of service reasons, the deploylall script will NOT restart individual components if they had already been previously started by other methods. It will only install systemd scripts. This implies that, right after a call to deploylall and before host/component restart, the system will stay in a mixed mode where systemd scripts are in place but components that were started without systemd, won't be controllable by it.

In order to align the configuration, you will need to run

```
shell> component stop sysd
shell> sudo systemctl start tcomponent
```

For example:

```
shell> connector stop sysd
shell> sudo systemctl start tconnector
```

Important

This note affects all versions up to and including v7.0.2. The workaround mentioned below is included as default from v7.0.3 onwards.

When a service is controlled by systemd, the relevant OS limits (such as open file limits) are not controlled in the normal way (via settings in the limits.conf file) and therefore for clusters with heavy workloads there is a risk that you may experience open file limits being exceeded which will affect the clusters operation.

To resolve this, you must ensure you increase the limits for each service. Follow the steps below to do this:

- Edit the service files in /etc/systemd/system. There will be one file for each service, for example treplicator.service

- Under the `[service]` stanza, add the following:

```
LimitNOFILE=65535
```

- When you have changed all the files, reload the systemctl by issuing the following:

```
systemctl daemon-reload
```

- Retstart each of the services, e.g:

```
systemctl restart treplicator
```

Note that the restart will cause a momentary outage to each component therefore only do this when you are sure it is safe to do so, and ensure you cluster is in *MAINTENANCE* mode.

See Section 4.3, "Configuring Startup on Boot".

To remove the scripts from the system, use undeployall.

9.11. 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 9.22. dsctl Commands

Option	Description
get	Return the available position information
help	Print the help display
reset	Clear the datasource position information
set	Set the position

These must be used in conjunction with one of the following options to select the required datasources or service:

Table 9.23. dsctl Command-line Options

Option	Description
`-conf`	Path to the static services properties file
`-ds`	Name of the datasource
`-service`	Name of the replication service to get information from

If more than one service or datasource has been configured, one of these options much be used to select the service. Otherwise, by default dsctl will use the corresponding configured service.

9.11.1. dsctl get Command

Table 9.24. dsctl Command-line Options

Option	Description
`-ascmd`	Generates the command required to set the datasource to the current position

Returns the current datasource status and position, returning the information as a JSON string. The example below has been formatted for clarity:

```
shell> dsctl get
[
  {
    "extract_timestamp":"2025-02-19 15:47:34.0",
    "eventid":"mysql-bin.000011:0000000000072857;8196",
    "fragno":0,
    "last_frag":true,
    "seqno":214,
```

```
    "update_timestamp":"2025-02-19 15:47:34.0",
    "shard_id":"hr",
    "applied_latency":0,
    "epoch_number":0,
    "task_id":0,
    "source_id":"db1"
  }
]
```

When the `-ascmd` option is used, the information is output in form of a command:

```
shell> dsctl get -ascmd
dsctl set -seqno 214 -epoch 0 -event-id "mysql-bin.000011:0000000000072857;8196" -source-id "db1"
```

If the `-reset` is used, then the generated command also includes the option. For example:

```
shell> dsctl get -ascmd -reset
dsctl set -seqno 214 -epoch 0 -event-id "mysql-bin.000011:0000000000072857;8196" -source-id "db1" -reset
```

9.11.2. dsctl set Command

Table 9.25. dsctl Command-line Options

Option	Description
`-epoch`	Epoch Number
`-event-id`	Source Event ID
`-reset`	Resets the datasources before performing set operation
`-seqno`	Sequence number
`-source-id`	Source ID

Sets the current replicator position. When using this option, the `-seqno`, `-epoch`, `-event-id`, and `-source-id` options must be specified to set the corresponding values in the replicator.

If a configuration exists, and you do not specify the `-reset` then an error will be displayed, for example:

```
shell> dsctl set -seqno 214 -epoch 0 -event-id "mysql-bin.000011:0000000000072857;8196" -source-id "db1"
Cannot set position unless existing position data is removed - reset first
```

Either issue the reset as a seperate option, or include in the single command, for example:

```
shell> dsctl reset
Service "alpha" datasource "global" catalog information cleared

shell> dsctl set -seqno 214 -epoch 0 -event-id "mysql-bin.000011:0000000000072857;8196" -source-id "db1"
Service "alpha" datasource "global" position was set to: seqno=214 epoch_number=0 eventid=mysql-bin.000011:0000000000072857;8196 source_id=db1
```

Or:

```
shell> dsctl set -seqno 214 -epoch 0 -event-id "mysql-bin.000011:0000000000072857;8196" -source-id "db1" -reset
Service "alpha" datasource "global" catalog information cleared
Service "alpha" datasource "global" position was set to: seqno=214 epoch_number=0 eventid=mysql-bin.000011:0000000000072857;8196 source_id=db1
```

9.11.3. dsctl reset Command

Clears the current replicator status and position information:

```
shell> dsctl reset
Service "alpha" datasource "global" catalog information cleared
```

9.11.4. dsctl help Command

Displays the current help text:

```
shell> dsctl help
Datasource Utility
Syntax: dsctl [conf|service] [-ds name] [operation]
Configuration (required if there's more than one service):
  -conf path       - Path to a static-<svc>.properties file
     OR
  -service name    - Name of a replication service to get datasource configuration from
Options:
 [-ds name]        - Name of the datasource (default: global)
```

```
Operations:
  get              - Return all available position information
     [-ascmd]      - Return all available position information as command
  set -seqno ###   - Set position (all four parameters required)
      -epoch ###
      -event-id AAAAAAAAA.######:#######
      -source-id AAA.AAA.AAA
     [-reset]      - Optionally reset the information first
  reset            - Clear datasource position information
  help             - Print this help display
```

9.12. env.sh Script

After installation, the env.sh can be used to setup the local environment, such as appending to the local $PATH.

If profile-script [507] is set during installation, then the local profile script will also be updated to ensure the env.sh file is loaded at login of the OS user.

```
shell> cat .bash_profile
...
# Begin Tungsten Environment for /opt/continuent
# Include the Tungsten variables
# Anything in this section may be changed during the next operation
if [ -f "/opt/continuent/share/env.sh" ]; then
    . "/opt/continuent/share/env.sh"
fi
# End Tungsten Environment for /opt/continuent
```

If not set, then the script can manually be sourced

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

If executable-prefix [493] is set, then the env.sh script will also configure aliases for all of the common executable binaries

For example, if executable-prefix [493] has been set to "mm", then aliases for executable binaries will be prefixed with this value, as shown in the small example below:

```
shell> alias
...
alias mm_thl='/opt/continuent/tungsten/tungsten-replicator/bin/thl'
alias mm_tpm='/opt/continuent/tungsten/tools/tpm'
alias mm_trepctl='/opt/continuent/tungsten/tungsten-replicator/bin/trepctl'
...
```

9.13. The manager Command

The manager is the wrapper script that handles the execution of the manager service.

Table 9.26. manager Commands

Option	Description
condrestart [338]	Restart only if already running
console [339]	Launch in the current console (instead of a daemon)
dump [339]	Request a Java thread dump (if manager is running)
install [339]	Install the service to automatically start when the system boots
remove [339]	Remove the service from starting during boot
restart [339]	Stop manager if already running and then start
start [339]	Start in the background as a daemon process
status [339]	Query the current status
stop [339]	Stop if running (whether as a daemon or in another console)

These commands and options are described below:

– condrestart [338]

Restart the manager, only if it is already running. This can be useful to use when changing configuration or performing database management within automated scripts, as the manager will be only be restart if it was previously running.

For example, if the manager is running, manager condrestart operates as manager restart:

```
shell> manager condrestart
Stopping Tungsten Manager Service...
Waiting for Tungsten Manager Service to exit...
Stopped Tungsten Manager Service.
Starting Tungsten Manager Service...
Waiting for Tungsten Manager Service......
running: PID:26646
```

However, if not already running, the operation does nothing:

```
shell> manager condrestart
Stopping Tungsten Manager Service...
Tungsten Manager Service was not running.
```

– `console` [339]

Launch in the current console (instead of a daemon)

– `dump` [339]

Request a Java thread dump (if manager is running)

– `install` [339]

Installs the startup scripts for running the manager at boot. For an alternative method of deploying these start-up scripts, see deployall.

– `remove` [339]

Removes the startup scripts for running the manager at boot. For an alternative method of removing these start-up scripts, see undeployall.

– `restart` [339]

Warning

Restarting a running manager temporarily stops and restarts replication.

Stops the manager, if it is already running, and then restarts it:

```
shell> manager restart
Stopping Tungsten Manager Service...
Stopped Tungsten Manager Service.
Starting Tungsten Manager Service...
Waiting for Tungsten Manager Service......
running: PID:26248
```

– `start` [339]

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

```
shell> manager start
Starting Tungsten Manager Service...
```

– `status` [339]

Checks the execution status of the manager:

```
shell> manager status
Tungsten Manager Service is running: PID:27015, Wrapper:STARTED, Java:STARTED
```

If the manager is not running:

```
shell> manager status
Tungsten Manager Service is not running.
```

This only provides the execution state of the manager, not the actual state of replication. To get detailed information on the status of replication use trepctl status.

– `stop` [339]

Stops the manager if it is already running:

```
shell> manager stop
Stopping Tungsten Manager Service...
Waiting for Tungsten Manager Service to exit...
Stopped Tungsten Manager Service.
```

If the cluster was configured with `auto-enable=false` [474] then you will need to put each node online individually.

9.14. The multi_trepctl Command

The multi_trepctl command provides unified status and operation support across your Tungsten Cluster installation across multiple hosts without the need to run the trepctl command across multiple hosts and/or services individually.

```
multi_trepctl
masterof
backups [ --by-service ] [ --fields appliedLastSeqNo | appliedLatency | host | role | serviceName | state ]
heartbeat [ --host, --hosts self ]
list [ --output json | list | name | tab | yaml ] [ --path, --paths ] [ --role, --roles ]
run [ --service, --services ] [ --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 |
```

9.14.1. 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 9.27. multi_trepctl Command-line Options

Option	Description
--by-service [340]	Sort the output by the service name
--fields [341]	Fields to be output during during summary
--host [341], --hosts [341]	Host or hosts on which to limit output
--output [341]	Specify the output format
--paths [341], --path [341]	Directory or directories to check when looking for tools
--role [342], --roles [342]	Role or roles on which to limit output
--service [342], --services [342]	Service or services on which to limit output
--skip-headers [342]	Skip the headers
--sort-by [342]	Sort by a specified field

Where:

- --by-service [340]

  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` [341]

  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` [341], `--hosts` [341]

  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` [341]

  Specify the output format.

  Table 9.28. multi_trepctl `--output` Option

Option	`--output` [341]	
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` [341], `--paths` [341]

  Limit the search for trepctl to the specified path or comma-separated list of paths. On a deployment with multiple services, the output will be limited by the services installed within the specified directories:

```
shell> multi_trepctl --path /opt/replicator
| host | servicename | role   | state   | appliedlastseqno | appliedlatency |
```

```
| db1  | west        | slave | ONLINE |               3 |          0.450 |
| db2  | west        | slave | ONLINE |               3 |          0.481 |
| db3  | west        | slave | ONLINE |               3 |          0.484 |
| db4  | east        | slave | ONLINE |               4 |          0.460 |
| db5  | east        | slave | ONLINE |               4 |          0.451 |
| db6  | east        | slave | ONLINE |               4 |          0.496 |
```

This is also useful when control of cross-site replicators is desired in MSMM topologies prior to v6.0.0.

For example, take all cross-site replicators offline:

```
shell> multi_trepctl --path /opt/replicator offline
```

To bring all cross-site replicators online:

```
shell> multi_trepctl --path /opt/replicator online
```

- `--role` [342], `--roles` [342]

  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` [342], `--services` [342]

  Limit the output to the specified service or comma-separated list of services:

```
shell>  multi_trepctl --service=east
| host  | servicename | role   | state         | appliedlastseqno | appliedlatency |
| east1 | east        | master | ONLINE        |               53 |          0.000 |
| west1 | east        | slave  | ONLINE        |               53 |        119.834 |
| west2 | east        | slave  | ONLINE        |               53 |        181.128 |
| west3 | east        | slave  | ONLINE        |               53 |        204.790 |
```

- `--skip-headers` [342]

  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` [342]

  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 |
```

9.14.2. 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 9.29. multi_trepctl Commands

Option	Description
masterof	List all the Primaries of configured hosts and services
backups	List all the backups available across all configured hosts and services
heartbeat	Inserts a heartbeat on all Primaries within the service
list	List the information about each service
run	Run the specified trepctl command on all hosts/services

In addition to the two primary commands, multi_trepctl can execute commands that would normally be applied to trepctl, running them on each selected host, service or directory according to the options. The output format and expectation is controlled through the list and run commands.

For example:

```
shell> multi_trepctl status
```

Outputs the long form of the status information (as per trepctl status) for each identified host.

9.14.2.1. 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 |
```

9.14.2.2. 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:
```

9.14.2.3. multi_trepctl masterof Command

Lists which hosts are Primaries of others within the configured services.

```
shell> multi_trepctl masterof
| servicename | host  | uri              |
| alpha       | host1 | thl://host1:2112/ |
```

9.14.2.4. 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.

9.14.2.5. multi_trepctl run Command

The multi_trepctl run command can be used where the output of the corresponding trepctl command cannot be formatted into a convenient list. For example, to execute a backup on every host within a deployment:

```
shell> multi_trepctl run backup
```

The same filters and host or service selection can also be made:

```
shell> multi_trepctl run backup --hosts=host1,host2,host3
host: host1
servicename: firstrep
output: |
  Backup completed successfully; URI=storage://file-system/store-0000000005.properties
---
host: host2
servicename: firstrep
output: |
  Backup completed successfully; URI=storage://file-system/store-0000000001.properties
```

```
---
host: host3
servicename: firstrep
output: |
  Backup completed successfully; URI=storage://file-system/store-0000000001.properties
...
```

Return from the command will only take place when remote commands on each host have completed and returned.

9.15. The query Command

Table 9.30. query Common Options

Option	Description
-conf PATH	Configuration file that contains values for connection properties (url, user and password)
-file PATH	File containing the SQL commands to run. If missing, read SQL commands from STDIN
-password	Prompt for password
-url JDBCURL	JDBC url of the database to connect to
-user USER	User used to connect to the database

The query command line tool can be used to issue SQL statements against a database.

The queries can either be entered via STDIN, or read in from a text file

The following example shows a SELECT statement issued via STDIN

```
shell> query -url "jdbc:mysql:thin://db2:13306/" -user tungsten -password
Enter password: ********
select * from tungsten_nyc.trep_commit_seqno;

[
  {
    "statement":"select * from tungsten_alpha.trep_commit_seqno;","rc":0,"results":
    [
      [
        {
          "task_id":0,
          "seqno":1,
          "fragno":-1,
          "last_frag":"1",
          "source_id":"db1",
          "epoch_number":0,
          "eventid":"mysql-bin.000013:0000000000001171;93",
          "applied_latency":null,
          "update_timestamp":"2025-02-20 10:58:41.0",
          "shard_id":null,
          "extract_timestamp":"2025-02-20 10:58:41.0",
          "connection_id":183
        }
      ]
    ],
    "error":null
  }
]
```

9.16. The replicator Command

The replicator is the wrapper script that handles the execution of the replicator service.

Table 9.31. replicator Commands

Option	Description
condrestart [345]	Restart only if already running
console [345]	Launch in the current console (instead of a daemon)
dump [345]	Request a Java thread dump (if replicator is running)
install [345]	Install the service to automatically start when the system boots
remove [345]	Remove the service from starting during boot

Option	Description
restart [345]	Stop replicator if already running and then start
start [346]	Start in the background as a daemon process
status [346]	Query the current status
stop [346]	Stop if running (whether as a daemon or in another console)

These commands and options are described below:

– condrestart [345]

Table 9.32. replicator Commands Options for condrestart [345]

Option	Description
offline	Start in OFFLINE state

Restart the replicator, only if it is already running. This can be useful to use when changing configuration or performing database management within automated scripts, as the replicator will be only be restarted 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 [345]

Table 9.33. replicator Commands Options for console [345]

Option	Description
offline	Start in OFFLINE state

Launch in the current console (instead of a daemon)

– dump [345]

Request a Java thread dump (if replicator is running)

– install [345]

Installs the startup scripts for running the replicator at boot. For an alternative method of deploying these start-up scripts, see deployall.

– remove [345]

Removes the startup scripts for running the replicator at boot. For an alternative method of removing these start-up scripts, see undeployall.

– restart [345]

Table 9.34. replicator Commands Options for restart [345]

Option	Description
offline	Stop and restart in OFFLINE state

Warning

Restarting a running replicator temporarily stops and restarts replication.

Stops the replicator, if it is already running, and then restarts it:

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

```
Stopped Tungsten Replicator Service.
Starting Tungsten Replicator Service...
Waiting for Tungsten Replicator Service......
running: PID:26248
```

– start [346]

Table 9.35. replicator Commands Options for start [346]

Option	Description
offline	Start in OFFLINE state

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

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

– status [346]

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 [346]

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.
```

If the cluster was configured with auto-enable=false [474] then you will need to put each node online individually.

9.17. The startall Command

The startall will start all configured services within the configured directory:

```
shell> startall
Starting Tungsten Replicator Service...
Waiting for Tungsten Replicator Service......
running: PID:2578
Starting Tungsten Manager Service...
Waiting for Tungsten Manager Service.........
running: PID:2722
Starting Tungsten Connector Service...
Waiting for Tungsten Connector Service.....
running: PID:2917
```

If a service is already running, then a notification of the current state will be provided, for example:

```
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-clustering-7.0.3-141_pid25898/tungsten-replicator/bin/../var/treplicator.pid
```

9.18. The stopall Command

The stopall command stops all running services if they are already running:

```
shell> stopall
Stopping Tungsten Connector Service...
Waiting for Tungsten Connector Service to exit...
Stopped Tungsten Connector Service.
```

```
Stopping Tungsten Manager Service...
Stopped Tungsten Manager Service.
Stopping Tungsten Replicator Service...
Waiting for Tungsten Replicator Service to exit...
Stopped Tungsten Replicator Service.
```

9.19. The tapi Command

The tapi command was added in version 7.0.0

The tapi command is designed to act as an interface to the Tungsten REST APIv2.

The original goal of this script was to provide a way to test and exercise the API which then grew to include additional functionality and convenience options.

Usage for tapi:

```
shell> tapi [options] [API_call_path]
```

See the tables below for a list of valid arguments:

Table 9.36. tapi Generic Options

Option	Description
`--debug`, `-d`	
`--help`, `-h`	
`--quiet`, `-q`	
`--service`, `-s`	Specify a service name. Used with --run, --getpolicy, --setpolicy and the --policy NRPE plugin
`--verbose`, `-v`	

Table 9.37. tapi CURL-related Options

Option	Description
`--curl`	Specify where to find curl executable (default: curl)
`--get`	Set the curl request to GET
`--host`	Specify the host name to connect to (default: 127.0.0.1)
`--nossl`, `--http`	Specify NON-SSL/http-only connection (default: https, SSL enabled)
`--port`	Specify the port number to connect to (default: 8090)
`--post`	Set the curl request to POST
`--put`	Set the curl request to PUT
`-p`, `--password`	Specify auth password (default: not defined)
`--request {GET|POST|PUT}`	Specify the curl request method (default: none)
`--user`, `-u`	Specify auth user (default: not defined)

Table 9.38. tapi Nagios/NRPE/Zabbix-related Options

Option	Description
`--allperf`, `--per-replica-perfdata`	When --allperf is specified with --perf, each service is listed separately; used with --latency
`-c`, `--critical`	Specify the critical threshold value in seconds; used with --latency
`--latency`	Check the replicator for latency as an NRPE plugin
`--online`	Check the manager to see if the datasource is online as an NRPE plugin
`--perf`, `--perfdata`	Display additional performance data; used with --latency
`--policy`	Check the manager to see if the cluster is in AUTOMATIC policy mode as an NRPE plugin
`--progress`	Check the replicator to see that it is making progress over time as an NRPE plugin

Option	Description
`--services`	Check the host for running services (Manager, Connector and Replicator) as an NRPE plugin
`--wait`, `-W`	Specify the time to wait in seconds during the --progress test (default: 1 second)
`--warn`, `-w`	Specify the warning threshold value in seconds; used with --latency
`--zabbix`, `-z`	Use Zabbix-style exit messages which are the same as the exit code, unlike NRPE which outputs a string starting with OK, WARNING or CRITICAL

Table 9.39. tapi Admin-related Options

Option	Description
`--create`	Connect to the specified host and port, then call the createAdminUser path with a special body to instantiate the user. Must supply --create-user and --create-password.
`--create-pass`	Sets password for the APIv2 admin user created using the --create flag. --create and --create-user must also be supplied.
`--create-user`	Sets the user name for the APIv2 admin user created using the --create flag. --create and --create-pass must also be supplied.
`--getds`	Display the main Manager dataservice name
`--getmain`	Display the main Replicator service name
`--getpolicy`	Get the current local policy. If --service {svcname} is specified, then the policy will be displayed for that service.
`--getports`	Display the API default port for each component
`--getsubs`	Display the Replicator sub-service(s)
`--includeServiceName`	Output service names.
`--no-connectors`	Append '?no-connectors=1' to all API calls to disable the Manager internal status calls to the Connectors
`--setauto`	Set the current policy to AUTOMATIC
`--setmaint`	Set the current policy to MAINTENANCE
`--setpolicy`	Set the current local policy, e.g. auto or maint. If --service {svcname} is specified, then the policy will be set for that service. One-letter abbreviations are allowed for the policy mode names.
`--topology`	Display the main Manager topology

Table 9.40. tapi Filter-related Options

Option	Description
`--connector`, `-C`	Specify the connector path
`--filter`, `-F`	Specify {string} to Limit the call path (typically used with --listapi)
`--manager`, `-M`	Specify the manager path
`--replicator`, `-R`	Specify the replicator path

Table 9.41. tapi API-related Options

Option	Description
`--allservices`	Operate on all available services. Used with --run
`--follow`, `--descend`, `-f`	If the resulting payload type is URIsPayload then call every key in the payload as a sub-item of the original path
`--listapi`, `-L`	List all available REST API calls by unique key
`--run`, `--alias`, `-r`	Specify an API call name by unique key. Use --listapi to see all keys/aliases

Table 9.42. tapi Status-related Options

Option	Description
`--affinity`	Display only the affinity-specific values from the '/connector/configuration/module/connector' path
`--allstats`	Display all statistics cluster-wide using the Manager API
`--connectorstatus`	Use the '/connector/configuration/module/connector' path
`--cs`, `--clusterstatus`	Use the 'manager/cluster/status' path
`--mgrstats`, `--managerstats`	Display Manager statistics cluster-wide using the Manager API
`--routers`	Display Connector statistics cluster-wide using the Manager API
`--status`	Use the 'manager/status' path
`--trstats`	Display Replicator statistics cluster-wide using the Manager API

Table 9.43. tapi Backup and Restore-related Options

Option	Description
`--backuptimeout`	Specify the backup timeout in seconds
`--mysqldump`	Use the mysqldump backup agent
`--restoretimeout`	Specify the restore timeout in seconds
`--restoreuri`	Specify the restore URI
`--storageagent`	Specify the storage agent
`--xtrabackup`	Use the xtrabackup backup agent
`--xtrabackupFull`	Use the xtrabackupFull backup agent
`--xtrabackupIncremental`	Use the xtrabackupIncremental backup agent

There are many cli options which invoke the various functions of the tapi script.

With no options specified, the tapi script will construct a curl command to query the Tungsten Manager API (default https://127.0.0.1:8090) using the {API_call_path} provided on the command line.

Basic Example

Running tapi `-M --status` would result in something like the following:

```
shell> tapi manager/status
Executing: curl -s --user tungsten:secret --insecure --request GET 'https://127.0.0.1:8090/api/v2/manager/status'
{"payloadType":"StatusPayload","payloadVersion":"1","payload":{"dataServiceName":"east","dataSourceName":"db1-demo.continuent.com",
"startTime":"2021-03-25T16:47:50.523 UTC","uptimeSeconds":1909,"state":"ONLINE","isCoordinator":true,"isWitness":false,
"managerPID":4052,"parentPID":4031,"policyMode":"MAINTENANCE","coordinator":"db1-demo.continuent.com"}}
```

If both `--user` and `--password` are defined, curl will use them in the call.

If either or both `--user` and `--password` are missing, tapi will attempt to derive the values using the tpm command.

Simple Admin Command Examples

```
tapi --getds
tapi --topology
tapi --setmaint
tapi --setauto

tapi --setpolicy maintenance
tapi --setpolicy m
tapi --setpolicy automatic
tapi --setpolicy a
```

Simple Replicator Command Examples

```
tapi --getmain
tapi --getsubs
```

Running API Command Examples

Show all available API calls in four columns: component, unique key, request type, request path:

```
tapi -R --listapi
```

Show all available API calls filtered by Manager [-M], Replicator [-R] or Connector [-C]:

```
tapi --listapi -M
tapi --listapi -R
tapi --listapi -C
```

Execute an API call by unique key per component:

```
tapi -R --listapi
tapi -R --run ping
tapi -R -v --run offline (like `trepctl offline`)
tapi -R -v --run online (like `trepctl online`)
```

Backup and Restore Examples

```
tapi -v -R --run backup --mysqldump
tapi -v -R --run backup --xtrabackupFull
tapi -v -R -j --run restore
tapi -v -R -j --run task d28465a2-6023-47c4-9a4c-20f93514db75
```

9.20. 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 9.44, "thl Options".

Table 9.44. thl Options

Option	Description
-conf path	Path to the configuration file containing the required replicator service configuration
-service servicename	Name of the service to be used when looking for THL information

For example, to execute a command on a specific service:

```
shell> thl index -service firstrep
```

Individual operations are selected by use of a specific command to the thl command. Supported commands are:

- dsctl — obtain syntax that can be used with the dsctl command to assist in positioning of the replicator.
- index — obtain a list of available THL files.
- info — obtain summary information about the available THL data.
- list — list one or more THL events.
- purge — purge THL data.
- tail — live tail of THL files.
- help — get the command help text.

Further information on each of these operations is provided in the following sections.

9.20.1. thl Position Commands

The thl command supports a number of position and selection command-line options that can be used to select an individual THL event, or a range of events, to be displayed.

- -seqno # [350]

  Valid for: thl list

  Output the THL sequence for the specific sequence number. When reviewing or searching for a specific sequence number, for example when the application of a sequence on a Replica has failed, the replication data for that sequence number can be individually viewed. For example:

  The output also includes the filename of the THL file on disk where the THL event is located:

  ```
  shell> thl list -seqno 99
  SEQ# = 99 / FRAG# = 0 (last frag)
  ```

```
- FILE = thl.data.0000000001
- TIME = 2025-02-21 09:11:28.0
- EPOCH# = 2
- EVENTID = mysql-bin.000015:0000000000030282;44
- SOURCEID = db1
- METADATA = [mysql_server_id=10;mysql_thread_id=44;dbms_type=mysql;tz_aware=true;service=alpha;shard=hr]
- GLOBAL OPTIONS = [autocommit=1;sql_auto_is_null=0;foreign_key_checks=1;unique_checks=1;sql_mode='NO_ENGINE_SUBSTITUTION,ONLY_FULL_GROUP_BY,STRICT_TRAN
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1, time_zone = '+00:00', ##charset = UTF-8]
- SQL(0) =
- ACTION = INSERT
- SCHEMA = hr
- TABLE = jobs
- ROW# = 0
- COL(1: ) (size = 6) = PU_MAN
- COL(2: ) (size = 18) = Purchasing Manager
- COL(3: ) (size = 4) = 8000
- COL(4: ) (size = 5) = 15000
```

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 #` [351] and/or `-high #` [351]

  `-from #` [351] and/or `-to #` [351]

  Valid for: thl list, thl purge

  Specify the start (`-from` [351]) or end (`-to` [351]) of the range of sequence numbers to be output. If only `-from` [351] is specified, then all sequence numbers from that number to the end of the THL are output. If `-to` [351] is specified, all sequence numbers from the start of the available log file to the specified sequence number are output. If both numbers are specified, output all the sequence numbers within the specified range.

  For example:

  ```
  shell> thl list -from 320
  ```

  Or:

  ```
  shell> thl list -low 320
  ```

  Will output all the sequence number fragments from number 320.

  ```
  shell> thl list -to 540
  ```

  Or:

  ```
  shell> thl list -high 540
  ```

  Will output all the sequence number fragments up to and including 540.

  ```
  shell> thl list -from 320 -to 540
  ```

  Or:

  ```
  shell> thl list -low 320 -high 540
  ```

  Will output all the sequence number fragments from number 320 up to, and including, sequence number 540.

- `-first` [351]

  Valid for: thl list, thl purge

  The `-first` [351] selects only the first stored THL event. For example:

  ```
  shell> thl list -first
  SEQ# = 0 / FRAG# = 0 (last frag)
  - FILE = thl.data.0000000001
  - TIME = 2025-02-20 10:56:33.0
  - EPOCH# = 0
  - EVENTID = mysql-bin.000013:0000000000000543;84
  ```

```
- SOURCEID = db1
```

- `-first #` [352]

  Valid for: thl list, thl purge

  The `-first #` [352] selects the specified number of events, starting from the first event. For example:

  ```
  shell> thl list -first 5
  ```

  Would display the first five events from the stored THL.

- `-last` [352]

  Valid for: thl list, thl purge

  The `-last` [352] selects only the last stored THL event. For example:

  ```
  shell> thl list -last
  SEQ# = 216 / FRAG# = 0 (last frag)
  - FILE = thl.data.0000000001
  - TIME = 2025-02-21 09:12:04.0
  - EPOCH# = 2
  - EVENTID = mysql-bin.000015:0000000000072872;102
  - SOURCEID = db1
  ...
  ```

  The use of this option can be particularly useful in the event of synchronisation or THL corruption due to a lack of diskspace. Using the thl purge command, the last THL event can be easily removed without having to work out the ranges and index information:

  ```
  shell> thl purge -last
  ```

- `-last #` [352]

  Valid for: thl list, thl purge

  The `-last #` [352] selects the specified number of events, starting from the last-# event. For example:

  ```
  shell> thl list -last 5
  ```

  When the THL index contains events from 1558-1601, would display events 1597 through to 1601.

9.20.2. thl dsctl Command

The dsctl command to the thl command outputs a dsctl command that can be executed against a replicator to assist in repositioning.

The command is displayed only, it is not executed

`thl` dsctl
[-seqno #]
[-event #] [-first] [-last]

- `-event eventid` [352]

  Output a dsctl command for the given eventid, for example:

  ```
  shell> thl dsctl -event mysql-bin.000017:0000000074628349
  dsctl -service alpha set -reset -seqno 916 -epoch 538 -event-id "mysql-bin.000017:0000000074628349;62" -source-id "centos1"
  ```

- `-seqno #` [350]

  Output a dsctl command for the given sequence number, for example:

  ```
  shell> thl dsctl -seqno 916
  dsctl -service alpha set -reset -seqno 916 -epoch 538 -event-id "mysql-bin.000017:0000000074628349;62" -source-id "centos1"
  ```

9.20.3. thl index Command

The index command to thl provides a list of all the available THL files and the sequence number range stored within each file:

```
shell> thl index
LogIndexEntry thl.data.0000000001(0:113)
LogIndexEntry thl.data.0000000002(114:278)
LogIndexEntry thl.data.0000000003(279:375)
LogIndexEntry thl.data.0000000004(376:472)
```

```
LogIndexEntry thl.data.0000000005(473:569)
LogIndexEntry thl.data.0000000006(570:941)
LogIndexEntry thl.data.0000000007(942:1494)
LogIndexEntry thl.data.0000000008(1495:1658)
LogIndexEntry thl.data.0000000009(1659:1755)
LogIndexEntry thl.data.0000000010(1756:1852)
LogIndexEntry thl.data.0000000011(1853:1949)
LogIndexEntry thl.data.0000000012(1950:2046)
LogIndexEntry thl.data.0000000013(2047:2563)
```

The optional argument `-no-checksum` [354] ignores the checksum information on events in the event that the checksum is corrupt.

9.20.4. 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, 2019-12-18 11:53:00)
newest file = thl.data.0000000041 (0.98 MB, 2019-12-18 12:34:32)
```

The optional argument `-no-checksum` [354] ignores the checksum information on events in the event that the checksum is corrupt.

9.20.5. thl list Command

The list command to the thl command outputs a list of the sequence number information from the THL. By default, the entire THL as stored on disk is output. Command-line options enable you to select individual sequence numbers, sequence number ranges, or all the sequence information from a single file.

`thl` list
[-seqno #]
[-low #] | [-from #] | [-high #] | [-to #]
[-last] [-last #] [-first] [-first #]
[-event #]
[-file filename] [-no-checksum] [-sql] [-sizes] [-sizesdetail] [-sizessummary] [-charset] [-headers] [-json] [-specs-] [-charset]

- `-event eventid` [352]

  Output THL found that matches the provided eventid. If no exact match found, a message will display details of an approximate match if found. See example below:

  An exact match is found:

  ```
  shell> thl list -event mysql-bin.000015:0000000000030282
  SEQ# = 99 / FRAG# = 0 (last frag)
  - FILE = thl.data.0000000001
  - TIME = 2025-02-21 09:11:28.0
  - EPOCH# = 2
  - EVENTID = mysql-bin.000015:0000000000030282;44
  - SOURCEID = db1
  ...
  ```

  No match found:

  ```
  Event not found : Approximative match found between seqno 915 (mysql-bin.000017:0000000074628153;62) and seqno 916 (mysql-bin.000017:0000000074628349;62
  ```

- `-file filename` [353]

  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` [353]

  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` [353]

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` [354]

  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.

```
shell> thl list -headers
...
SEQ# = 215 / FRAG# = 0 (last frag)
- FILE = thl.data.0000000001
- TIME = 2025-02-21 09:11:29.0
- EPOCH# = 2
- EVENTID = mysql-bin.000015:0000000000072471;44
- SOURCEID = db1
SEQ# = 216 / FRAG# = 0 (last frag)
- FILE = thl.data.0000000001
- TIME = 2025-02-21 09:12:04.0
- EPOCH# = 2
- EVENTID = mysql-bin.000015:0000000000072872;102
- SOURCEID = db1
...
```

  For more information on the fields displayed, see Section E.1.1, “THL Format”.

- `-json`

  Only valid with the `-headers` option, the header information is output for the selected sequence numbers from the THL in JSON format. The field contents are identical, with each fragment of each THL sequence being contained in a JSON object, with the output consisting of an array of the these sequence objects. For example:

```
[
...
 {
   "seqno": 215,
   "epoch": 2,
   "frag": 0,
   "lastFrag": true,
   "time": "2025-02-21 09:11:29.0",
   "eventId": "mysql-bin.000015:0000000000072471;44",
   "sourceId": "db1",
   "comments": ""
 },
 {
   "seqno": 216,
   "epoch": 2,
   "frag": 0,
   "lastFrag": true,
   "time": "2025-02-21 09:12:04.0",
   "eventId": "mysql-bin.000015:0000000000072872;102",
   "sourceId": "db1",
   "comments": ""
 },
...
]
```

- `-sizes`

  Shows the size information for a given THL event, describing either the size of the SQL, or the number of rows within the given event. For example:

```
shell> thl list -sizes
SEQ#  Frag# Tstamp
...
10  0 2025-02-21 09:11:27.0   Event total:      1 chunks      24 bytes in SQL statements  0 rows
11  0 2025-02-21 09:11:28.0   Event total:      1 chunks      26 bytes in SQL statements  0 rows
12  0 2025-02-21 09:11:28.0   Event total:      1 chunks      24 bytes in SQL statements  0 rows
```

```
13  0 2025-02-21 09:11:28.0   Event total:      1 chunks      19 bytes in SQL statements  0 rows
14  0 2025-02-21 09:11:28.0   Event total:      1 chunks      24 bytes in SQL statements  0 rows
...
```

Summary information is also output identicating an overall count of the changes. For example:

```
Total ROW chunks: 203 with 224 updated rows (93%)
Total STATEMENT chunks: 14 with 1473 bytes (6%)
217 events processed
```

This information can be useful when viewing or monitoring the replication progress as it can help to indicate and identify the size of a specific transaction, particularly if the transaction is large. This can be particularly useful in combination with the `-first` [351] and/or `-last` [352].

For more detailed information on individual fragments within a sequence (and for large transactions there will be multiple fragments), use the thl list -sizesdetail command.

- `-sizesdetail`

  Shows detailed size information for a given THL event, describing either the size of the SQL, or the number of rows within the given event per fragment within each event, and with a summary for each event total. For very large THL event sizes this provide more detailed information about the size and makeup of the event. For example:

```
shell> thl list -sizesdetail -last
SEQ#  Frag# Tstamp      Chunks    SQL Data        Row Data
1604  0 2025-02-21 11:04:53.0 123 chunks  SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604  1 2025-02-21 11:04:53.0 123 chunks  SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604  2 2025-02-21 11:04:53.0 123 chunks  SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604  3 2025-02-21 11:04:53.0 123 chunks  SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604  4 2025-02-21 11:04:53.0 123 chunks  SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604  5 2025-02-21 11:04:53.0 123 chunks  SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604  6 2025-02-21 11:04:53.0 123 chunks  SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604  7 2025-02-21 11:04:53.0 123 chunks  SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604  8 2025-02-21 11:04:53.0 123 chunks  SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604  9 2025-02-21-29 11:04:53.0  123 chunks  SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604  10  2025-02-21 11:04:53.0 123 chunks  SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604  11  2025-02-21 11:04:53.0 7 chunks  SQL 0 bytes (0 avg bytes per chunk) Rows 2535 (362 avg rows per chunk)
     Event total: 1360 chunks  0 bytes in SQL statements 504498 rows
```

  Summary information is also output identicating an overall count of the changes. For example:

```
Total ROW chunks: 69487 with 18257671 updated rows (100%)
Total STATEMENT chunks: 0 with 0 bytes (0%)
628 events processed
```

  This information can be useful when viewing or monitoring the replication progress as it can help to indicate and identify the size of a specific transaction, particularly if the transaction is large. This can be particularly useful in combination with the `-first` [351] and/or `-last` [352].

- `-sizessummary`

  Outputs only the size summary information for the requested THL:

```
shell> thl list -sizessummary
Total ROW chunks: 69487 with 18257671 updated rows (100%)
Total STATEMENT chunks: 0 with 0 bytes (0%)
628 events processed
```

- `-specs`

  Shows the column specifications, such as identified type, length, and additional settings, when viewing events within row-based replication. This can be helpful when examining THL data in heterogeneous replication deployments.

  For example:

```
shell> thl list -seqno 216 -specs
SEQ# = 216 / FRAG# = 0 (last frag)
- FILE = thl.data.0000000001
- TIME = 2025-02-21 09:12:04.0
- EPOCH# = 2
- EVENTID = mysql-bin.000015:0000000000072872;102
- SOURCEID = db1
- METADATA = [mysql_server_id=10;mysql_thread_id=102;dbms_type=mysql;tz_aware=true;is_metadata=true;service=alpha;shard=tungsten_alpha;heartbeat=MASTER_
- GLOBAL OPTIONS = [autocommit=1;sql_auto_is_null=0;foreign_key_checks=1;unique_checks=1;sql_mode='NO_ENGINE_SUBSTITUTION,ONLY_FULL_GROUP_BY,STRICT_TRAN
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1, time_zone = '+00:00', ##charset = UTF-8]
- SQL(0) =
- ACTION = UPDATE
```

```
- SCHEMA = tungsten_alpha
- TABLE = heartbeat
- ROW# = 0
 - COL(index=1 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=) (size = 1) = 1
 - COL(index=2 name= type=0 [NULL] length=0 unsigned=false blob=false desc=) (size = 0) = NULL
 - COL(index=3 name= type=0 [NULL] length=0 unsigned=false blob=false desc=) (size = 0) = NULL
 - COL(index=4 name= type=93 [TIMESTAMP] length=0 unsigned=false blob=false desc=) (size = 21) = 2025-02-21 09:12:04.0
 - COL(index=5 name= type=0 [NULL] length=0 unsigned=false blob=false desc=) (size = 0) = NULL
 - COL(index=6 name= type=0 [NULL] length=0 unsigned=false blob=false desc=) (size = 0) = NULL
 - COL(index=7 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=) (size = 1) = 0
 - COL(index=8 name= type=12 [VARCHAR] length=0 unsigned=false blob=false desc=) (size = 13) = MASTER_ONLINE
 - KEY(index=1 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=) (size = 1) = 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.

9.20.6. thl purge Command

The purge command to the thl command deletes sequence number information from the THL files.

`thl` purge
[-low #] | [-from #] | [-high #] | [-to #]
[-y] [-no-checksum]

> **Warning**
>
> Purging all data requires that the THL information either be recreated from the source table, or reloaded from the Primary replicator.

The purge command deletes the THL data according to the following rules:

- Without any specification, a purge command will delete all of the stored THL information.
- When only `-high` (or `-to`) is specified, delete all the THL data up to and including the specified sequence number.
- When only `-low` (`-from`) 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 (or `-from` 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` option (or `-to` option) to specify the highest number to be removed (6021):

```
shell> thl purge -to 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
2025-02-20 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` option (or the `-from` option) to specify the sequence number:

```
shell> thl purge -from 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
2025-02-20 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` [354] ignores the checksum information on events in the event that the checksum is corrupt.

When purging, the THL files must be writeable; the replicator must either be offline or stopped when the purge operation is completed.

A purge operation may fail for the following reasons:

- Fatal error: The disk log is not writable and cannot be purged.

  The replicator is currently running and not in the `OFFLINE` state. Use trepctl offline to release the write lock n the THL files.

- Fatal error: Deletion range invalid; must include one or both log end points: low seqno=0 high seqno=1000

  An invalid sequence number or range was provided. The purge operation will refuse to purge events that do not exist in the THL files and do not match a valid file boundary, i.e. the low figure must match the start of one file and the high the end of a file. Use thl index to determine the valid ranges.

9.20.6.1. Purging THL Log Information on a Replica

Warning

Purging the THL on a Replica node can potentially remove information that has not yet been applied to the database. Please check and ensure that the THL data that you are purging has been applied to the database before continuing.

The THL files can be explicitly purged to recover disk space, but you should ensure that the currently applied sequence no to the database is not purged, and that additional hosts are not reading the THL information.

To purge the logs on a Replica node:

1. Determine the highest sequence number from the THL that you want to delete. To purge the logs up until the latest sequence number, you can use trepctl to determine the highest applied sequence number:

```
shell> trepctl services
Processing services command...
NAME              VALUE
----              -----
appliedLastSeqno: 3672
appliedLatency  : 331.0
role            : slave
serviceName     : alpha
serviceType     : local
started         : true
state           : ONLINE
Finished services command...
```

2. Shun the Replica datasource and put the replicator into the offline state using cctrl:

```
shell> cctrl
[LOGICAL] /alpha > datasource host1 shun
[LOGICAL] /alpha > replicator host1 offline
```

Important

NEVER Shun the Primary datasource!

3. Use the thl command to purge the logs up to the specified transaction sequence number. You will be prompted to confirm the operation:

```
shell> thl purge -high 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
2024-04-16 14:09:42,384 [ - main] INFO  thl.THLManagerCtrl Transactions deleted
```

4. Recover the host back into the cluster:

```
shell> cctrl
[LOGICAL] /alpha > datasource host1 recover
```

You can now check the current THL file information:

```
shell> thl index
LogIndexEntry thl.data.0000000024(3240:3672)
```

9.20.6.2. Purging THL Log Information on a Primary

Warning

Purging the THL on a Primary node can potentially remove information that has not yet been applied to the Replica databases. Please check and ensure that the THL data that you are purging has been applied to the database on all Replicas before continuing.

Important

If the situation allows, it may be better to switch the Primary role to a current, up-to-date Replica, then perform the steps to purge THL from a Replica on the old Primary host using Section 9.20.6.1, “Purging THL Log Information on a Replica”.

Warning

Follow the below steps with great caution! Failure to follow best practices will result in Replicas unable to apply transactions, forcing a full re-provisioning. For those steps, please see Section 6.6.1.2, “Provision or Reprovision a Replica”.

The THL files can be explicitly purged to recover disk space, but you should ensure that the currently applied sequence no to the database is not purged, and that additional hosts are not reading the THL information.

To purge the logs on a Primary node:

1. Determine the highest sequence number from the THL that you want to delete. To purge the logs up until the latest sequence number, you can use trepctl to determine the highest applied sequence number:

```
shell> trepctl services
Processing services command...
NAME              VALUE
----              -----
appliedLastSeqno: 3675
appliedLatency  : 0.835
role            : master
serviceName     : alpha
serviceType     : local
started         : true
state           : ONLINE
Finished services command...
```

2. Set the cluster to *MAINTENANCE* mode and put the replicator into the offline state using cctrl:

```
shell> cctrl
[LOGICAL] /alpha > set policy maintenance
[LOGICAL] /alpha > replicator host1 offline
```

3. Use the thl command to purge the logs up to the specified transaction sequence number. You will be prompted to confirm the operation:

```
shell> thl purge -high 3670
WARNING: The purge command will break replication if you delete all events or »
   delete events that have not reached all slaves.
Are you sure you wish to delete these events [y/N]?
y
Deleting events where SEQ# <=3670
2013-04-16 14:09:42,384 [ - main] INFO  thl.THLManagerCtrl Transactions deleted
```

4. Set the cluster to *AUTOMATIC* mode and put the replicator online using cctrl:

```
shell> cctrl
[LOGICAL] /alpha > replicator host1 online
[LOGICAL] /alpha > set policy automatic
```

You can now check the current THL file information:

```
shell> thl index
LogIndexEntry thl.data.0000000024(3240:3672)
```

9.20.7. thl tail Command

Note

The thl tail command was introduced in verion 7.0.3

The thl tail command can be used to view a live stream of THL as it is either being generated by an extractor, or received on a replica from an upstream extractor.

On its own, the command will output the THL entry, as shown in the following example:

```
shell> thl tail
SEQ# = 219 / FRAG# = 0 (last frag)
- FILE = thl.data.0000000001
- TIME = 2025-02-21 09:48:13.0
- EPOCH# = 2
- EVENTID = mysql-bin.000015:0000000000080957;44
- SOURCEID = db1
- METADATA = [mysql_server_id=10;mysql_thread_id=44;dbms_type=mysql;tz_aware=true;service=alpha;shard=hr]
- GLOBAL OPTIONS = [autocommit=1;sql_auto_is_null=0;foreign_key_checks=1;unique_checks=1;sql_mode='NO_ENGINE_SUBSTITUTION,ONLY_FULL_GROUP_BY,STRICT_TRANS_
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
```

```
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1, time_zone = '+00:00', ##charset = UTF-8]
- SQL(0) =
- ACTION = INSERT
- SCHEMA = hr
- TABLE = employees
- ROW# = 0
 - COL(1: ) (size = 3) = 206
 - COL(2: ) (size = 7) = William
 - COL(3: ) (size = 5) = Gietz
 - COL(4: ) (size = 6) = WGIETZ
 - COL(5: ) (size = 12) = 515.123.8181
 - COL(6: ) (size = 10) = 2002-06-07
 - COL(7: ) (size = 10) = AC_ACCOUNT
 - COL(8: ) (size = 7) = 8300.00
 - COL(9: ) (size = 0) = NULL
 - COL(10: ) (size = 3) = 205
 - COL(11: ) (size = 3) = 110
```

With the addition of the `-sql` switch, the command will output just the SQL statements, as shown in the following example:

```
shell> thl tail -sql
/* seqno: 11, fragno: 0 (last), epoch: 7, 2025-02-02 14:07:18.0, mysql-bin.072822:0000000000033589;65, centos1 */
/* SEQ# = 11 - SQL rendering of row change events is not supported */

/* seqno: 12, fragno: 0 (last), epoch: 7, 2025-02-02 14:07:57.0, mysql-bin.072822:0000000000034044;67, centos1 */
USE tungsten_alpha;
UPDATE tungsten_alpha.heartbeat SET source_tstamp= '2025-02-02 15:07:57.108', salt= 4, name= 'NEW5'  WHERE id= 1;
```

> **Note**
>
> The rendering of sql events for use with the `-sql` is currently not supported against ROW based events, and will result in the error shown in the example above

In both instances, output to the screen will continue until CTRL+C is pressed.

9.20.8. thl help Command

The help command to the thl command outputs the current help message text.

9.21. The tmonitor Command

The tmonitor command was added in version 6.1.13

tmonitor is a tool for the management and testing of external Prometheus exporters (node and mysqld), and for the testing of internal exporters (Manager, Connector and Replicator).

The tmonitor command is located in the `$CONTINUENT_ROOT/tungsten/cluster-home/bin` directory.

> **Note**
>
> The tmonitor command will only be available in the `PATH` if the Tungsten software has been installed with the configuration option `profile-script` [507] included.

See the table below for a list of valid arguments:

Table 9.45. tmonitor Common Options

Option	Description
`--connector`, `-C`	Specify the Tungsten Connector exporter (test and Tungsten Clustering only)
`--connectorPort`	Set the port to access the Connector exporter (default: 8093)
`--external`, `-e`, `-x`	Operate only upon External-Scope exporters
`--help`, `-h`	Display help text
`--internal`, `-i`	Operate only upon Internal-Scope exporters
`--manager`, `-M`	Specify the Tungsten Manager exporter (test and Tungsten Clustering only)
`--managerPort`	Set the port to access the Manager exporter (default: 8092)
`--mysql`, `--mysqld`, `-m`	Specify the MySQL exporter

Option	Description
`--node`, `-n`	Specify the node exporter
`--replicator`, `-R`	Specify the Tungsten Replicator exporter (test only)
`--replicatorPort`	Set the port to access the Replicator exporter (default: 8091)
`-t`, `-T`	Issue '-t test' to show Tungsten-specific metric lines. Issue '-T test' to show meetric help and headers
`--verbose`, `-v`	

Exporters that require an external binary to function (i.e. node_exporter and mysqld_exporter) are considered to have an External Scope.

Exporters that do not require an external binary to function (i.e. manager, replicator and connector) are considered to have an Internal Scope.

By default, tmonitor {action} will act upon all available exporters.

If any exporter is specified on the CLI, then only those listed on the CLI will be acted upon.

The curl command must be available in the PATH for the `status` and `test` actions to function.

tmonitor status will use curl to test the exporter on `localhost`.

tmonitor test will use curl to fetch and print the metrics from one or more exporters on `localhost`.

tmonitor install will configure the specified exporter (or all external exporters if none is specified) to start at boot.

tmonitor remove will stop the specified exporter (or all external exporters if none is specified) from starting at boot.

Both the `install` and `remove` actions will attempt to auto-detect the boot sub-system. Currently, `init.d` and `systemd` are supported.

```
shell> tmonitor help
...
>>> Usage:

   tmonitor [args] {action}

   = Actions available for all exporter scopes:

      status - validate the service via curl (short output)
      test - validate the service via curl (full output)

   = Actions available for External-scope exporters only:

      start - launch the exporter process
      stop - kill the exporter process

      install - configure the exporter to start at boot
      remove - stop the exporter from starting at boot


>>> Arguments:

   [-h|--help]
   [-v|--verbose]

   [-i|--internal] Only act upon exporters with an internal scope
   [-e|-x|--external] Only act upon exporters with an external scope

   --internal and --external may not be specified together.

   = Internal Scope Exporters:

     [-C|--connector] Specify the Tungsten Connector exporter
     [-M|--manager] Specify the Tungsten Manager exporter
     [-R--replicator] Specify the Tungsten Replicator exporter

   = External Scope Exporters:

     [-m|--mysql|--mysqld] Specify the MySQL exporter
     [-n|--node] Specify the Node exporter
```

Example: View the status of all exporters:

```
shell> tmonitor status
Tungsten Connector exporter running ok
Tungsten Manager exporter running ok
MySQL exporter running ok
```

```
Node exporter running ok
Tungsten Replicator exporter running ok
All exporters running ok (Up: Tungsten Connector, Tungsten Manager, MySQL, Node, Tungsten Replicator)
```

Example: Start all exporters:

```
shell> tmonitor start
tungsten@db1-demo:/home/tungsten # tmonitor start
Node exporter started successfully on port 9100.
MySQL exporter started successfully on port 9104.

shell> tmonitor start
The Node exporter is already running
The MySQL exporter is already running
```

Example: Test all exporters:

```
shell> tmonitor test | wc -l
3097

shell> tmonitor test | grep '== '
== Metrics for the connector exporter:
== Metrics for the manager exporter:
== Metrics for the mysql exporter:
== Metrics for the node exporter:
== Metrics for the replicator exporter:

shell> tmonitor test | less
```

Example: Stop all exporters:

```
shell> tmonitor stop
All exporters stopped
```

Example: View the status of all exporters filtered by scope:

```
shell> tmonitor status -i
Tungsten Connector exporter running ok
Tungsten Manager exporter running ok
Tungsten Replicator exporter running ok
All internal exporters running ok (Up: Tungsten Connector, Tungsten Manager, Tungsten Replicator)

shell> tmonitor status -x
MySQL exporter running ok
Node exporter running ok
All external exporters running ok (Up: MySQL, Node)
```

Example: Install init.d boot scripts for all external exporters:

```
shell> tmonitor install
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:

sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install


shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install
node_exporter init.d boot script installed and activated

Use either `sudo service node_exporter start` or `tmonitor --node start` now to start the Node exporter.

mysqld_exporter init.d boot script installed and activated

Use either `sudo service mysqld_exporter start` or `tmonitor --mysql start` now to start the MySQL exporter.
```

Example: Install systemd boot scripts for all external exporters:

```
shell> tmonitor install
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:

sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install


shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install
Created symlink from /etc/systemd/system/multi-user.target.wants/node_exporter.service to /etc/systemd/system/node_exporter.service.
node_exporter systemd boot script installed and enabled

Use either `sudo systemctl start node_exporter` or `tmonitor --node start` now to start the Node exporter.
```

```
Created symlink from /etc/systemd/system/multi-user.target.wants/mysqld_exporter.service to /etc/systemd/system/mysqld_exporter.service.
mysqld_exporter systemd boot script installed and enabled

Use either `sudo systemctl start mysqld_exporter` or `tmonitor --mysql start` now to start the MySQL exporter.
```

Example: Remove init.d boot scripts for all external exporters:

```
shell> tmonitor remove
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:

sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove

shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove
node_exporter is still running, unable to remove. Please run either `tmonitor --node stop` or `sudo service node_exporter stop`, then retry this operation

shell> tmonitor stop
Node exporter stopped
MySQL exporter stopped

shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove
node_exporter init.d boot script de-activated and removed
mysqld_exporter init.d boot script de-activated and removed
```

Example: Remove systemd boot scripts for all external exporters:

```
shell> tmonitor stop
Node exporter stopped
MySQL exporter stopped

shell> tmonitor remove
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:

sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove

shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove
node_exporter systemd unit boot script disabled and removed
mysqld_exporter systemd unit boot script disabled and removed
```

9.22. The tpasswd Command

Table 9.46. tpasswd Common Options

Option	Description
`--create`, `-c`	Creates a new user/password
`--delete`, `-d`	Delete a user/password combination
`-e`, `--encrypted.password`	Encrypt the password
`--file`, `-f`	Specify the location of the security.properties file
`-help`, `-h`	Display help text
`-p`, `--password.file.location`	Specify the password file location
`--target`, `-t`	Specify the target application
`-ts`, `--truststore.location`	

Using the --compare.to option

This `--compare.to` switch allows for comparing currently used `passwords.store` file to a given one. If the files contain identical users and passwords, it will display `true` and return 0. If not, it will display `false` and return error code 2. Example:

```
shell> Given passwords file identical to current one: true
```

9.23. The tprovision Script

> tprovision was previously known as tungsten_provision_slave and was renamed in v7.0.0
>
> For instructions for releases older than this, please refer to the documentation here

The tprovision script allows you to easily provision, or reprovision, a database server using information from a remote host.

`tprovision [ -c, --create-master ] [ --help, -h ] [ --method, -m ] [ --port, -p ] [ --seed ] [ --source, -s ] [ --threads, -t ]`

Where:

Table 9.47. tprovision Command-line Options

Option	Description
`-c, --create-master`	Use to flag to the script that the node being provisioned needs to be a Primary. Valid for Composite Active/Active only. Forces the provision of a failed Primary and will reset services.
`--help, -h`	Show help text
`--method, -m`	Backup method to use. Valid methods are mysqldump, xtrabackup, rsync (v7.0.2+), and mysqlclone (v7.1.0+).
`--port, -p`	Port to use to connect to MySQL when using mysqldump, or ssh port when using xtrabackup.
`--seed`	Just do a single non-locking pass when using rsync.
`--source, -s`	Server to use as a source for the backup
`--threads, -t`	Number of parallel threads to use for xtrabackup. Increasing this number on large databases may improve backup speeds. If not supplied, default will be based on the default for the revision of xtrabackup in use.

Important

It is recommend to run this script in a utility such as screen in case the terminal gets disconnected.

For both mysqldump and xtrabackup methods, the script will perform a streaming backup from the source node to the target node.

The script will automatically put all replication services offline prior to beginning. If the services were online, the script will put them back online following a successful completion. All THL logs will be cleared prior to going online. The replicator will start replication from the position reflected on the source host.

Provisioning will fail from a Replica that is stopped, or if the Replica is not in the `ONLINE` state.

Using xtrabackup

The script will run validation prior to starting to make sure the needed scripts are available. The provision process will run Xtrabackup on the source server and stream the contents to the server you are provisioning. After taking the backup, the script will prepare the directory and restart the MySQL server

Using mysqldump

The script will run mysqldump by default.

Using rsync

Note

Provisioning using rsync was introduced in v7.0.2

The script will copy the source database to the target using rsync, using two passes. The first pass will simply copy the database live. This will produce an inconsistent database on the target but will have seeded the target database.

Pass 2 will quiesce the source database and do a final rsync. Because pass 1 had already seeded the data, pass 2 should be quick, minimizing the downtime on the source database.

To only run the first pass, use the `--seed` option. This is useful to seed the target in advance and can be done multiple times to get the target seeded ahead of time to reduce downtime during a maintenance window.

Using rsync is not recommended when the source database is a primary, due to the database being quiesced and thus potentially causing downtime. The script normally will not allow you to do this, however you can override this check with the option `--i-am-sure` and force a provision from a primary database.

Compatibility

The script only works with MySQL at this time.

Logging

The script will log output to the `/opt/continuent/service_logs` directory.

Example

To reprovision the Replica db3 from another Replica, db2. Using xtrabackup

```
db3-shell> tprovision --source db2 --method xtrabackup
```

To reprovision the Replica db3 from another Replica, db2. Using the default, mysqldump

```
db3-shell> tprovision --source db2
```

To reprovision the Primary db1 from the Primary, db4, in the remote cluster. Using Xtrabackup. This is only applicable to Composite Active/Active topologies

```
db1-shell> tprovision --source db4 -c --method xtrabackup
```

To reprovision the replica db3 from another replica, db2, using rsync and only seeding the database on db3

```
db3-shell> tprovision --source db2 --method rsync --seed
```

After seeding [above], run the rsync again. The amount of time the database is locked now should be much less since we've already seeded the database changes. We will wait 10 seconds to acquire the lock:

```
db3-shell> tprovision --source db2 --method rsync --sleep-after-flush 10
```

9.24. The trepctl Command

The trepctl command provides the main status and management interface to Tungsten Replicator. The trepctl command is responsible for:

- Putting the replicator online or offline
- Pause a specific stage within the replicator
- Performing backup and restore operations
- Skipping events in the THL in the event of an issue
- Getting status and active configuration information

The operation and control of the command is defined through a series of command-line options which specify general options, replicator wide commands, and service specific commands that provide status and control over specific services.

The trepctl command by default operates on the current host and configured service. For installations where there are multiple services and hosts in the deployment. Explicit selection of services and hosts is handled through the use of command-line options, for more information see Section 9.24.1, “trepctl Options”.

For individual operations, trepctl uses a sub-command structure on the command-line that specifies which operation is to be performed. There are two classifications of commands, global commands, which operate across all replicator services, and service-specific commands that perform operations on a specific service and/or host. For information on the global commands available, see Section 9.24.2, “trepctl Global Commands”. Information on individual commands can be found in Section 9.24.3, “trepctl Service Commands”.

9.24.1. trepctl Options

Table 9.48. trepctl Command-line Options

Option	Description
`-all-services`	Execute command against all available services.
`-host`	Host name of the replicator. Defaults to localhost if not supplied.
`-port`	RMI Port number of the replicator. Defaults to 10000
`-retry`	Number of times to retry the connection. Pass integer value.
`-service`	Name of the replicator service. Mandatory of more than one service available.
`-verbose`	Enable verbose messages for operations

Global command-line options enable you to select specific hosts and services. If available, trepctl will read the active configuration to determining the host, service, and port information. If this is unavailable or inaccessible, the following rules are used to determine which host or service to operate upon:

- If no host is specified, then trepctl defaults to the host on which the command is being executed.
- If no service is specified:
  - If only one service has been configured, then trepctl defaults to showing information for the configured service.
  - If multiple services are configured, then trepctl returns an error, and requests a specific service be selected.

To use the global options:

- `-host`

  Specify the host for the operation. The replicator service must be running on the remote host for this operation to work.

- `-port`

  Specify the base TCP/IP port used for administration. The default is port 10000; port 10001 is also used. When using different ports, `port` and `port+1` is used, i.e. if port 4996 is specified, then port 4997 will be used as well. When multiple replicators are installed on the same host, different numbers may be used.

- `-service`

  The servicename to be used for the requested status or control operation. When multiple services have been configured, the servicename must be specified, otherwise the operation will fail and you will be presented with examples of commands to use, for example:

```
shell> trepctl status
Processing status command...
Operation failed: You must specify a service name with the -service flag because there is more than one
service available. The currently available status commands are:
trepctl -service north status
trepctl -service north_from_east status
trepctl -service north_from_west status
```

- `-all-services`

  When multiple services are available, you can perform actions against all services at the same time, for example"

```
shell> trepctl -all-service offline
```

- `-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.

9.24.2. 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 9.49. trepctl Replicator Wide Commands

Option	Description
`kill`	Shutdown the replication services immediately
`services`	List the configured replicator services
`servicetable`	List all the currently configures services in a tabular format
`thl`	Interact with dynamic THL options
`version`	Show the replicator version number and build

These commands can be executed on the current or a specified host. Because these commands operate for replicators irrespective of the service configuration, selecting or specifying a service is note required.

9.24.2.1. trepctl kill Command

The trepctl kill command terminates the replicator without performing any cleanup of the replicator service, THL or sequence number information stored in the database. Using this option may cause problems when the replicator service is restarted.

`trepctl` kill [`-y`]

When executed, trepctl will ask for confirmation:

```
shell> trepctl kill
Do you really want to kill the replicator process? [yes/NO]
```

The default is no. To kill the service, ignoring the interactive check, use the `-y` option:

```
shell> trepctl kill -y
Sending kill command to replicator
Replicator appears to be stopped
```

9.24.2.2. 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 [`-c`] [`-full`] [`-json`] [`-r`]

For example:

```
shell> trepctl services
Processing services command...
NAME              VALUE
----              -----
appliedLastSeqno: 2541
appliedLatency  : 0.48
role            : master
serviceName     : alpha
serviceType     : local
started         : true
state           : ONLINE
Finished services command...
```

For more information on the fields displayed, see Section E.2, “Generated Field Reference”.

For a replicator with multiple services, the information is output for each configured service:

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

The information can be reported in JSON format by using the `-json` option to the command:

```
shell> trepctl services -json
```

```
[
{
  "serviceType" : "local",
  "appliedLatency" : "0.48",
  "serviceName" : "alpha",
  "appliedLastSeqno" : "2541",
  "started" : "true",
  "role" : "master",
  "state" : "ONLINE"
}
]
```

The information is output as an array of objects, one object for each service identified.

If the `-full` option is added, the output includes full details of the service. This is the same as running trepctl status or, for hosts with multiple services, trepctl -all-services status.

The `-full` can also be combined with the `-json` option, for example:

```
shell> trepctl services -json -full
[
{
"serviceType":"local",
"sourceId":"db1",
"appliedLastGoodSeqno":"221",
"role":"master",
"pendingErrorEventId":"NONE",
"timezone":"GMT","siteName":"default"
,"useSSLConnection":"false",
"pipelineSource":"\/var\/lib\/mysql",
"uptimeSeconds":"298.328",
"resourceJdbcDriver":"org.drizzle.jdbc.DrizzleDriver",
"appliedLastEventId":"mysql-bin.000016:0000000000000959;3309",
"seqnoType":"java.lang.Long",
"pendingExceptionMessage":"NONE",
"resourceVendor":"mysql",
"appliedLatency":"0.448",
"clusterName":"alpha",
"host":"db1",
"state":"ONLINE",
"rmiPort":"10000",
"resourceJdbcUrl":"jdbc:mysql:thin:\/\/db1:13306\/${DBNAME}?jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull&tinyInt1isBit=false&allowMult
"simpleServiceName":"alpha",
"pendingErrorCode":"NONE",
"masterConnectUri":"thl:\/\/localhost:\/",
"pendingErrorSeqno":"-1",
"latestEpochNumber":"221",
"started":"true",
"offlineRequests":"NONE",
"timeInStateSeconds":"2.395",
"minimumStoredSeqNo":"0",
"serviceName":"alpha",
"currentEventId":"mysql-bin.000016:959",
"autoRecoveryTotal":"0",
"version":"Tungsten Clustering 7.0.3 build 141",
"masterListenUri":"thl:\/\/db1:2112\/",
"transitioningTo":"",
"appliedLastSeqno":"221",
"dataServerHost":"db1",
"appliedLastGoodLatency":"0.448",
"relativeLatency":"2.785",
"extensions":"",
"channels":"1",
"currentTimeMillis":"1740496159786",
"maximumStoredSeqNo":"221",
"pendingError":"NONE",
"autoRecoveryEnabled":"false"
}
]
```

The information can be refreshed automatically, and continuously, using the `-r` option. Pass an integer value representing number of seconds between each refresh. To quit, press CTRL+C.

To only refresh for a specific number of iterations, the `-c` option can be include, for example, the following command will refresh every 2 seconds, for a total of 10 iterations:

```
shell> trepctl services -r 2 -c 10
Processing services command...
NAME              VALUE
----              -----
appliedLastSeqno: 2541
appliedLatency  : 0.48
```

```
role            : master
serviceName     : alpha
serviceType     : local
started         : true
state           : ONLINE
Finished services command...
```

9.24.2.3. trepctl servicetable Command

The trepctl servicetable command outputs a list of all the current services and current status information in a tabular format to make it easy to determine multi-service installations.

`trepctl` servicetable [`-c`] [`-r`]

For example:

```
shell> trepctl servicetable
Processing servicetable command...
Service              | Status                       | Role       | MasterConnectUri              |      SeqNo |    Latency
-------------------- | ---------------------------- | ---------- | ----------------------------- | ---------- | ----------
alpha                | ONLINE                       | slave      | thl://trfiltera:2112/         |        322 |       0.00
beta                 | ONLINE                       | slave      | thl://ubuntuheterosrc:2112/   |         12 |    4658.59
Finished servicetable command...
```

The information can be refreshed automatically, and continuously, using the `-r` option. Pass an integer value representing number of seconds between each refresh. To quit, press CTRL+C.

To only refresh for a specific number of iterations, the `-c` option can be include, for example, the following command will refresh every 5 seconds, for a total of 10 iterations:

```
shell> trepctl servicetable -r 5 -c 10
```

For more information on the fields displayed, see Section E.2, “Generated Field Reference”.

9.24.2.4. trepctl thl Command

The trepctl thl command is used to dynamically enable or disable on-disk encryption and compression of THL.

These options can also be set via tpm options. For more information see Section 6.18.4, “THL Encryption and Compression”

`trepctl` thl [`-compression`] [`-encryption`]

`-compression` [369]

Pass `enable` or `disable` to the `-compression` [369] option to enable or disable THL compression accordingly.

Note that the replicators must be offline to change this option and the same enable|diable option must be applied to all hosts before bringing replication online:

First, place the cluster into *`MAINTENANCE`*:

```
shell> cctrl
cctrl> set poicy maintenance
```

On every host, place the services offline, and enable (or disable) compression:

```
shell> trepctl [-service <servicename>] offline
shell> trepctl [-service <servicename>] thl -compression [enable|disable]
```

When complete, return the cluster to *`AUTOMATIC`* which will return the replicators to online:

```
shell> cctrl
cctrl> set policy automatic
```

`-encryption` [369]

Pass `enable` or `disable` to the `-encryption` [369] option to enable or disable THL encryption accordingly.

Note that the replicators must be offline to change this option and the same enable|diable option must be applied to all hosts before bringing replication online:

First, place the cluster into *`MAINTENANCE`*:

```
shell> cctrl
```

```
cctrl> set poicy maintenance
```

On every host, place the services offline, and enable (or disable) encryption:

```
shell> trepctl [-service <servicename>] offline
shell> trepctl [-service <servicename>] thl -encryption [enable|disable]
```

When complete, return the cluster to *AUTOMATIC* which will return the replicators to online:

```
shell> cctrl
cctrl> set poicy automatic
```

Pass genkey [370] to the -encryption [369] option to generate a new encryption key. Unlike enabling/disbaling encrpytion, to change generate a new encryption key, the replication service must be online.

```
shell> trepctl [-service <servicename>] online
shell> trepctl [-service <servicename>] thl -encryption genkey
```

Important

The genkey [370] should ONLY be executed on the primary node.

9.24.2.5. trepctl version Command

The trepctl version command outputs the version number of the specified replicator service.

trepctl version

```
shell> trepctl version
Tungsten Replicator 7.0.3 build 141
```

The system can also be used to obtain remote version:

```
shell> trepctl -host host2 version
Tungsten Replicator 7.0.3 build 141
```

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.

9.24.3. 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 9.50. trepctl Service Commands

Option	Description
backup	Backup database
capabilities	List the configured replicator capabilities
check	Generate consistency check
clear	Clear one or all dynamic variables
clients	List clients connected to this replicator
error	Will output full stack trace of the last error, if any. If no error, response will be empty.
flush	Synchronize transaction history log to database
heartbeat	Insert a heartbeat event with optional name
load	Load the replication service
offline	Set replicator to OFFLINE state
offline-deferred	Set replicator OFFLINE at a future point in the replication stream
online	Set Replicator to ONLINE with start and stop points

Option	Description
pause	Pause the replicator. Specify the stage using the -stage option and optional time using -time
perf	Print detailed performance information
properties	Display a list of all internal properties
purge	Purge non-Tungsten logins on database
qs	Print a simplified quick replicator status
reset	Deletes the replicator service
restore	Restore database on specified host
resume	Resume a paused replicator. Specify the stage using the -stage option.
setdynamic	Set dynamic properties
setrole	Set replicator role
shard	List, add, update, and delete shards
status	Print replicator status information
unload	Unload the replication service
wait	Wait for the replicator to reach a specific state, time or applied sequence number

The following sections detail each command individually, with specific options, operations and information.

9.24.3.1. trepctl backup Command

The trepctl backup command performs a backup of the corresponding database for the selected service.

`trepctl` backup [`-backup agent`] [`-limit s`] [`-storage agent`]

Where:

Table 9.51. trepctl backup Command Options

Option	Description
`-backup agent` [371]	Select the backup agent
`-limit s` [371]	The period to wait before returning after the backup request
`-storage agent` [371]	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 9.24.3.19, "trepctl restore Command". Different backup solutions may require that the replicator be placed into the `OFFLINE` state before the backup is performed.

A log of the backup operation will be stored in the replicator log directory, in a file corresponding to the backup tool used (e.g. `mysqldump.log`).

If multiple backup agents have been configured, the backup agent can be selected on the command-line:

```
shell> trepctl backup -backup mysqldump
```

If multiple storage agents have been configured, the storage agent can be selected using the `-storage` [371] 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` [371] 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` [371] 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_2025-02-25_18-14_11.sql.gz length=0
... backup.FileSystemStorageAgent Stored backup storage properties: »
file=/opt/continuent/backups/store-0000000003.properties length=314
... backup.BackupTask Backup completed normally: »
uri=storage://file-system/store-0000000003.propertiess
```

The URI can be used during a restore.

9.24.3.2. trepctl capabilities Command

The `capabilities` command outputs a list of the supported capabilities for this replicator instance.

`trepctl` capabilities

The information output will depend on the configuration and current role of the replicator service. Different services on the same host may have different capabilities. For example:

```
shell> trepctl capabilities
Replicator Capabilities
Roles:             [master, slave]
Replication Model: push
Consistency Check: true
Heartbeat:         true
Flush:             true
```

The fields output are as follows:

- *`Roles`*

  Indicates whether the replicator can be a *`master`* or *`slave`*, or both.

- *`Replication Model`*

  The model used by the replication system. The default model for MySQL for example is push, where information is extracted from the binary log and pushed to Replicas that apply the transactions. The pull model is used for heterogeneous deployments.

- *`Consistency Check`*

  Indicates whether the internal consistency check is supported. For more information see Section 9.24.3.3, “trepctl check Command”.

- *`Heartbeat`*

  Indicates whether the heartbeat service is supported. For more information see Section 9.24.3.8, “trepctl heartbeat Command”.

- *`Flush`*

  Indicates whether the trepctl flush operation is supported.

9.24.3.3. trepctl check Command

The `check` command operates by running a CRC check on the schema or table specified, creating a temporary table containing the check data and values during the process. The data collected during this process is then written to a consistency table within the replication configuration schema and is used to verify the table data consistency on the Primary and the Replica.

Warning

Because the check operation is creating a temporary table containing a CRC of each row within the specified schema or specific table, the size of the temporary table created can be quite large as it consists of CRC and row

count information for each row of each table (within the specified row limits). The configured directory used by MySQL for temporary table creation will need a suitable amount of space to hold the temporary data.

9.24.3.4. trepctl clear Command

The trepctl clear command deletes any dynamic properties configured within the replicator service.

`trepctl` clear

Dynamic properties include the current active role for the service. The dynamic information is stored internally within the replicator, and also stored within a properties file on disk so that the replicator can be restarted.

For example, the replicator role may be temporarily changed to receive information from a different host or to act as a Primary in place of a Replica. The replicator can be returned to the initial configuration for the service by clearing this dynamic property:

```
shell> trepctl clear
```

9.24.3.5. trepctl clients Command

Outputs a list of the clients that have been connected to the Primary service since it went online. If a Replica service goes offline or is stopped, it will still be reported by this command.

`trepctl` clients [`-json`]

Where:

Table 9.52. trepctl clients Command Options

Option	Description
`-json` [373]	Output the information as JSON

The command outputs the list of clients and the management port on which they can be reached:

```
shell> trepctl clients
Processing clients command...
host4:10000
host2:10000
host3:10000
Finished clients command...
```

A JSON version of the output is available when using the `-json` [373] option:

```
shell> trepctl clients -json
[
{
"rmiPort": "10000",
"rmiHost": "host4"
},
{
"rmiPort": "10000",
"rmiHost": "host2"
},
{
"rmiPort": "10000",
"rmiHost": "host3"
}
]
```

The information is divided first by host, and then by the RMI management port.

9.24.3.6. trepctl error Command

The trepctl error command will output a full stack trace of the last occuring error, if any,

An empty response will be returned in the event of there being no error

`trepctl` error

```
shell> trepctl -service <serviceName> error

Event application failed: seqno=10 fragno=0 message=Table hr.regions not found in database. Unable to generate a valid statement.
com.continuent.tungsten.replicator.applier.ApplierException: Table hr.regions not found in database. Unable to generate a valid statement.
at com.continuent.tungsten.replicator.applier.JdbcApplier.getTableMetadata(JdbcApplier.java:582)
at com.continuent.tungsten.replicator.applier.JdbcApplier.fillColumnNames(JdbcApplier.java:494)
```

```
at com.continuent.tungsten.replicator.applier.JdbcApplier.getColumnInformation(JdbcApplier.java:1236)
at com.continuent.tungsten.replicator.applier.MySQLApplier.applyOneRowChangePrepared(MySQLApplier.java:418)
at com.continuent.tungsten.replicator.applier.JdbcApplier.applyRowChangeData(JdbcApplier.java:1460)
at com.continuent.tungsten.replicator.applier.JdbcApplier.apply(JdbcApplier.java:1576)
at com.continuent.tungsten.replicator.applier.ApplierWrapper.apply(ApplierWrapper.java:100)
at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.apply(SingleThreadStageTask.java:871)
at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.runTask(SingleThreadStageTask.java:601)
at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.run(SingleThreadStageTask.java:185)
at java.base/java.lang.Thread.run(Thread.java:834)
```

9.24.3.7. trepctl flush Command

On a Primary, the trepctl flush command synchronizes the database with the transaction history log, flushing the in memory queue to the THL file on disk. The operation is not supported on a Replica.

`trepctl` flush [`-limit s`]

Internally, the operation works by inserting a heartbeat event into the queue, and then confirming when the heartbeat event has been committed to disk.

To flush the replicator:

```
shell> trepctl flush
Master log is synchronized with database at log sequence number: 3622
```

The flush operation is always initiated, and by default trepctl will wait until the operation completes. Using the `-limit` option, the amount of time the command-line waits before returning can be specified:

```
shell> trepctl flush -limit 1
```

9.24.3.8. trepctl heartbeat Command

Inserts a heartbeat into the replication stream, which can be used to identify replication points.

`trepctl` heartbeat [`-name`] [`-tz s`]

The heartbeat system is a way of inserting an identifiable event into the THL that is independent of the data being replicated. This can be useful when performing different operations on the data where specific checkpoints must be identified.

To insert a standard heartbeat:

```
shell> trepctl heartbeat
```

When performing specific operations, the heartbeat can be given an name:

```
shell> trepctl heartbeat -name dataload
```

Heartbeats insert a transaction into the THL using the transaction metadata and can be used to identify whether replication is operating between replicator hosts by checking that the sequence number has been replicated to the Replica. Because a new transaction is inserted, the sequence number is increased, and this can be used to identify if transactions are being replicated to the Replica without requiring changes to the database. To check replication using the heartbeat:

1. Check the current transaction sequence number on the Primary:

   ```
   shell> trepctl status
   Processing status command...
   NAME                     VALUE
   ----                     -----
   appliedLastEventId     : mysql-bin.000009:0000000000008998;0
   appliedLastSeqno       : 3630
   ...
   ```

2. Insert a heartbeat event:

   ```
   shell> trepctl heartbeat
   ```

3. Check the sequence number again:

   ```
   shell> trepctl status
   Processing status command...
   NAME                     VALUE
   ----                     -----
   appliedLastEventId     : mysql-bin.000009:0000000000009310;0
   appliedLastSeqno       : 3631
   ```

4. Check that the sequence number on the Replica matches:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000009:0000000000009310;0
appliedLastSeqno       : 3631
```

Heartbeats are given implied names, but can be created with explicit names that can be tracked during specific events and operations.

For example, when loading a specific set of data, the information may be loaded and then a backup executed on the Replica before enabling standard replication. This can be achieved by configuring the Replica to go offline when a specific heartbeat event is seen, loading the data on the Primary, inserting the heartbeat when the load has finished, and then performing the Replica backup:

1. On the Replica:

   ```
   Replica shell> trepctl offline-deferred -at-heartbeat dataload
   ```

   The trepctl offline-deferred configures the Replica to continue in the online state until the specified event, in this case the heartbeat, is received. The deferred state can be checked by looking at the status output, and the *offlineRequests* field:

   ```
   Processing status command...
   NAME                     VALUE
   ----                     -----
   appliedLastEventId     : mysql-bin.000009:0000000000008271;0
   appliedLastSeqno       : 3627
   appliedLatency         : 0.704
   ...
   offlineRequests        : Offline at heartbeat event: dataload
   ```

2. On the Primary:

   ```
   Primary shell> mysql newdb < newdb.load
   ```

3. Once the data load has completed, insert the heartbeat on the Primary:

   ```
   Primary shell> trepctl heartbeat -name dataload
   ```

   The heartbeat will appear in the transaction history log after the data has been loaded and will identify the end of the load.

4. When the heartbeat is received, the Replica will go into the offline state. Now a backup can be created with all of the loaded data replicated from the Primary. Because the Replica is in the offline state, no further data or changes will be recorded on the Replica

This method of identifying specific events and points within the transaction history log can be used for a variety of different purposes where the point within the replication stream without relying on the arbitrary event or sequence number.

9.24.3.8.1. trepctl heartbeat Time Zone Handling

When the Replicator inserts a heartbeart there is an associated timezone. The default uses the Replicator host's timezone.

The `-tz` option to the trepctl heartbeat command may be used to force the use of a specific timezone.

For example, use GMT as the timezone when inserting a heartbeat:

```
shell> trepctl heartbeat -tz NONE
```

Use the Replicator host's timezone to insert the heartbeat:

```
shell> trepctl heartbeat -tz HOST
```

Use the given timezone to insert the heartbeat:

```
shell> trepctl heartbeat -tz {valid timezone id}
```

If the MySQL server timezone is different from the host timezone (which is strongly not recommended), then `-tz {valid timezone id}` should be used instead where `{valid timezone id}` should be the same as the MySQL server timezone.

9.24.3.8.2. trepctl heartbeat Internal Implementation

Internally, the heartbeat system operates through a tag added to the metadata of the THL entry and through a dedicated `heartbeat` [693] 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 = 2025-02-25 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= '2025-02-25 12:14:57',
salt= 9, name= 'dataload'  WHERE id= 1
```

During replication, Replicas identify the heartbeat and record this information into their own `heartbeat` [693] 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.

9.24.3.9. trepctl load Command

Load the replicator service.

`trepctl` load

Load the replicator service. The service name must be specified on the command-line, even when only one service is configured:

```
shell> trepctl load
Operation failed: You must specify a service name using -service
```

The service name can be specified using the `-service` option:

```
shell> trepctl -service alpha load
Service loaded successfully: name=alpha
```

9.24.3.10. trepctl offline Command

The trepctl offline command puts the replicator into the offline state, stopping replication.

`trepctl` offline [`-immediate`]

To put the replicator offline:

```
shell> trepctl offline
```

While offline:

- Transactions are not extracted from the source dataserver.
- Transactions are not applied to the destination dataserver.

Certain operations on the replicator, including updates to the operating system and dataserver should be performed while in the offline state.

By default, the replicator goes offline in deferred mode, allowing the current transactions being read from the binary log, or applied to the dataserver to complete, the sequence number table in the database is updated, and the replicator is placed offline, stopping replication.

To stop replication immediately, within the middle of an executing transaction, use the `-immediate` option:

```
shell> trepctl offline -immediate
```

9.24.3.11. trepctl offline-deferred Command

The trepctl offline-deferred sets a future sequence, event or heartbeat as the trigger to put the replicator in the offline state.

`trepctl` offline-deferred [`-at-event event`] [`-at-heartbeat [heartbeat]`] [`-at-seqno seqno`] [`-at-time YYYY-MM-DD_hh:mm:ss`]

Where:

Table 9.53. trepctl offline-deferred Command Options

Option	Description
`-at-event event` [377]	Go offline at the specified event
`-at-heartbeat [heartbeat]` [377]	Go offline when the specified heartbeat is identified

Option	Description
-at-seqno seqno [377]	Go offline at the specified sequence number
-at-time YYYY-MM-DD_hh:mm:ss [377]	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 9.24.3.10, "trepctl offline Command".

The supported triggers are:

- -at-seqno [377]

  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 [377]

  Specifies the event where replication should stop:

  ```
  shell> trepctl offline-deferred -at-event 'mysql-bin.000009:0000000000088140;0'
  ```

  Because there is not a one-to-one relationship between global transaction IDs and events, the replicator will go offline at a transaction that has an event ID higher than the deferred event ID. If the event specification is located within the middle of a THL transaction, the entire transaction is applied.

- -at-heartbeat [377]

  Specifies the name of a specific heartbeat to look for when replication should be stopped.

- -at-time [377]

  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 2025-02-25_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: 2025-02-25 00:00:00 EDT
```

Offline deferred settings are cleared when the replicator is put into the offline state, either manually or automatically.

9.24.3.12. trepctl online Command

The trepctl online command puts the replicator into the online state. During the state change from offline to online various options can be used to control how the replicator goes back on line. For example, the replicator can be placed online, skipping one or more faulty transactions or disabling specific configurations.

`trepctl` online [`-base-seqno x`] [`-force`] [`-from-event event`] [`-no-checksum`] [`-skip-seqno seqdef`] [`-until-event event`] [`-until-heartbeat [name]`] [`-until-seqno seqno`] [`-until-time YYYY-MM-DD_hh:mm:ss`]

Where:

Table 9.54. trepctl online Command Options

Option	Description
`-base-seqno x`	On a Primary, restart replication using the specified sequence number
`-force`	Force the online state
`-from-event event`	Start replication from the specified event
`-no-checksum`	Disable checksums for all events when going online
`-skip-seqno seqdef`	Skip one, multiple, or ranges of sequence numbers before going online
`-until-event event` [379]	Define an event when replication will stop
`-until-heartbeat [name]` [379]	Define a heartbeat when replication will stop
`-until-seqno seqno` [379]	Define a sequence no when replication will stop
`-until-time YYYY-MM-DD_hh:mm:ss` [380]	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.

9.24.3.12.1. Going Online from Specific Transaction Points

If there is one, or more, event in the THL that could not be applied to the Replica because of a mismatch in the data (for example, a duplicate key), the event or events can be skipped using the `-skip-seqno` option. For example, the status shows that a statement failed:

```
shell> trepctl status
...
pendingError           : Event application failed: seqno=5250 fragno=0 »
message=java.sql.SQLException: Statement failed on slave but succeeded on master
...
```

To skip the single sequence number, `5250`, shown:

```
shell> trepctl online -skip-seqno 5250
```

The sequence number specification can be specified according to the following rules:

- A single sequence number:

  ```
  shell> trepctl online -skip-seqno 5250
  ```

- A sequence range:

  ```
  shell> trepctl online -skip-seqno 5250-5260
  ```

- A comma-separated list of individual sequence numbers and/or ranges:

  ```
  shell> trepctl online -skip-seqno 5250,5251,5253-5260
  ```

9.24.3.12.2. Going Online from a Base Sequence Number

> **Note**
>
> To set the position of the replicator, the dsctl command can also be used.

Alternatively, the base sequence number, the transaction ID where replication should start, can be specified explicitly:

```
shell> trepctl online -base-seqno 5260
```

Warning

Use of `-base-seqno` should be restricted to replicators in the *`master`* role only. Use on Replicas may lead to duplication or corruption of data.

Note

If issuing `-base-seqno` and `-from-event` together, you need to also issue the `-force` option for it to succeed, otherwise a warning will be displayed

9.24.3.12.3. Going Online from a Specific Event

Note

To set the position of the replicator, the dsctl command can also be used.

If the source event (for example, the MySQL binlog position) is known, this can be used as the reference point when going online and restarting replication:

```
shell> trepctl online -from-event 'mysql-bin.000011:0000000000002552;0'
```

When used, replication will start from the next event within the THL. The event ID provided must be valid. The event cannot be found in the THL, the operation will fail.

Note

If issuing `-base-seqno` and `-from-event` together, you need to also issue the `-force` option for it to succeed, otherwise a warning will be displayed

9.24.3.12.4. Going Online Until Specific Transaction Points

There are times when it is useful to be able to online until a specific point in time or in the replication stream. For example, when performing a bulk load parallel replication may be enabled, but only a single applier stream is required once the load has finished. The replicator can be configured to go online for a limited period, defined by transaction IDs, events, heartbeats, or a specific time.

The replicator must be in the offline state before the deferred online specifications are made. Multiple deferred online states can be specified in the same command when going online.

The setting of a future offline state can be seen by looking at the *`offlineRequests`* field when checking the status:

```
shell> trepctl status
...
minimumStoredSeqNo     : 0
offlineRequests        : Offline at sequence number: 5262;Offline at time: 2014-01-01 00:00:00 EST
pendingError           : NONE
...
```

If the replicator goes offline for any reason before the deferred offline state is reached, the deferred settings are lost.

Going Online Until Specified Sequence Number

To go online until a specific transaction ID, use `-until-seqno` [379]:

```
shell> trepctl online -until-seqno 5260
```

This will process all transactions up to, and including, sequence 5260, at which point the replicator will go offline.

Going Online Until Specified Event

To go online until a specific event ID:

```
shell> trepctl online -until-event  'mysql-bin.000011:0000000000003057;0'
```

Replication will go offline when the event ID up to the specified event has been processed.

Going Online Until Heartbeat

To go online until a heartbeat event:

```
shell> trepctl online -until-heartbeat
```

Heartbeats are inserted into the replication stream periodically, replication will stop once the heartbeat has been seen before the next transaction. A specific heartbeat can also be specified:

```
shell> trepctl online -until-heartbeat load-finished
```

Going Online Until Specified Time

To go online until a specific date and time:

```
shell> trepctl online -until-time 2014-01-01_00:00:00
```

Replication will go offline once the transaction being processed at the time specified has completed.

9.24.3.12.5. Going Online by Force

In situations where the replicator needs to go online, the online state can be forced. This changes the replicator state to online, but provides no guarantees that the online state will remain in place if another, different, error stops replication.

```
shell> trepctl online -force
```

9.24.3.12.6. Going Online without Validating Checksum

In the event of a checksum problem in the THL, checksums can be disabled using the `-no-checksum` option:

```
shell> trepctl online -no-checksum
```

This will bring the replicator online without reading or writing checksum information.

> **Important**
>
> Use of the `-no-checksum` option disables both the reading and writing of checksums on log records. If starting the replicator without checksums to get past a checksum failure, the replicator should be taken offline again once the offending event has been replicated. This will avoid generating too many local records in the THL without checksums.

9.24.3.13. trepctl pause Command

The trepctl pause command allows you to pause a specific stage of the replicator, either indefinitely or for a specific time.

`trepctl` pause [`-stage stage-to-pause`] [`-time value-in-seconds`]

Where:

Table 9.55. trepctl pause Command Options

Option	Description
`-stage stage-to-pause`	Used to specify the stage to pause, for example thl-to-q
`-time value-in-seconds`	Used to specify the time for a stage to remain paused. Specify in seconds. If not supplied, stage will pause indefinitely, or until a resume command is issued, or until the replicator is restarted.

To pause the thl-to-q stage of the replicator use the standard form of the command:

```
shell> trepctl pause -stage thl-to-q
```

To pause the thl-to-q stage of the replicator for 2 minutes (120 seconds) use the command:

```
shell> trepctl pause -stage thl-to-q -time 120
```

If no time specified, the stage will remain paused until a resume command is issued, or until the replicator is restarted

9.24.3.14. trepctl perf Command

Display a list of all the internal properties. The list can be filtered.

`trepctl` perf [`-c`] [`-r`]

The perf outputs performance information on a stage by stage basis from the current replicator. The information has been reformatted and extracted from the existing replicator status, task and stage information available through other commands and requests, but reformatted and with values calculated to make identifying specific performance metrics quicker.

For example, on a typical extraction replicator:

```
Statistics since last put online 9265.385s ago
Stage       | Seqno | Latency |      Events |  Extraction |   Filtering |    Applying |       Other |       Total
binlog-to-q |  1604 |  8.779s |          14 |     60.173s |      0.109s |      0.015s |      0.004s |     60.301s
                        Avg time per Event |      4.298s |      0.008s |      0.000s |      0.001s |      4.307s
                          Filters in stage | colnames -> pkey
q-to-thl    |  1604 | 10.613s |          14 |     56.858s |      0.020s |      5.247s |      0.028s |     62.153s
                        Avg time per Event |      4.061s |      0.001s |      0.002s |      0.375s |      4.440s
                          Filters in stage | enumtostring -> settostring
```

On an applier:

```
Statistics since last put online 38.418s ago
Stage         | Seqno | Latency |      Events |  Extraction |   Filtering |    Applying |       Other |       Total
remote-to-thl |  3246 |  1.143s |          42 |     37.831s |      0.001s |      0.403s |      0.011s |     38.246s
                        Avg time per Event |      0.901s |      0.000s |      0.000s |      0.010s |      0.911s
thl-to-q      |  3246 |  1.209s |        1654 |     37.113s |      0.005s |      1.090s |      0.098s |     38.306s
                        Avg time per Event |      0.022s |      0.000s |      0.000s |      0.001s |      0.023s
q-to-dbms     |  3235 |  3.746s |        1644 |     22.226s |      0.019s |     15.242s |      0.338s |     37.825s
                        Avg time per Event |      0.014s |      0.000s |      0.000s |      0.009s |      0.023s
                          Filters in stage | mysqlsessions -> pkey
```

The individual statistics shown are as follows:

- All statistics within the replicator are reset when the replicator goes `ONLINE`. The statistics shown are therefore displayed relative to the current uptime for the replicator.
- For each stage, the following information is shown:
  - Stage name
  - Seqno — this is the current *`SEQNO`* [679] number for the specified stage. A difference in sequence numbers is possible (as seen in the applier example above) during startup or synchronisation.
  - Latency — the latency of this stage compared to the commit time of the original transaction.
  - Events — the number of THL events processed by this stage.

  Statistics are then shown for each stage, two rows, first for the time to process all of the specified events, and then an average processing time for the events processed during that time within that stage. The individual statistics shown are as follows:
  - Extraction — the time taken to extract the event from the current source. On an extractor, this is the source database (for example, the binary log in MySQL). On other stages this is the time to read from disk or the remote replicator the THL event.
  - Filtering — the time taken to process the events through the filters configured in the specified stage.
  - Applying — the time taken to apply the event to the end of the stage, whether that is to THL on disk, the next queue in preparation for the next stage, or the target database.
  - Other — the time taken for other parts of the stage process, this includes waiting for thread management, updating internal structures, and recording information in the target datasource system, such as `trep_commit_seqno` [693].
- Filters in stage — The list of filters configured for this stage in the order in which they are applied to the event.

For convenience, the performance display can be set to refresh with a configured interval using the trepctl perf -r 5 command.

The `-c` can also be used with the `-r` option to limit the number of refresh iterations, for exmaple trepctl perf -r 5 -c 10

In the event that the replicator is currently offline, no statistics are displayed:

```
shell> trepctl perf
Currently not online; performance stats not available
State: Safely Offline for 6.491s
```

9.24.3.15. trepctl properties Command

Display a list of all the internal properties. The list can be filtered.

`trepctl` properties [`-filter name`] [`-values`]

The list of properties can be used to determine the current configuration:

```
shell> trepctl properties
{
```

```
"replicator.store.thl.log_file_retention": "7d",
"replicator.filter.bidiSlave.allowBidiUnsafe": "false",
"replicator.extractor.dbms.binlog_file_pattern": "mysql-bin",
"replicator.filter.pkey.url": »
"jdbc:mysql:thin://host2:3306/tungsten_alpha?createDB=true",
...
}
```

> **Note**
>
> Passwords are not displayed in the output.

The information is output as a JSON object with key/value pairs for each property and corresponding value.

The list can be filtered using the `-filter` option:

```
shell> trepctl properties -filter shard
{
"replicator.filter.shardfilter": »
"com.continuent.tungsten.replicator.shard.ShardFilter",
"replicator.filter.shardbyseqno": »
"com.continuent.tungsten.replicator.filter.JavaScriptFilter",
"replicator.filter.shardbyseqno.shards": "1000",
"replicator.filter.shardfilter.enforceHome": "false",
"replicator.filter.shardfilter.unknownShardPolicy": "error",
"replicator.filter.shardbyseqno.script": »
"../../tungsten-replicator//samples/extensions/javascript/shardbyseqno.js",
"replicator.filter.shardbytable.script": »
"../../tungsten-replicator//samples/extensions/javascript/shardbytable.js",
"replicator.filter.shardfilter.enabled": "true",
"replicator.filter.shardfilter.allowWhitelisted": "false",
"replicator.shard.default.db": "stringent",
"replicator.filter.shardbytable": »
"com.continuent.tungsten.replicator.filter.JavaScriptFilter",
"replicator.filter.shardfilter.autoCreate": "false",
"replicator.filter.shardfilter.unwantedShardPolicy": "error"
}
```

The value or values from filtered properties can be retrieved by using the `-values` option:

```
shell>  trepctl properties -filter site.name -values
default
```

If a filter that would select multiple values is specified, all the values are listed without field names:

```
shell> trepctl properties -filter shard -values
com.continuent.tungsten.replicator.shard.ShardFilter
com.continuent.tungsten.replicator.filter.JavaScriptFilter
1000
false
../../tungsten-replicator//samples/extensions/javascript/shardbyseqno.js
error
../../tungsten-replicator//samples/extensions/javascript/shardbytable.js
true
false
stringent
com.continuent.tungsten.replicator.filter.JavaScriptFilter
false
error
```

9.24.3.16. trepctl purge Command

Forces all logins on the attached database, other than those directly related to Tungsten Cluster, to be disconnected. The command is only supported on a Primary, and can be used to disconnect users before a switchover or taking a Primary offline to prevent further use of the system.

`trepctl` purge [`-limit s`] [`-y`]

Where:

Table 9.56. trepctl purge Command Options

Option	Description
`-limit s` [383]	Specify the waiting time for the operation
`-y` [383]	Indicates that the command should continue without interactive confirmation

Warning

Use of the command will disconnect running users and queries and may leave the database is an unknown state. It should be used with care, and only when the dangers and potential results are understood.

To close the connections:

```
shell> trepctl purge
Do you really want to purge non-Tungsten DBMS sessions? [yes/NO]
```

You will be prompted to confirm the operation. To skip this confirmation and purge connections, use the `-y` [383] option:

```
shell> trepctl purge -y
Directing replicator to purge non-Tungsten sessions
Number of sessions purged: 0
```

An optional parameter, `-wait` [383], defines the period of time that the operation will wait before returning to the command-line.

An optional parameter, `-limit` [383], defines the period of time that the operation will wait before returning to the command-line.

9.24.3.17. trepctl qs Command

The trepctl qs (quickstatus) command provides a quicker, simpler, status display for the replicator showing only the critical information in a human-readable form. For example:

```
shell> trepctl qs
State: alpha Online for 4.21s, running for 1781.766s
Latency: 18.0s from source DB commit time on thl://ubuntuheterosrc.mcb:2112/ into target database
     1216.315s since last source commit
Sequence: 4804 last applied, 0 transactions behind (0-4804 stored) estimate 0.00s before synchronization
```

The information presented is as follows:

- State: alpha Online for 4.21s, running for 1781.766s

  The top line shows the basic status information about the replicator:

  - The name of the service [`alpha`].
  - The replicator's current state [`Online`] and the time in that state.
  - The amount of time the replicator has been running.

- Latency: 18.0s from source DB commit time on thl://ubuntuheterosrc.mcb:2112/ into target database

  The second lines shows the latency information, the information shown is based on the role of the replicator. The above line is shown on an applier, where the latency information shows the write delay into the target database, where the information is coming from, and applying to the target database. For a Primary (extractor) the information shown describes the latency from extraction into the THL files:

  ```
  State: alpha Online for 1699091.442s, running for 1699093.138s
  Latency: 0.113s from DB commit time on ubuntuheterosrc into THL
       1679.354s since last database commit
  Sequence: 4859 last applied, 0 transactions behind (0-4859 stored) estimate 0.00s before synchronization
  ```

- 1216.315s since last source commit

  The next line shows the interval since the last time there was a database commit. On a Primary (extractor) is the time between the last database commit to the binary log and the information being written to THL. On a Replica, it's the time between the last database commit *on the source database* and when the transaction was written to the target.

- Sequence: 4804 last applied, 0 transactions behind (0-4804 stored) estimate 0.00s before synchronization

  The last line shows the sequence information:

  - The last applied sequence number (to THL on a Primary, or to the target database on a Replica).
  - The number of transactions behind the current stored transaction list. This is an indication on a Replica of how far behind in transactions (not latency) the Replica is from the Primary.
  - The range of transactions currently stored (from minimum to maximum stored sequence number).
  - An estimate of the how long it will take to apply the outstanding transactions. The calculation is made by determining the average rate transactions are being applied (either extraction or applying) against the number of outstanding transactions. It assumes all outstanding

transactions are of an equal size. The actual THL transaction size is not taken into account. For information on THL sizes, try the thl list -sizes command.

If the replicator is offline due to being deliberately placed offline using trepctl offline then the basic information and status is shown:

```
shell> trepctl qs
State: Safely Offline for 352.775s
```

In the event of a replicator failure of some kind this will be reported in the output:

```
State: alpha Faulty (Offline) for 2.613s
Error Reason: SEQNO 4859 did not apply
Error: CSV loading failed: schema=test table=msg CSV file=/opt/continuent/tmp/staging/alpha/staging0/test-msg-4859.csv
» message=Wrapped com.continuent.tungsten.replicator.ReplicatorException: OS command failed: command=cqlsh --keyspace=test
» --execute="copy stage_xxx_msg (tungsten_opcode,tungsten_seqno,tungsten_row_id,tungsten_commit_timestamp,id,msg) from
» '/opt/continuent/tmp/staging/alpha/staging0/test-msg-4859.csv' with NULL='NULL';" rc=1 stdout= stderr=Connection
» error: ('Unable to complete the operation against any hosts', {})
```

For convenience, the performance display can be set to refresh with a configured interval using the trepctl qs -r 5 command.

The -c can also be used with the -r option to limit the number of refresh iterations, for exmaple trepctl qs -r 5 -c 10

9.24.3.18. trepctl reset Command

The trepctl reset command resets an existing replicator service, performing the following operations:

- Deleting the local THL and relay directories
- Removes the Tungsten schema from the dataserver
- Removes any dynamic properties that have previously been set

The service name must be specified, using -service.

trepctl reset [-all] [-db] [-relay] [-thl] [-y]

Where:

Table 9.57. trepctl reset Command Options

Option	Description
-all [384]	Deletes the thl directory, relay logs directory and tungsten database for the service.
-db [385]	Deletes the tungsten_{service_name} database for the service
-relay [385]	Deletes the relay directory for the service
-thl [384]	Deletes the thl directory for the service
-y [384]	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 [384] option:

```
shell> trepctl -service alpha reset -y
```

Then put the replicator back online again:

```
shell> trepctl online
```

You can also reset only part of the overall service by including one of the following options:

- Reset all components of the service.
- Reset the THL. This is equivalent to running thl purge.

- Reset the relay log contents.
- Reset the database, including emptying the `trep_commit_seqno` [693] and other control tables.

9.24.3.19. trepctl restore Command

Restores the database on a host from a previous backup.

`trepctl` restore

Once the restore has been completed, the node will remain in the `OFFLINE` state. The datasource should be switched `ONLINE` using trepctl:

```
shell> trepctl online
```

Any outstanding events from the Primary will be processed and applied to the Replica, which will catch up to the current Primary status over time.

9.24.3.20. trepctl resume Command

The trepctl resume command allows you to resume a specific stage of the replicator that has been paused using the `pause` option.

`trepctl` resume [`-stage stage-to-resume`]

Where:

Table 9.58. trepctl resume Command Options

Option	Description
`-stage stage-to-resume`	Used to specify the stage to resume, for example thl-to-q

To resume the thl-to-q stage of the replicator use the standard form of the command:

```
shell> trepctl resume -stage thl-to-q
```

9.24.3.21. trepctl setdynamic Command

The trepctl setdynamic command allows you to change certain dynamic properties without the need to execute tpm update

`trepctl` setdynamic [`-property`] [`-value`]

Where:

Table 9.59. trepctl setdynamic Command Options

Option	Description
`-property`	Specify the property to change
`-value`	Specify the value of the specified -property

To change a property, specify the property using the `-property` parameter.

> **Important**
>
> To change a property dynamically, the service must first be `OFFLINE`

```
shell> trepctl setdynamic -property <property>
```

The list of properties that can be dynamically changed are as follows:

- `replicator.autoRecoveryMaxAttempts` : This allows you to dynamically alter the behaior of the autoRecovery options. A value specificed greater than 0 will enable the autoRecovery. Set a value of 0 to disable.

  The following example enables autoRecovery with the MaxAttempts value of 10

  ```
  shell> trepctl -service beta setdynamic -property replicator.autoRecoveryMaxAttempts -value 10
  ```

  The following example disables autoRecovery

  ```
  shell> trepctl -service beta setdynamic -property replicator.autoRecoveryMaxAttempts -value 0
  ```

- `replicator.thl.protocol.client.serialization` : This allows you to dynamically alter THL transfer protocol for In-Flight compression.

  The following example sets the protocol to DEFLATE

  ```
  shell> trepctl setdynamic -property replicator.thl.protocol.client.serialization -value DEFLATE
  ```

  For more information on THL transfer protocol, see Section 6.18.4, "THL Encryption and Compression"

9.24.3.22. trepctl setrole Command

The trepctl setrole command changes the role of the replicator service. This command can be used to change a configured host between Replica and Primary roles, for example during switchover.

`trepctl` setrole [`-role master` | `slave` | `relay` | `thl-applier` | `thl-client` | `thl-server`] [`-uri`]

Where:

Table 9.60. trepctl setrole Command Options

Option	Description
`-role`	Replicator role
`-uri` [386]	URI of the Primary

To change the role of a replicator, specify the role using the `-role` parameter. The replicator must be offline when the role change is issued:

```
shell> trepctl setrole -role master
```

When setting a Replica, the URI of the Primary can be optionally supplied:

```
shell> trepctl setrole -role slave -uri thl://host1:2112/
```

See Section 6.3.5, "Understanding Replicator Roles" for more details on each role.

9.24.3.23. trepctl shard Command

The trepctl shard command provides and interface to the replicator shard system definition system.

`trepctl` shard [`-delete shard`] [`-insert shard`] [`-list`] [`-update shard`]

Where:

Table 9.61. trepctl shard Command Options

Option	Description
`-delete shard`	Delete a shard definition
`-insert shard`	Add a new shard definition
`-list`	List configured shards
`-update shard`	Update a shard definition

The replicator shard system is used during multi-site replication configurations to control where information is replicated.

9.24.3.23.1. Listing Current Shards

To obtain a list of the currently configured shards:

```
shell> trepctl shard -list
shard_id  master  critical
alpha     sales   true
```

The shard map information can also be captured and then edited to update existing configurations:

```
shell> trepctl shard -list>shard.map
```

9.24.3.23.2. Inserting a New Shard Configuration

To add a new shard map definition, either enter the information interactively:

```
shell> trepctl shard -insert
Reading from standard input
...
1 new shard inserted
```

Or import from a file:

```
shell> trepctl shard -insert  < shard.map
Reading from standard input
1 new shard inserted
```

9.24.3.23.3. Updating an Existing Shard Configuration

To update a definition:

```
shell> trepctl shard -update  < shard.map
Reading from standard input
1 shard updated
```

9.24.3.23.4. Deleting a Shard Configuration

To delete a single shard definition, specify the shard name:

```
shell> trepctl shard -delete alpha
```

9.24.3.24. trepctl status Command

The trepctl status command provides status information about the selected data service. The status information by default is a generic status report containing the key fields of status information. More detailed service information can be obtained by specifying the status name with the `-name` parameter.

The format of the command is:

`trepctl` status [`-c`] [`-json`] [`-name channel-assignments` | `services` | `shards` | `stages` | `stores` | `tasks` | `watches`] [`-r`]

Where:

Table 9.62. trepctl status Command Options

Option	Description
`-c`	Used with the -r option to refresh the status display *c* number of times.
`-json`	Output the information in JSON format
`-name`	Select a specific group of status information
`-r`	Refresh the status display

For example, to get the basic status information:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000018:0000000000070227;140
appliedLastGoodLatency : 0.464
appliedLastGoodSeqno   : 206
appliedLastSeqno       : 206
appliedLatency         : 0.464
autoRecoveryEnabled    : false
autoRecoveryTotal      : 0
channels               : 1
clusterName            : alpha
currentEventId         : NONE
currentTimeMillis      : 1740560260007
dataServerHost         : db2
extensions             :
host                   : db2
latestEpochNumber      : 0
masterConnectUri       : thl://db1:2112/
masterListenUri        : thl://db2:2112/
maximumStoredSeqNo     : 206
minimumStoredSeqNo     : 0
offlineRequests        : NONE
```

```
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : thl://db1:2112/
relativeLatency        : 5.007
resourceJdbcDriver     : org.drizzle.jdbc.DrizzleDriver
resourceJdbcUrl        : jdbc:mysql:thin://db2:13306/${DBNAME}?jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull&tinyInt1isBit=false&allowM
resourceVendor         : mysql
rmiPort                : 10000
role                   : slave
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : db2
state                  : ONLINE
timeInStateSeconds     : 64.375
timezone               : GMT
transitioningTo        :
uptimeSeconds          : 82.508
useSSLConnection       : false
version                : Tungsten Clustering 7.0.3 build 141
Finished status command...
```

For more information on the field information output, see Section E.2, “Generated Field Reference”.

For convenience, the display can be set to refresh with a configured interval using the trepctl status -r 5 command.

The `-c` can also be used with the `-r` option to limit the number of refresh iterations, for exmaple trepctl status -r 5 -c 10 will refresh every 5 seconds for a total of 10 iterations

9.24.3.24.1. Getting Detailed Status

More detailed information about selected areas of the replicator status can be obtained by using the `-name` option.

Detailed Status: Channel Assignments

When using a single threaded replicator service, the trepctl status -name channel-assignments [388] will output an empty status. In parallel replication deployments, the trepctl status -name channel-assignments [388] listing will output the list of schemas and their assigned channels within the configured channel quantity configuration. For example, in the output below, only two channels are shown, although five channels were configured for parallel apply:

```
shell> trepctl status -name channel-assignments
Processing status command (channel-assignments)...
NAME      VALUE
----      -----
channel : 0
shard_id: test
NAME      VALUE
----      -----
channel : 0
shard_id: tungsten_alpha
Finished status command (channel-assignments)...
```

Detailed Status: Services

The trepctl status -name services [388] status output shows a list of the currently configure internal services that are defined within the replicator.

```
shell> trepctl status -name services
Processing status command (services)...
NAME              VALUE
----              -----
accessFailures  : 0
active          : true
maxChannel      : -1
name            : channel-assignment
storeClass      : com.continuent.tungsten.replicator.channel.ChannelAssignmentService
totalAssignments: 0
Finished status command (services)...
```

Detailed Status: Shards

The trepctl status -name shards [388] status output lists the individual shards in operation, most useful when parallel apply has been configured within the replicator, showing a summary of last applied sequences and the corresponding binlog references.

In an environment not configured with parallel apply, the shards output will just show a single entry

Detailed Status: Stages

The trepctl status -name stages [389] 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)...
```

Detailed Status: Stores

The trepctl status -name stores [389] status output lists the individual internal stores used for replicating THL data. This includes both physical (on disk) THL storage and in-memory storage. This includes the sequence number, file size and retention information.

For example, the information shown below is taken from a Primary service, showing the stages, `binlog-to-q` which reads the information from the binary log, and the in-memory `q-to-thl` that writes the information to THL.

```
shell> trepctl status -name stages
Processing status command (stages)...
NAME                 VALUE
----                 -----
applier.class      : com.continuent.tungsten.replicator.storage.InMemoryQueueAdapter
applier.name       : queue
blockCommitRowCount: 1
committedMinSeqno  : 224
extractor.class    : com.continuent.tungsten.replicator.extractor.mysql.MySQLExtractor
extractor.name     : dbms
name               : binlog-to-q
processedMinSeqno  : 224
taskCount          : 1
NAME                 VALUE
----                 -----
applier.class      : com.continuent.tungsten.replicator.thl.THLStoreApplier
applier.name       : autoflush-thl-applier
blockCommitRowCount: 10
committedMinSeqno  : 224
extractor.class    : com.continuent.tungsten.replicator.storage.InMemoryQueueAdapter
extractor.name     : queue
name               : q-to-thl
processedMinSeqno  : 224
taskCount          : 1
```

```
Finished status command (stages)...
```

When running parallel replication, the output shows the store name, sequence number and status information for each parallel replication channel:

```
shell> trepctl status -name stores
Processing status command (stores)...
NAME                      VALUE
----                      -----
activeSeqno             : 15
doChecksum              : false
flushIntervalMillis     : 0
fsyncOnFlush            : false
logConnectionTimeout    : 28800
logDir                  : /opt/continuent/thl/alpha
logFileRetainMillis     : 604800000
logFileSize             : 100000000
maximumStoredSeqNo      : 16
minimumStoredSeqNo      : 0
name                    : thl
readOnly                : false
storeClass              : com.continuent.tungsten.replicator.thl.THL
timeoutMillis           : 2147483647
NAME                      VALUE
----                      -----
criticalPartition       : -1
discardCount            : 0
estimatedOfflineInterval: 0.0
eventCount              : 1
headSeqno               : 16
intervalGuard           : AtomicIntervalGuard (array is empty)
maxDelayInterval        : 60
maxOfflineInterval      : 5
maxSize                 : 10
name                    : parallel-queue
queues                  : 5
serializationCount      : 0
serialized              : false
stopRequested           : false
store.0                 : THLParallelReadTask task_id=0 thread_name=store-thl-0 »
                        hi_seqno=16 lo_seqno=16 read=1 accepted=1 discarded=0 events=0
store.1                 : THLParallelReadTask task_id=1 thread_name=store-thl-1 »
                        hi_seqno=16 lo_seqno=16 read=1 accepted=0 discarded=1 events=0
store.2                 : THLParallelReadTask task_id=2 thread_name=store-thl-2 »
                        hi_seqno=16 lo_seqno=16 read=1 accepted=0 discarded=1 events=0
store.3                 : THLParallelReadTask task_id=3 thread_name=store-thl-3 »
                        hi_seqno=16 lo_seqno=16 read=1 accepted=0 discarded=1 events=0
store.4                 : THLParallelReadTask task_id=4 thread_name=store-thl-4 »
                        hi_seqno=16 lo_seqno=16 read=1 accepted=0 discarded=1 events=0
storeClass              : com.continuent.tungsten.replicator.thl.THLParallelQueue
syncInterval            : 10000
Finished status command (stores)...
```

Detailed Status: Tasks

The trepctl status -name tasks [390] command outputs the current list of active tasks within a given service, with one block for each stage within the replicator service.

```
shell> trepctl status -name tasks
Processing status command (tasks)...
NAME                    VALUE
----                    -----
appliedLastEventId    : mysql-bin.000038:0000000011253929;-1
appliedLastSeqno      : 1604
appliedLatency        : 8.779
applyTime             : 0.015
averageBlockSize      : 3.500
cancelled             : false
commits               : 4
currentBlockSize      : 0
currentLastEventId    : mysql-bin.000038:0000000011253929;-1
currentLastFragno     : 11
currentLastSeqno      : 1604
eventCount            : 14
extractTime           : 60.173
filterTime            : 0.109
lastCommittedBlockSize: 12
lastCommittedBlockTime: 59.145
otherTime             : 0.004
stage                 : binlog-to-q
state                 : extract
taskId                : 0
timeInCurrentEvent    : 8804.187
```

```
NAME                   VALUE
----                   -----
appliedLastEventId    : mysql-bin.000038:0000000011253929;-1
appliedLastSeqno      : 1604
appliedLatency        : 10.613
applyTime             : 5.247
averageBlockSize      : 2.800
cancelled             : false
commits               : 5
currentBlockSize      : 0
currentLastEventId    : mysql-bin.000038:0000000011253929;-1
currentLastFragno     : 11
currentLastSeqno      : 1604
eventCount            : 14
extractTime           : 56.858
filterTime            : 0.02
lastCommittedBlockSize: 12
lastCommittedBlockTime: 5.092
otherTime             : 0.028
stage                 : q-to-thl
state                 : extract
taskId                : 0
timeInCurrentEvent    : 8802.323
Finished status command (tasks)...
```

The list of tasks and information provided depends on the role of the host, the number of stages, and whether parallel apply is enabled.

Detailed Status: Watches

The trepctl status -name watches [391] command outputs the current list of tasks the replicator is waiting on before a specific action.

For example, if you issue trepctl offline-deferred -at-seqno x, the the output of watches will show the stages waiting on the specific seqno.

The following example show the use of offline-deferred and the subsequent resulting output from watches

```
shell> trepctl offline-deferred -at-seqno 234
shell> trepctl status -name watches
Processing status command (watches)...
NAME       VALUE
----       -----
action   : cancel tasks
cancelled: false
committed: false
done     : false
matched  : [[0:false]]
predicate: SeqnoWatchPredicate seqno=234
stage    : remote-to-thl
NAME       VALUE
----       -----
action   : cancel tasks
cancelled: false
committed: false
done     : false
matched  : [[0:false]]
predicate: SeqnoWatchPredicate seqno=234
stage    : thl-to-q
NAME       VALUE
----       -----
action   : cancel tasks
cancelled: false
committed: false
done     : false
matched  : [[0:false]]
predicate: SeqnoWatchPredicate seqno=234
stage    : q-to-dbms
Finished status command (watches)...
```

9.24.3.24.2. Getting JSON Formatted Status

Status information can also be requested in JSON format. The content of the information is identical, only the representation of the information is different, formatted in a JSON wrapper object, with one key/value pair for each field in the standard status output.

Examples of the JSON output for each status output are provided below. For more information on the fields displayed, see Section E.2, "Generated Field Reference".

trepctl status JSON Output

```
[
{
 "serviceType":"local",
 "sourceId":"db4",
```

```
 "appliedLastGoodSeqno":"5",
 "role":"relay",
 "pendingErrorEventId":"NONE",
 "timezone":"GMT",
 "siteName":"default",
 "useSSLConnection":"false",
 "pipelineSource":"thl:\/\/db1:2112\/",
 "uptimeSeconds":"818.794",
 "resourceJdbcDriver":"org.drizzle.jdbc.DrizzleDriver",
 "appliedLastEventId":"mysql-bin.000018:0000000000001432;140",
 "seqnoType":"java.lang.Long",
 "pendingExceptionMessage":"NONE",
 "resourceVendor":"mysql",
 "appliedLatency":"0.0",
 "clusterName":"beta",
 "host":"db4",
 "state":"GOING-ONLINE:SYNCHRONIZING",
 "rmiPort":"10000",
 "resourceJdbcUrl":"jdbc:mysql:thin:\/\/db4:13306\/${DBNAME}?jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull&tinyInt1isBit=false&allowMul
 "simpleServiceName":"beta",
 "pendingErrorCode":"NONE",
 "masterConnectUri":"thl:\/\/db1:2112\/",
 "pendingErrorSeqno":"-1",
 "latestEpochNumber":"0",
 "offlineRequests":"NONE",
 "timeInStateSeconds":"534.415",
 "minimumStoredSeqNo":"0",
 "currentEventId":"NONE",
 "serviceName":"beta",
 "autoRecoveryTotal":"0",
 "version":"Tungsten Clustering 7.0.3 build 141",
 "masterListenUri":"thl:\/\/db4:2112\/",
 "transitioningTo":"",
 "appliedLastSeqno":"5",
 "dataServerHost":"db4",
 "appliedLastGoodLatency":"0.0",
 "relativeLatency":"743.295",
 "extensions":"",
 "channels":"1",
 "currentTimeMillis":"1740560996295",
 "maximumStoredSeqNo":"206",
 "pendingError":"NONE",
 "autoRecoveryEnabled":"false"
}
]
```


Detailed Status: Channel Assignments JSON Output


```
shell> trepctl status -name channel-assignments -json
[
{
   "channel" : "0",
   "shard_id" : "cheffy"
},
{
   "channel" : "0",
   "shard_id" : "tungsten_alpha"
}
]
```


Detailed Status: Services JSON Output


```
shell> trepctl status -name services -json
[
{
   "totalAssignments" : "2",
   "accessFailures" : "0",
   "storeClass" : "com.continuent.tungsten.replicator.channel.ChannelAssignmentService",
   "name" : "channel-assignment",
   "maxChannel" : "0"
}
]
```


Detailed Status: Shards JSON Output


```
shell> trepctl status -name shards -json
[
{
   "stage" : "q-to-dbms",
   "appliedLastEventId" : "mysql-bin.000007:0000000007224342;0",
   "appliedLatency" : "63.099",
   "appliedLastSeqno" : "2514",
   "eventCount" : "16",
```

```
   "shardId" : "cheffy"
}
]
```

Detailed Status: Stages JSON Output

```
shell> trepctl status -name stages -json
[
{
   "applier.name" : "thl-applier",
   "applier.class" : "com.continuent.tungsten.replicator.thl.THLStoreApplier",
   "name" : "remote-to-thl",
   "extractor.name" : "thl-remote",
   "taskCount" : "1",
   "committedMinSeqno" : "2504",
   "blockCommitRowCount" : "1",
   "processedMinSeqno" : "-1",
   "extractor.class" : "com.continuent.tungsten.replicator.thl.RemoteTHLExtractor"
},
{
   "applier.name" : "parallel-q-applier",
   "applier.class" : "com.continuent.tungsten.replicator.storage.InMemoryQueueAdapter",
   "name" : "thl-to-q",
   "extractor.name" : "thl-extractor",
   "taskCount" : "1",
   "committedMinSeqno" : "2504",
   "blockCommitRowCount" : "10",
   "processedMinSeqno" : "-1",
   "extractor.class" : "com.continuent.tungsten.replicator.thl.THLStoreExtractor"
},
{
   "applier.name" : "dbms",
   "applier.class" : "com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier",
   "filter.2.name" : "bidiSlave",
   "name" : "q-to-dbms",
   "extractor.name" : "parallel-q-extractor",
   "filter.1.name" : "pkey",
   "taskCount" : "1",
   "committedMinSeqno" : "2504",
   "filter.2.class" : "com.continuent.tungsten.replicator.filter.BidiRemoteSlaveFilter",
   "filter.1.class" : "com.continuent.tungsten.replicator.filter.PrimaryKeyFilter",
   "filter.0.class" : "com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter",
   "blockCommitRowCount" : "10",
   "filter.0.name" : "mysqlsessions",
   "processedMinSeqno" : "-1",
   "extractor.class" : "com.continuent.tungsten.replicator.storage.InMemoryQueueAdapter"
}
]
```

Detailed Status: Stores JSON Output

```
shell> trepctl status -name stores -json
[
{
   "logConnectionTimeout" : "28800",
   "doChecksum" : "false",
   "name" : "thl",
   "flushIntervalMillis" : "0",
   "logFileSize" : "100000000",
   "logDir" : "/opt/continuent/thl/alpha",
   "activeSeqno" : "2561",
   "readOnly" : "false",
   "timeoutMillis" : "2147483647",
   "storeClass" : "com.continuent.tungsten.replicator.thl.THL",
   "logFileRetainMillis" : "604800000",
   "maximumStoredSeqNo" : "2565",
   "minimumStoredSeqNo" : "2047",
   "fsyncOnFlush" : "false"
},
{
   "storeClass" : "com.continuent.tungsten.replicator.storage.InMemoryQueueStore",
   "maxSize" : "10",
   "storeSize" : "7",
   "name" : "parallel-queue",
   "eventCount" : "119"
}
]
```

Detailed Status: Tasks JSON Output

```
shell> trepctl status -name tasks -json
[
{
```

```
   "filterTime" : "0.0",
   "stage" : "remote-to-thl",
   "currentLastFragno" : "1",
   "taskId" : "0",
   "currentLastSeqno" : "2615",
   "state" : "extract",
   "extractTime" : "604.297",
   "applyTime" : "16.708",
   "averageBlockSize" : "0.982     ",
   "otherTime" : "0.017",
   "appliedLastEventId" : "mysql-bin.000007:0000000111424440;0",
   "appliedLatency" : "63.787",
   "currentLastEventId" : "mysql-bin.000007:0000000111424440;0",
   "eventCount" : "219",
   "appliedLastSeqno" : "2615",
   "cancelled" : "false"
},
{
   "filterTime" : "0.0",
   "stage" : "thl-to-q",
   "currentLastFragno" : "1",
   "taskId" : "0",
   "currentLastSeqno" : "2615",
   "state" : "extract",
   "extractTime" : "620.715",
   "applyTime" : "0.344",
   "averageBlockSize" : "1.904     ",
   "otherTime" : "0.006",
   "appliedLastEventId" : "mysql-bin.000007:0000000111424369;0",
   "appliedLatency" : "63.834",
   "currentLastEventId" : "mysql-bin.000007:0000000111424440;0",
   "eventCount" : "219",
   "appliedLastSeqno" : "2615",
   "cancelled" : "false"
},
{
   "filterTime" : "0.263",
   "stage" : "q-to-dbms",
   "currentLastFragno" : "1",
   "taskId" : "0",
   "currentLastSeqno" : "2614",
   "state" : "apply",
   "extractTime" : "533.471",
   "applyTime" : "61.618",
   "averageBlockSize" : "1.160     ",
   "otherTime" : "24.052",
   "appliedLastEventId" : "mysql-bin.000007:0000000110392640;0",
   "appliedLatency" : "63.178",
   "currentLastEventId" : "mysql-bin.000007:0000000110392711;0",
   "eventCount" : "217",
   "appliedLastSeqno" : "2614",
   "cancelled" : "false"
}
]
```


Detailed Status: Tasks JSON Output


```
shell> trepctl offline-deferred -at-seqno 208
shell> trepctl status -name watches -json
[
{
  "predicate":"SeqnoWatchPredicate seqno=208",
  "committed":"false",
  "stage":"remote-to-thl",
  "action":"cancel tasks",
  "cancelled":"false",
  "matched":"[[0:false]]",
  "done":"false"
},
{
  "predicate":"SeqnoWatchPredicate seqno=208",
  "committed":"false",
  "stage":"thl-to-q",
  "action":"cancel tasks",
  "cancelled":"false",
  "matched":"[[0:false]]",
  "done":"false"
},
{
  "predicate":"SeqnoWatchPredicate seqno=208",
  "committed":"false",
  "stage":"q-to-dbms",
  "action":"cancel tasks",
  "cancelled":"false",
```

```
  "matched":"[[0:false]]",
  "done":"false"
}
]
```

9.24.3.25. trepctl unload Command

Unload the replicator service.

`trepctl` unload [`-y`]

Unload the replicator service entirely. An interactive prompt is provided to confirm the shutdown:

```
shell> trepctl unload
Do you really want to unload replication service alpha? [yes/NO]
```

To disable the prompt, use the `-y` [395] option:

```
shell> trepctl unload -y
Service unloaded successfully: name=alpha
```

The name of the service unloaded is provided for confirmation.

9.24.3.26. trepctl wait Command

The trepctl wait command waits for the replicator to enter a specific state, or for a specific sequence number to be applied to the dataserver.

`trepctl` wait [`-applied seqno`] [`-limit s`] [`-state st`]

Where:

Table 9.63. trepctl wait Command Options

Option	Description
`-applied seqno` [395]	Specify the sequence number to be waited for
`-limit s` [395]	Specify the number of seconds to wait for the operation to complete
`-state st` [395]	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`), or for a specific sequence number to be applied. For example, to wait for the replicator to go into the `ONLINE` state:

```
shell> trepctl wait -state ONLINE
```

This can be useful in scripts when the state maybe changed (for example during a backup or restore operation), allowing for an operation to take place once the requested state has been reached. Once reached, trepctl returns with exit status 0.

To wait a specific sequence number to be applied:

```
shell> trepctl wait -applied 2000
```

This can be useful when performing bulk loads where the sequence number where the bulk load completed is known, or when waiting for a specific sequence number from the Primary to be applied on the Replica. Unlike the offline-deferred operation, no change in the replicator is made. Instead, trepctl simply returns with exit status 0 when the sequence number has bee successfully applied.

If the optional `-limit` [395] 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.

9.25. The tungsten_find_orphaned Command

The tungsten_find_orphaned command assists with locating orphaned events in both the MySQL binary logs as well as in the THL.

This program is designed to handle three failure scenarios:

1. Orphaned MySQL binary logs that were not extracted into THL before a failover

2. Orphaned THL on old Primary that did not make it to the new Primary
3. Orphaned Replica THL on new Primary or online Replica of the new Primary that did not get applied to the database before getting promoted to new Primary

The default action with no arguments is Scenario 1 to locate orphaned MySQL binary logs which have not been extracted into THL on an old Primary before recovery.

For Scenarios 1 and 2

By default, the tungsten_find_orphaned command expects to be run on a (possibly failed) Primary before recovery.

Once any new Replica THL is appended to any existing extracted THL, the tungsten_find_orphaned command will be unable to find the delta between the binlogs and the THL.

If any events are located in the binlogs that do not exist in the THL on disk, they will be counted, and a summary displayed at the end.

Instructions on how to display any actual orphaned events in the binary logs will be provided as well.

For Scenario 2

If the `latestEpochNumber` from the new Primary trepctl status output is provided via the `-e` option, tungsten_find_orphaned will check for THL differences from old to new Primaries.

For Scenario 3

Use `--new` on a NEW Primary to attempt to automatically locate the latest `trepsvc.log` file and determine if there is orphaned THL which exists on disk but had not yet been applied to the database.

If tungsten_find_orphaned is unable to auto-locate the `trepsvc.log` file, please specify the full path to the latest one using `--log` (or `-l`).

Table 9.64. tungsten_find_orphaned Options

Option	Description
`--debug`, `-d`	Enable additional debug output.
`--epoch`, `-e`	Use the --epoch option on an OLD Primary to specify the latestEpochNumber from the new Primary in order to find orphaned THL which did not get copied to the new Primary. Note: Use `tungsten_find_orphaned -E`on the new Primary to obtain the latestEpochNumber (Scenario 2)
`--epochget`, `-E`	Use option --epochget on a NEW Primary to locate the latestEpochNumber for use on the OLD Primary with --epoch (sets quiet mode to true) (Scenario 2)
`--help`, `-h`	Show help text
`--log`, `-l`	Use option --log on a NEW Primary specify the full path to the latest trepsvc.log file. Used when tungsten_find_orphaned is unable to auto-locate trepsvc.log (Scenario 3)
`--new`, `-n`	Use option --new on a NEW Primary to attempt to automatically locate the latest trepsvc.log file and determine if there is orphaned THL which exists on disk but had not yet been applied to the database. (Scenario 3)
`--path`, `-p`	Specify the full path to the executable directory where the thl and trepctl commands are located.
`--quiet`, `-q`	Prevent final message from being output.
`--service`, `-s`	Specify the service name to use with the various commands.
`--thl`	Specify the full path to the thl command executable file.
`--tpm`	Specify the full path to the tpm command executable file.
`--trepctl`	Specify the full path to the trepctl command executable file.
`--verbose`, `-v`	Show verbose output

Examples:

Run on old Primary to find orphaned binary logs not extracted to thl before failover with verbose output (Scenario 1):

```
shell> tungsten_find_orphaned -v
```

Locate THL on an old Primary that was not transferred to the new Primary before failover (Scenario 2):

```
shell> tungsten_find_orphaned -v -e {latestEpochNumber_from_new_Primary}
```

Run on new Primary to find unapplied THL (Scenario 3):

```
shell> tungsten_find_orphaned -v -n
```

9.26. The tungsten_find_position Command

The tungsten_find_position assists with locating event information in the THL and producing a dsctl set command as output.

The tungsten_find_position command performs the following steps:

- Get the MySQL binary log position to search for from the CLI and validate
- Validate paths and commands
- Load all available service names and validate any specified service against that list
- Check if this Replicator is in the Primary role
- Locate the supplied binary log position in the available THL
- Parse the THL found, if any
- Generate the dsctl command and display

Table 9.65. tungsten_find_position Options

Option	Description
--debug, -d	Enable additional debug output.
--file, -f	Pass the -file argument to the thl list command to make the location operation more efficient - the script will not have to parse ALL of the available THL
--help, -h	Show help text
--high, --to	Pass specified seqno to the thl command to signify the ending position to scan to. Will scan from the start of all available thl unless the --from|--low options are provided
--low, --from	Pass specified seqno to the thl command to signify the starting position to scan from. Will continue to scan to the end of all available thl unless the --to|--high options are provided
--path, -p	Specify the full path to the executable directory where the thl and trepctl commands are located.
--quiet, -q	Prevent final message from being output.
--service, -s	Specify the service name to use with the various commands.
--test, -t	Test mode - do not perform the actual update on the files
--thl	Specify the full path to the thl command executable file.
--trepctl	Specify the full path to the trepctl command executable file.
--verbose, -v	Show verbose output

Below is a sample session:

```
shell> tungsten_find_position mysql-bin.000030:0000000000001981
dsctl set -reset -seqno 4 -epoch 2 -event-id "mysql-bin.000030:0000000000001981;-1" -source-id "db1"
```

9.27. The tungsten_find_seqno Command

The tungsten_find_seqno command is a wrapper script for the tpm find-seqno command.

For usage instructions and options, see Section 10.5.10, “tpm find-seqno Command”

9.28. The tungsten_get_mysql_datadir Script

The tungsten_get_mysql_datadir command will gather and display actual running values for the data directory (datadir) checked against multiple sources (running mysql value, mysqld default value and the tungsten configuration) and then resolve any sym-links to the physical destination.

```
tungsten_get_mysql_datadir [ --debug, -d ] [ --flavor, -f ] [ --flavorful, -F ] [ --help, -h ] [ --verbose, -v ]
```

Where:

Table 9.66. tungsten_get_mysql_datadir Command-line Options

Option	Description
--debug, -d	
--flavor, -f	Also print out the MySQL server type as a tag in lower case (i.e. mysql, percona or mariadb)
--flavorful, -F	Also print out the MySQL server type (i.e. MySQL Community, Percona Server or MariaDB Server)
--help, -h	Show help text
--verbose, -v	

The sudo command is used along with the which command to located the mysqld executable so as to gather the defaults.

9.29. The tungsten_get_status Script

The tungsten_get_status command will display datasources and replicators for all nodes in all services, along with seqno and latency values.

> **Note**
>
> This script will only gather information for Standalone, Composite Active/Active or Composite Active/Passive clusters

9.30. The tungsten_get_ports Script

Table 9.67. tungsten_get_ports Options

Option	Description
--extra, -x	Display the listening IP and source pair too
--help, -h	Show help text
--internal, -i	Display only ports used for INTERNAL purposes
--list, -l	Display the static database
--showhost, -s	Display the short hostname as the first field

The tungsten_get_ports command will display the running Tungsten processes and the associated TCP ports that those processes are listening on.

Example:

```
shell> tungsten_get_ports
REPLICATOR[3678] 2112 THL
REPLICATOR[3678] 8091 Prometheus
REPLICATOR[3678] 8097 REST API
REPLICATOR[3678] 10000 RMI/JMX
REPLICATOR[3678] 10001 RMI/JMX
REPLICATOR[3678] 32000 Wrapper Liveness Checks
REPLICATOR[3678] 42679 INTERNAL RMI
MANAGER[4117] 7800 Group Communications
MANAGER[4117] 8090 REST API
MANAGER[4117] 8092 Prometheus
MANAGER[4117] 9997 RMI/JMX
MANAGER[4117] 11999 Router Gateway Port for Connector
MANAGER[4117] 12000 RMI/JMX
MANAGER[4117] 32001 Wrapper Liveness Checks
MANAGER[4117] 42957 INTERNAL RMI
MANAGER[4117] 46131 INTERNAL RMI
CONNECTOR[4551] 3100 RMI/JMX
CONNECTOR[4551] 3101 RMI/JMX
```

```
CONNECTOR[4551] 3306 MySQL R/W
CONNECTOR[4551] 3307 MySQL R/O
CONNECTOR[4551] 8093 Prometheus
CONNECTOR[4551] 8096 REST API
CONNECTOR[4551] 32002 Wrapper Liveness Checks
```

9.31. The tungsten_merge_logs Script

The tungsten_merge_logs command is designed to aid troubleshooting by consolidating the various log files into one place ordered by time.

```
tungsten_merge_logs [ --after ] [ --before ] [ --connector, --C ] [ --debug, --d ] [ --extension, --X ] [ --help, -h ] [ --log-limit, --L ] [ --manager, --M ] [ --quiet, --q ] [ --replicator, --R ] [ --stdout, --O ] [ --verbose, -v ]
```

Where:

Table 9.68. tungsten_merge_logs Command-line Options

Option	Description
--after	Discard all lines prior to {TIMESTAMP}
--before	Discard all lines after {TIMESTAMP}
--connector, --C	Include the connector log files
--debug, --d	
--extension, --X	Specify the file extension (default: log) Do NOT include the period
--help, -h	Show help text
--log-limit, --L	Specify the quantity of log files to gather (default: unlimited). When the wrapper rotates log files, it appends a period and an integer to the end of the log file name, when .1 is the newest and .2 is older and .3 older than that, etc. This parameter will gather the base log file plus limit minus one rotated files.
--manager, --M	Include the manager log files
--quiet, --q	
--replicator, --R	Include the replicator log files
--stdout, --O	Send merged logs to STDOUT instead of a file
--verbose, -v	Show verbose output

With no options specified, the tungsten_merge_logs script will gather all log files in the current directory and below.

For example:

```
shell> cd
shell> tpm diag --all
shell> tar xvzf tungsten-diag-2021-11-15-16-37-33.tgz
shell> cd tungsten-diag-2021-11-15-16-37-33
shell> tungsten_merge_logs
```

Would result in something like the following:

```
New merged log file ./merged.log created!
```

All logs files are gathered by default.

If you specify any of --connector, --replicator or --manager, then only the log files for the specifed components will be gathered.

Using multiple options will aggregate the logs from the specified components.

Use of the --log-limit option works as follows:

- a --log-limit of 1 means gather the base file only, i.e. trepsvc.log
- a --log-limit of 2 means gather the base file and the first backup file, i.e. trepsvc.log and trepsvc.log.1
- a --log-limit of 3 means gather the base file and the first two backup files, i.e. trepsvc.log, trepsvc.log.1 and trepsvc.log.2

When supplying the --before or --after options, the {TIMESTAMP} must be specified as a single argument wrapped in quotes, in the format of 'yyyy/mm/dd hh:mm:ss', including a single space between the date and time. Hours are in 24-hour time, and all values should be left-padded with zeros. For example:

```
shell> tungsten_merge_logs --before '2021/09/27 21:58:02'
```

9.32. The tungsten_monitor Script

The tungsten_monitor script provides a mechanism for monitoring the cluster state when monitoring tools like Nagios aren't available. It implements the Tungsten Script Interface as well as these additional options.

```
tungsten_monitor [ --check-log ] [ --connector-timeout ] [ --dataservices ] [ --diagnostic-package ] [ --directory ] [ --disk ] [ --elb-script ] [ --email ] [ --force ] [ --help, -h ] [ --ignore ] [ --info, -i ] [ --json ] [ --latency ] [ --lock-dir ] [ --lock-timeout ] [ --mail ] [ --max-backup-age ] [ --net-ssh-option ] [ --notice, -n ] [ --reset ] [ --subject ] [ --validate ] [ --verbose, -v ]
```

Where:

Table 9.69. tungsten_monitor Command-line Options

Option	Description
--check-log	Email any lines in the log file that match the egrep expression. --check-log=tungsten-manager/log/tmsvc.log:OFFLINE
--connector-timeout	Number of seconds to wait for a connector response
--dataservices	This list of dataservices to monitor
--diagnostic-package	Create a diagnostic package if any issues are found
--directory	The $CONTINUENT_ROOT directory to use for running this command. It will default to the directory you use to run the script.
--disk	Display a warning if any disk usage is above this percentage
--elb-script	The xinetd script name that is responding to ELB liveness checks
--email	Email address to send to when mailing any notifications
--force	Continue operation even if script validation fails
--help, -h	Show help text
--ignore	Ignore notices that use this key
--info, -i	Display info, notice, warning, and error messages
--json	Output all messages and the return code as a JSON object
--latency	The maximum allowed latency for replicators
--lock-dir	Directory to store log and lock files in
--lock-timeout	The number of minutes to sleep a notice after sending it
--mail	Path to the mail program to use for sending messages
--max-backup-age	Maximum age in seconds of valid backups
--net-ssh-option	Provide custom SSH options to use for communication to other hosts.
--notice, -n	Display notice, warning, and error messages
--reset	Remove all entries from the lock directory
--subject	Email subject line
--validate	Only run script validation
--verbose, -v	Verbose

General Operation

Each time the tungsten_monitor runs, it will run a standard set of checks. The set of checks will be determined automatically based on the current node configuration (for example, --connector-timeout check will only run if the node has a connector installed). Additional checks may be turned on using command line options.

- Check that all Tungsten services for this host are running
- Check that all replication services and datasources are ONLINE
- Check that replication latency does not exceed a specified amount
- Check that the local connector is responsive

- Check disk usage

An example of adding it to crontab:

```
shell> crontab -l
10 * * * * /opt/continuent/tungsten/cluster-home/bin/tungsten_monitor >/dev/null 2>/dev/null
```

All messages will be sent to `/opt/continuent/share/tungsten_monitor/lastrun.log`.

Note that when all tungsten_monitor checks pass, the script will not print anything to the standard output.

Sending results via email

The tungsten_monitor is able to send you an email when problems are found. It is suggested that you run the script as root so it is able to use the mail program without warnings.

Alerts are cached to prevent them from being sent multiple times and flooding your inbox. You may pass `--reset` to clear out the cache or `--lock-timeout` to adjust the amount of time this cache is kept. The default is 3 hours.

```
shell> crontab -l
10 * * * * /opt/continuent/tungsten/cluster-home/bin/tungsten_monitor --from=you@yourcompany.com \
    --to=group@yourcompany.com >/dev/null 2>/dev/null
```

Monitoring log files

The tungsten_monitor can optionally monitor log files for certain keywords. This example will alert you to any lines in `trepsvc.log` that include OFFLINE.

```
shell> tungsten_monitor --check-log=tungsten-replicator/log/trepsvc.log:OFFLINE
```

Monitoring backup status

Knowing you have a recent backup is an important part any Tungsten deployment. The tungsten_monitor will look for the latest backup across all datasources and compare it to the value `--max-backup-age`. This example will let you know if a valid backup has not been taken in 3 days.

```
shell> tungsten_monitor --max-backup-age=259200
```

Compatibility

The script only works with MySQL at this time.

9.33. The tungsten_newrelic_event Command

The tungsten_newrelic_event script utilises existing tunsten monitor scripts and inserts the results into New Relic.

`tungsten_newrelic_event` [`--account`] [`--connector`, `-c`] [`--critical`, `-C`] [`--node`, `-n`] [`--services`, `-s`] [`--curl`] [`--debug`, `-d`] [`--help`, `-h`] [`--hostname`] [`--key`] [`--latency`, `-l`] [`--noexec`] [`--online`, `-o`] [`--policy`, `-p`] [`--progress`, `-r`] [`--service`] [`--test`] [`--timeout`, `-t`]
tmpfile [`--verbose`, `--v`] [`--warn`, `-W`]

Where:

Table 9.70. tungsten_newrelic_event Command-line Options

Option	Description
`--account`	Use to spceify your New Relic Account ID
`--connector`, `-c`	Run check_tungsten_services -c to check for the Connector
`--critical`, `-C`	(default: 15) Specify the Critical alert latency level in seconds; optionally used for check_tungsten_latency
`-n`, `--node`	Run check_tungsten_services -r to check for the Manager and the Replicator
`-s`, `--services`	Run check_tungsten_services -r -c to check for the Manager, the Replicator and the Connector
`--curl`	Use to specify full path the curl binary
`--debug`, `-d`	Enabling debug also implies enabling of verbose. Debug is VERY chatty, avoid unless essential
`--help`, `-h`	
`--hostname`	Use to specify full path the hostname binary

Option	Description
`--key`	Use to specify your New Relic Insights API Insert key
`--latency`, `-l`	Run check_tungsten_latency, optionally specify --warn and --crit
`--noexec`	Do not execute the Tungsten Nagios Check script
`--online`, `-o`	Run check_tungsten_online, optionally specify a service with --service
`--policy`, `-p`	Run check_tungsten_policy
`--progress`, `-r`	Run check_tungsten_progress, optionally specify a timeout with --timeout, optionally specify a service with --service
`--service`	Specify a service name; optionally used for check_tungsten_online and check_tungsten_progress
`--test`	Do not execute the New Relic API Insert call
`--timeout`, `-t`	[default: 1] Specify the time to wait for progress to occur; optionally used for check_tungsten_progress
`tmpfile`	Use to specify full path the temp JSON file. Default: ./new_relic_event.json
`--verbose`, `--v`	
`--warn`, `-W`	[default: 5] Specify the Warning alert latency level in seconds; optionally used for check_tungsten_latency

Usage

```
shell> tungsten_newrelic_event --account {New Relic Account ID} --key {New Relic Insights API Insert Key} [args]
```

By default all of the following Nagios check scripts under `$CONTINUENT_ROOT/tungsten/cluster-home/bin` are executed and the results of each are inserted into NewRelic as the associated EventType.

Executable	EventType
check_tungsten_latency	CheckTungstenLatency
check_tungsten_online	CheckTungstenOnline
check_tungsten_policy	CheckTungstenPolicy
check_tungsten_progress	CheckTungstenProgress
check_tungsten_services `-r -c`	CheckTungstenServices
check_tungsten_services `-r`	CheckTungstenNode
check_tungsten_services `-c`	CheckTungstenConnector

If you specify a check to execute by using one or more cli args, then only those checks specified will be run.

curl must be installed and available on the PATH.

You must provide your New Relic Account ID and your New Relic Insights API Insert Key for the script to function.

You may obtain the API Insert Key at https://insights.newrelic.com/accounts/{New Relic Account ID}/manage/api_keys

9.34. The tungsten_post_process Command

The tungsten_post_process command is a wrapper script for the tpm post-process command.

For usage instructions and options, see Section 10.5.16, “tpm post-process Command”

9.35. The tungsten_prep_upgrade Script

The tungsten_prep_upgrade command is a utility script which assists in the migration process from earlier Multi-Site/Active-Active topologies to Composite Active/Active topologies.

For a full end-to-end example of using this script, see Section 6.15.4, “Upgrade/Convert: From Multi-Site/Active-Active (MSAA) to Composite Active/Active (CAA)”

```
tungsten_prep_upgrade [--all|--service] {service} [--path {fullpath_to_replicator_dir}] [arguments]
```

Where:

Table 9.71. tungsten_prep_upgrade Command-line Options

Option	Description
--alldb	Loop through all services on this node (based on existing tungsten_* schemas).
--debug	Debug mode is VERY chatty, avoid it unless you really need it.
--dump, -d	Backup tracking databases using mysqldump (default: tungsten_{service})
--dumpdb, -D	Specify database to dump
--force, -f	Force the operation.
--help, -h	Show help text
--host, -H	Specify the database hostname or IP address (Default: 127.0.0.1)
--keep, --get, -g, -k	Get the current tracking position schema for the specified service and save it as json to a text file
--offline, -o	Take the specific service to the offline-deferred state at-heartbeat (default: offline_for_upg)
--offlinehb, -O	Specify the name of the heartbeat for the offline-deferred operation
--password, -w	Specify the database password
--path, -p	Full path to the cross-site replicator directory (default: /opt/replicator)
--port, -P	Specify the database listener port (default: 13306)
--restore, -r	Load the tracking database backup for the specified service.
--service, -s	Specify the service name to act upon.
--start	Bring up the cross-site replicator process
--stop	Gracefully shut down the cross-site replicator process
--targetdir, -T	Specify directory target for dump
--user, -u	Specify the database user
--verbose, -v	Show verbose output

If --{service|all} is not specified, tungsten_prep_upgrade will attempt to derive a list of one or more service names from trepctl services.

For operations that require MySQL access, tungsten_prep_upgrade will attempt to auto-locate the database user and password from tpm reverse.

Below are various examples:

- Take all replicator services offline automatically, or take a specific service offline:

```
shell> tungsten_prep_upgrade -o
~or~
shell> tungsten_prep_upgrade --service london --offline
shell> tungsten_prep_upgrade --service tokyo --offline
```

> **Note**
>
> To invoke the actual deferred offline set with `--offline`, use the below CLUSTER-specific trepctl command (i.e. from `/opt/continuent`, not `/opt/replicator`) on the Primary hosts within each cluster:
>
> ```
> shell> trepctl heartbeat -name offline_for_upg
> ```

- Get (keep) the current tracking position schema for the specified service and save it as json to a text file (default: ~/position-{service}-YYYYMMDDHHMMSS.txt)

```
shell> tungsten_prep_upgrade -g
~or~
shell> tungsten_prep_upgrade --service nyc --get
(NOTE: saves to ~/position-nyc-YYYYMMDDHHMMSS.txt)
shell> tungsten_prep_upgrade --service tokyo --get
(NOTE: saves to ~/position-tokyo-YYYYMMDDHHMMSS.txt)
```

- Gracefully shut down the cross-site replicator process:

```
shell> tungsten_prep_upgrade --stop
```

- Backup (dump) tracking databases using mysqldump (default: tungsten_{service})

```
shell> tungsten_prep_upgrade -d --alldb
~or~
shell> tungsten_prep_upgrade --service london --dump
shell> tungsten_prep_upgrade --service tokyo --dump
```

- Load (restore) the tracking database backup for the specified service. --all is unavailable with --restore.

```
shell> tungsten_prep_upgrade -s nyc -u tungsten -w secret -r
shell> tungsten_prep_upgrade -s tokyo -u tungsten -w secret -r
~or~
shell> tungsten_prep_upgrade --service nyc --user tungsten --password secret --restore
shell> tungsten_prep_upgrade --service tokyo --user tungsten --password secret --restore
```

> **Note**
>
> A restore may take place after the cross-site replicator is uninstalled, and so certain information is required on the command line (i.e. service, user and password)

9.36. The tungsten_purge_thl Command

The tungsten_purge_thl command is a wrapper script for the tpm purge-thl command.

For usage and examples, see Section 10.5.18, “tpm purge-thl Command”

9.37. The tungsten_reset_manager Command

The tungsten_reset_manager assists with the graceful reset of the manager's dynamic state files on disk.

Table 9.72. tungsten_reset_manager Options

Option	Description
`--debug`, `-d`	Enable additional debug output.
`--force`, `-f`	Bypass manager and policy checks
`--help`, `-h`	Show help text
`--list`, `-l`	Prints out path(s) to be cleared. One per line.
`--start`, `-s`	Restarts the Manager process.
`--verbose`, `-v`	Show verbose output
`--yes`, `-y`	Bypass interactive confirmation prompt

- The tungsten_reset_manager command stops the manager and then deletes the manager's dynamic state files.
- For a proper service-wide reset to work, this command needs to be run on every node within the local cluster to ensure all managers have been stopped at the same time. This prevents any manager from storing an old state in memory.
- The dynamic state files are automatically rebuilt when the manager is started.
- The `{TUNGSTEN_HOME}/tungsten/cluster-home/conf/cluster/{SERVICE_NAME}/datasource/` directory needs to be emptied, for example, given a service name of `alpha` and a TUNGSTEN_HOME of `/opt/continuent`, the full path would be `/opt/continuent/tungsten/cluster-home/conf/cluster/alpha/data-source/`.
- The tungsten_reset_manager command requires the cluster to be in *`MAINTENANCE`* mode first.

```
shell> tungsten_reset_manager
Policy is not MAINTENANCE.

Please select one node only and execute `echo "set policy maintenance" | cctrl`

If you are sure the policy is maintenance you may proceed by using -f or --force
```

- By default, the command expects the manager to be running so the policy may be checked. If you are sure the policy is *`MAINTENANCE`* you may proceed by using ether the `-f` or `--force` flags.

Below is a sample session.

First, set policy to *MAINTENANCE* on a single node only:

```
shell> echo "set policy maintenance" | cctrl
Tungsten Clustering 7.0.3 build 141
north: session established, encryption=false, authentication=false
[LOGICAL] /alpha > set policy maintenance
policy mode is now MAINTENANCE
[LOGICAL] /alpha >
Exiting...
```

Warning

Enable *MAINTENANCE* mode on a single host only.

On all nodes in the local cluster, execute the tungsten_reset_manager command:

```
shell> tungsten_reset_manager
About to stop the manager and empty directory:
   /opt/continuent/tungsten/cluster-home/conf/cluster/north/datasource

Do you wish to continue? [y/N] y
Stopping Tungsten Manager Service...
Stopped Tungsten Manager Service.
Manager directory /opt/continuent/tungsten/cluster-home/conf/cluster/north/datasource cleared.

Done!

Make sure all nodes have been cleared first, then execute `manager start` on each node one-by-one, starting with the Primary.

When all managers are running, pick one node and execute `echo "set policy automatic" | cctrl`
```

Once the tungsten_reset_manager command has completed successfully on all nodes in the local cluster, execute the manager start command on all nodes:

```
shell> manager start
Starting Tungsten Manager Service...
Waiting for Tungsten Manager Service.........
running: PID:3562
```

Once themanager has started successfully on all nodes in the local cluster, set policy to *AUTOMATIC* on a single node only:

```
shell> echo "set policy automatic" | cctrl
Tungsten Clustering 7.0.3 build 141
north: session established, encryption=false, authentication=false
[LOGICAL] /alpha > set policy automatic
policy mode is now AUTOMATIC
[LOGICAL] /alpha >
Exiting...
```

Warning

Enable *AUTOMATIC* mode on a single host only.

At this point, the manager state should be completely back to normal. Check it using the cctrl ls command.

9.38. The tungsten_send_diag Script

The tungsten_send_diag command is a utility script which assists in the upload of files to Continuent support.

tungsten_send_diag may be used in place of the Section 10.5.7, “tpm diag Command” to generate a diagnostic package.

`tungsten_send_diag [ --args, -a ] [ --case, -c ] [ --cleanup ] [ --contentType ] [ --debug ] [ --diag, -d ] [ --email, -e ] [ --file, -f ] [ --help, -h ] [ --tpm, -t ] [ --verbose, -v ]`

Where:

Table 9.73. tungsten_send_diag Command-line Options

Option	Description
`--args, -a`	Specify arguments to be passed to tpm diag. Arguments must be passed in quotes. Requires the --diag option.
`--case, -c`	Specify the support case number
`--cleanup`	When specified, will cause the removal of the diagnostic archive file generated by the --diag argument.

Option	Description
`--contentType`	Specify the Content-Type for a file you are uploading
`--debug`	Debug mode is VERY chatty, avoid it unless you really need it.
`--diag`, `-d`	Automatically generate a tpm diag zip file and upload it
`--email`, `-e`	Email address to embed into the uploaded file name
`--file`, `-f`	File name to upload
`--help`, `-h`	Show help text
`--tpm`, `-t`	Full path to the tpm command you wish to use to execute a tpm diag
`--verbose`, `-v`	Show verbose output

You must specify either `--diag`, `--tpm`, or `--file`, but not both. For example:

```
shell> tungsten_send_diag --diag -c 1234
```

To have tpm diag gather all nodes, add the `--args '--all'`, for example:

```
shell> tungsten_send_diag --diag -c 1234 --args '--all'
```

You must specify either `--email` or `--case`, and you may provide both if you wish. For example:

```
shell> tungsten_send_diag -f example.zip -e you@yourdomain.com -c 1234
```

Using `--tpm` to specify one or more tpm commands implies the `--diag` option, you do not need to specify `--diag` if you use `--tpm` (or `-t`). For example:

```
shell> tungsten_send_diag -c 1234 -t /opt/replicator/tungsten/tools/tpm
```

You may generate multiple diags by specifying multiple tpm binaries with multiple arguments, i.e.:

```
shell> tungsten_send_diag -c 1234 -t /opt/continuent/tungsten/tools/tpm -t /opt/replicator/tungsten/tools/tpm
```

9.38.1. Updating tungsten_send_diag

> **Note**
>
> If you are unable to use tungsten_send_diag due to network rules and restrictions within your organisation, then you do not need to complete these steps.

Continuent is committed to implementing security best practices. To safeguard your uploaded diagnostic data, we routinely rotate the access keys used for upload functionality in our products. As a result, we regularly release a new version of tungsten_send_diag outside of our normal software release cycle.

Updated releases are made available via the download portal here

To install the new file:

1. Download the latest file from the download portal.
2. Upload the file to all hosts running Tungsten software. A good suggestion would be to place the file into `$HOME` path of the tungsten OS user.
3. Ensure the ownership of the file is correct

   ```
   shell> sudo chown tungsten: tungsten_send_diag
   ```
4. Copy the new tungsten_send_diag file to the running directory tree.

   Determine the running directory location, normally the `/opt/continuent` directory.

   The value of the running directory location is determined by either option `home-directory` [494] or `install-directory` [494] within your `tungsten.ini` file (they do the same thing).

   Another way to easily determine the running directory location is to check the value of the shell variable `$CONTINUENT_ROOT` as the tungsten OS user, for example:

   ```
   shell> su - tungsten
   shell> echo $CONTINUENT_ROOT
   ```

```
/opt/continuent
```

5. Copy the file to the running path of the Tungsten deployment determined above:

```
shell> su - tungsten
shell> cp ~/tungsten_send_diag /opt/continuent/tungsten/tungsten-replicator/bin
```

6. Copy the new tungsten_send_diag file to the software staging directory tree.

> **Note**
>
> This step will ensure that the new version of tungsten_send_diag is preserved should you execute ./tools/tpm update --replace-release from the staging directory during subsequent updates.

Determine the software staging directory, which by default would be under `/opt/continuent/software/`. If unsure, the value of this is determined by executing tpm query staging, for example:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141
```

For convenience, you can display only the needed staging directory path:

```
shell> tpm query staging | cut -d: -f2
/opt/continuent/software/tungsten-clustering-7.0.3-141
```

Now copy the file to the software staging path determined above:

```
shell> cp ~/tungsten_send_diag /opt/continuent/software/tungsten-clustering-7.0.3-141/tungsten-replicator/bin/
```

> **Note**
>
> If upgrading to a new version of Tungsten that does not include the latest version of tungsten_send_diag, the above steps will need to be carried out post upgrade.

9.39. The tungsten_show_processlist Script

The tungsten_show_processlist is a read-only script which will show the output from tungsten show processlist on all available Connectors.

This requires the mysql command-line client and a running manager for cctrl to poll.

`tungsten_show_processlist [ -a, --all ] [ -d, --debug ] [ -f, --filter ] [ -g, --grep ] [ -h, --help ] [ -m, --manager ] [ -p, --port ] [ -v, --verbose ]`

Where:

Table 9.74. tungsten_show_processlist Command-line Options

Option	Description
`--all`, `-a`	Display disconnected users
`--debug`, `-d`	[-d implies -v also] Debug mode is VERY chatty, avoid it unless you really need it.
`--filter`, `-f`	Specify field/regex pairs as field=regex separated by pipe symbols, like this: --filter 'User=app_user|db=db7.*'. Available Fields: connector, DataSource, Id, User, Host, db, Command, Time, State, Info
`--grep`, `-g`	Filter by regex on entire line, case insensitive only
`--help`, `-h`	
`--manager`, `-m`	Specify maanager host for cctrl to poll. Default: localhost
`--port`, `-p`	Provide the connector port number
`--verbose`, `-v`	

9.40. The tungsten_skip_all Command

The tungsten_skip_all command assists with skipping replicator errors that you deem safe to skip.

`tungsten_skip_all [ --auto ] [ --debug, -d ] [ --help, -h ] [ --info, -i ] [ --max, -m ] [ --path, -p ] [ --quiet, -q ] [ --service, -s ] [ --trepctl ] [ --verbose, -v ]`

Where:

Table 9.75. tungsten_skip_all Options

Option	Description
--auto	Automatically skip and seqno is an error state - ONLY USE THIS WHEN ASKED TO BY CONTINUENT SUPPORT
--debug, -d	Debug mode if very chatty. Avoid it unless you really need to. Implies --verbose and --info.
--help, -h	
--info, -i	Displays additional information, but not fully verbose.
--max, -m	Specify the maimum number of iterations.
--path, -p	Supply full path to trepctl and thl executables if NOT installed in default directory
--quiet, -q	
--service, -s	Supply service name to use - Required if more than one replication service running
--trepctl	Supply name of trepctl executable if different from default.
--verbose, -v	Displays additional information. Implies --info

Warning

Blindly skipping replication errors without fully understanding the consequences could lead to data drift. This action should only be performed providing a full understanding of the error has been analysed and deemed to be safe to skip by yourself and/or you business.

The tungsten_skip_all command performs the following steps:

- Gather a list of replicator service names using trepctl services | grep serviceName.
- Starts an inifinite loop.
- Loops through all services, or uses the service specified on the cli.
- Checks the service status via trepctl status -json [391].
- If the pendingErrorSeqno is not -1, then processes the error state.
- By default, if there is an error condition, a detailed message is displayed, and the user may skip the seqno interactively.
- If the tungsten_skip_all command is called with --auto then the seqno with the error will be skipped automatically.
- If the maximum number of loops has been reached (default: 100), the script will exit.
- Sleeps for 3 seconds by default, then iterate

9.41. The tungsten_skip_seqno Script

The tungsten_skip_seqno allows events to be skipped based on filters, allowing the Tungsten Replicator to come back online with less manual intervention.

tungsten_skip_seqno [--auto] [--debug, -d] [--filter, -f] [--filters] [--help, -h] [--info, -i] [--max, -m] [--path, -p] [--quiet, -q] [--service, -s] [--tpm] [--trepctl] [--verbose, -v]

Where:

Table 9.76. tungsten_skip_seqno Command-line Options

Option	Description
--auto	Automatically skip when seqno is an error state - ONLY USE THIS WHEN ASKED TO BY CONTINUENT SUPPORT
--debug, -d	Implies -v and -i also. Debug mode is VERY chatty, avoid it unless you really need it.
--filter, -f	Specify a single regex to match in the pendingExceptionMessage. If a filter is specified, then the new behavior will be to skip if the

Option	Description
	regex matches the pendingExceptionMessage, and do not not skip if there is no match. The regex will be matched against the pendingExceptionMessage like this: /yourRegexValue/gm Example, to match foreign key errors in a specific table: foreign key constraint fails.*?`yourSchemaName`.`yourTableName`
`--filters`	Specify a plain text file with one or more regexes, one per line, to match in the pendingExceptionMessage.
`--help`, `-h`	
`--info`, `-i`	Display additional information, but not fully verbose
`--max`, `-m`	Specify the maximum number of iterations
`--path`, `-p`	Specify the path to the trepctl executable if not installed in default location.
`--quiet`, `-q`	
`--service`, `-s`	Specify the {service_name} to act against if more than one service running
`--tpm`	Specify tpm executable name and path if changed from default.
`--trepctl`	Specify trepctl executable name and path if changed from default.
`--verbose`, `-v`	Display additional information (implies --info)

General Operation

By default, the tungsten_skip_seqno command will:

- Gather a list of replicator service names using trepctl services `| grep serviceName`
- Start an inifinite loop
- Loop thru all services or use the service specified on the cli using the `--service` option
- Check the service status via trepctl status -json [391]
- If the `pendingErrorSeqno` is not -1, then process the error state
- By default, if there is an error condition, a detailed message is displayed, and the user may skip the seqno interactively
- if the tungsten_skip_seqno command is called with `--auto` then the seqno with the error will be skipped automatically
- if the maximum number of loops has been reached (default: 100), the script will exit. Use `--max` to adjust this value
- Sleep for 3 seconds by default, then iterate

9.42. The undeployall Command

The undeployall command removes startup the startup and reboot scripts crteated by deployall, disabling automatic startup and shutdown of available services.

> **Important**
>
> If you have the `install=true` [494] tpm option in your configuration, then you should either remove this entry or set the value to `false` to ensure the startup scripts are not re-enabled during a future tpm update

To enable the scripts on the system, use deployall.

9.43. The zabbix_tungsten_latency Command

The zabbix_tungsten_latency command reports a 0 (False) or 1 (True) depending on whether the latency across the nodes in the cluster is above a specific level.

Table 9.77. zabbix_tungsten_latency Options

Option	Description
`-c`	Report a critical status if the latency is above this level

Option	Description
-w	Report a warning status if the latency is above this level

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> zabbix_tungsten_latency -w 0.1 -c 0.5
0
```

9.44. The zabbix_tungsten_online Command

The zabbix_tungsten_online command checks whether all the services for a given service and host are online and running.

Within a Tungsten Cluster service, the replicator, manager and connector services are checked. All must be online for a 0 (True) response.

Table 9.78. zabbix_tungsten_online Options

Option	Description
-c	Enable composite dataservice checks (default: disabled)
-d	Debug mode (enables verbose mode also)
-h	Display the help text
-a	Check manager services on all nodes, not just localhost.
-port	RMI port for the replicator being checked
-r	Disable replicator checks (default: enabled)
-s	Specify the service you would like to check (default: all available services)
-v	Verbose mode.

By default, the script will check all manager and replication services for the localhost

To also check the manager services on the other cluster nodes, use -a

You can also check the cluster-wide composite status using -c

> **Warning**
>
> Using -a or -c on multiple nodes will alert on all monitored nodes for the same offline service

The command outputs a 0 (True) or a 1 (False) status.

For example:

```
shell> zabbix_tungsten_online
0
```

If you have multiple services installed, use the -s to specify the service:

```
shell> zabbix_tungsten_online -s alpha
0
```

9.45. The zabbix_tungsten_progress Command

The zabbix_tungsten_progress command determines whether the replicator is actually making progress by executing a heartbeat operation and monitoring for this operation to complete within an optional time period (default is 1 second).

Table 9.79. zabbix_tungsten_progress Options

Option	Description
-s	Service name to check (optional)
-t	Give a time period during which progress should be identified

The command outputs a 0 (True) or 1 (False) Status

The time delay can be added on busy systems to ensure that the replicator is progressing

For example, to wait 5 seconds to ensure the replicator is progressing:

```
shell> zabbix_tungsten_progress  -t 5
0
```

Optionally, specify the replication service name using the `-s` option. This is normally only needed for Composite Active/Active or Composite Dynamic Active/Active deployments where there is no single default replication service.

```
shell> zabbix_tungsten_progress -t 5 -s east_from_west
0
```

9.46. The zabbix_tungsten_services Command

The zabbix_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 9.80. check_tungsten_services Options

Option	Description
`-c`	Check the Connector service status.
`-h`	Display the help text.
`-r`	Check the replication services status.

The command outputs a 0 (True) or 1 (False) Status

> **Note**
>
> The zabbix_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 zabbix_tungsten_online command.

To check the services:

- To check the replicator and manager services are executing:

```
shell> zabbix_tungsten_services -r
0
```

- To check the connector services:

```
shell> zabbix_tungsten_services -c
0
```

Chapter 10. The tpm Deployment Command

tpm, or the Tungsten Package Manager, is a complete configuration, installation and deployment tool for Tungsten Clustering. It includes some utility commands to simplify those and other processes. In order to provide a stable system, all configuration changes must be completed using tpm. tpm makes use of ssh enabled communication and the sudo support as required by the Appendix B, *Prerequisites*.

For deployments, tpm operates by use of an INI file, typically `/etc/tungsten/tungsten.ini`.

tpm uses the `INI` file to configure the service on the local host. The `INI` file must be created on each host that will run Tungsten Clustering. tpm only manages the services on the local host; in a multi-host deployment, upgrades, updates, and configuration must be handled separately on each host.

> **Important**
>
> In older releases, a Staging deployment method was available.
>
> For backwards compatability, this is still supported however it will be deprecated in a future release and therefore it is not recommended for new deployments. Existing deployments should be converted when possible. See Section 10.3, "Converting from Staging to INI" for details on converting.

Within a staging configuration a tpm configuration is created by defining the command-line arguments that define the deployment type, structure and any additional parameters. tpm then installs all the software on all the required hosts by using ssh to distribute Tungsten Clustering and the configuration, and optionally automatically starts the services on each host. tpm manages the entire deployment, configuration and upgrade procedure.

During installation and updates, the tpm tool works as follows:

- tpm reads the 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.
- Each host is accessed if possible (using ssh), and various checks are performed, for example, checking database configuration, whether certain system parameters match required limits, and that the environment is suitable for running Tungsten Clustering.
- During an installation or upgrade, tpm copies the software to the `releases` direcotry within the location defined by `install-directory` [494].
- During a new installation, a number of other directories are created, and the release is symlinked to the core running directory `tungsten`
- 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.
- The components of Tungsten Clustering are then started (installation) or restarted according to the configuration options.

This method of operation ensures:

- Active configurations and properties are not updated until validation is completed. This prevents a running installation from being affected by an incompatible or potentially dangerous change to the configuration.
- Services are not stopped/restarted unnecessarily.
- During an upgrade or update, the time required to reconfigure and restart is kept to a minimum.

Because of this safe approach to performing configuration, downtime is minimized, and the configuration is always based on files that are separate from, and independent of, the live configuration.

> **Important**
>
> tpm always creates the active configuration from the combination of the template files and parameters given to tpm. This means that changes to the underlying property files within the configuration are overwritten by tpm when the service is configured or updated.

In addition to the commands that tpm supports for the installation and configuration, the command also supports a number of other utility and information modes, for example, the tpm reverse and tpm query commands return 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"; and configuration options, which determine the parameters that configure individual services, which are detailed in Section 10.8, "tpm Configuration Options".

10.1. Processing Installs and Upgrades

The tpm command is designed to simplify the deployment activity across all hosts in a dataservice. These operations can be executed on each host in parallel to speed up the process. The general processis as follows:

- 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. It is always advisable to understand and fix any errors that occur, but should you wish to ignore them, you can skip them by adding the `skip-validation-check=ChecktoSkip` [444]. When re-running the tpm install command this check will be bypassed. The names of the checks are always displayed as part of the error in parenthesis, for example, the above error is fired by the validation check called `MySQLConnectorPermissionsCheck`

  A full list of the validation checks can be viewed at Section 10.7, “tpm Validation Checks”

- Deploy and write configuration files

  If validation is successful, we will move on to deploying and writing the actual configuration files. The tpm command uses a JSON file that summarizes the configuration. The Tungsten processes use many different files to store the configuration and tpm is responsible for writing them.

  The `/opt/continuent/releases` directory will start to collect multiple directories after you have run multiple upgrades. We keep the previous versions in case a downgrade is needed or for review at a later date. If your upgrade has been successful, you can remove old directories using the clean_release_directory command. If you remove them manually, make sure you do not remove the directory that is linked to by the `/opt/continuent/tungsten` symlink as this is the current active release.

  > **Note**
  >
  > Do not change configuration files by hand other than the `/etc/tungsten/tungsten.ini`. This will cause future updates to fail. One of the validation checks compares the file that tpm has written with the current file. If there are differences, validation will fail.
  >
  > This is done to make sure that any configuration changes made by hand are not wiped out without giving you a chance to save them. You can run tpm query modified-files to see what, if any, changes have been made.

- Start Tungsten services

  After the installation is fully configured, the tpm command will start services on all of the hosts if the tpm option `start` [514] or `start-and-report` [514] was set. This process is slightly different depending on if you are doing a clean install or and upgrade.

  - Install

    1. Check if `--start` [514] or `--start-and-report` [514] were provided in the configuration
    2. Start the Tungsten Replicator and Tungsten Manager on all hosts
    3. Wait for the Tungsten Manager to become responsive
    4. Start the Tungsten Connector on all hosts

- Upgrade

  1. Put all dataservices into *MAINTENANCE* mode
  2. Stop the Tungsten Replicator and Tungsten Manager on all nodes
  3. Start the Tungsten Replicator and Tungsten Manager on all hosts if the services were previously running
  4. Wait for the Tungsten Manager to become responsive
  5. Stop the old Tungsten Connector and Start the new Tungsten Connector. If `--no-connectors` [504] was provided on the command line then this will not occur. You must go to each server running Tungsten Connector and run tpm promote-connector.

10.2. tpm INI File Configuration

tpm uses an INI file to manage host configuration. This is the preferred deployment method and should be used instead of the Staging deployment method for new installations.

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.

10.2.1. Creating an INI file

When using an INI configuration, installation and updates will still be done using the tpm command. Instead of providing configuration information on the command line, the tpm command will look for an INI file in three files:

1. `$HOME/tungsten.ini`.
2. `/etc/tungsten/tungsten.ini`
3. `/etc/tungsten.ini`

tpm will automatically search all `tungsten*.ini` files within the `/etc/tungsten` directory.

An alternative directory can be searched using `--ini` [443] option to tpm. This option can also be used to specify a specific ini file if you choose to name the file something different, for example `--ini /my/directory/myconfig.ini`

The INI file(s) must be readable by the tungsten system user.

> **Warning**
>
> If multiple files are found that match the default search criteria they will all be read. This can present unexpected results.
>
> The tpm output will notify if more than one file is being read. This situation should be avoided.

Here is an example of a `/etc/tungsten/tungsten.ini` file that would setup a simple dataservice.

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

```
[defaults]
user=tungsten
install-directory=/opt/continuent
profile-script=~/.bash_profile
replication-user=tungsten
replication-password=password
replication-port=13306
application-user=app_user
application-password=secret
application-port=3306
rest-api-admin-user=apiuser
rest-api-admin-password=secret
install=true
connector-rest-api-address=0.0.0.0
manager-rest-api-address=0.0.0.0
replicator-rest-api-address=0.0.0.0

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

```
connectors=host4
```

The section name inside of the square brackets is used as the service name. The `[defaults]` section should always be included and is especially useful if defining multiple services in a single file where you want certain options to apply globally.

Service names should be Alphanumeric characters only, and may include a hyphen (-) or an underscore (_). No other special characters should be used. Whilst there is no enforced max length of string you can define, for simplicity and easy administration, service names shoud be kept short and meaningful, ideally 15 characters or less.

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.

10.2.2. Installation with INI File

Once you have created the `tungsten.ini` file, the tpm command will recognize it and use it for configuration. Unpack the software into `/opt/continuent/software` and run the tpm install command.

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> ./tools/tpm install
```

or

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> ./tools/tpm install --ini /my/directory/myconfig.ini
```

The tpm command will read the `tungsten.ini` file and setup all dataservices on the current server.

10.2.3. 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-clustering-7.0.3-141.tar.gz
shell> cd tungsten-clustering-7.0.3-141
shell> ./tools/tpm update --replace-release
```

Important

The use of `--replace-release` [443] is not mandatory for minor configuration changes, however it is highly recommended.

Using this option will ensure that underlying metadata and property files are cleanly rebuilt, thus ensuring any new or deprecated properties between releases are correctly added/removed acordingly.

After unpacking the new software into the staging directory, the tpm update command will read the `tungsten.ini` configuration and install the new software. All services will be stopped and the new services will be started.

Note

The tpm update command may cause a brief outage while restarting the connectors. This will occur if you are upgrading to a new version You can avoid that with:

```
shell> ./tools/tpm update --replace-release --no-connectors
```

Afterwards, the connectors must still be restarted on each server to pick up the new version or any changes, this can be done when convenient, by running:

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

During the lifetime of the cluster, switches may happen and the current Primary may well be a different node than what is reflected in the static ini file in the `master` [499] line. Normally, this difference is ignored during an update or an upgrade.

However, if a customer has some kind of procedure (i.e. automation) which hand-edits the ini configuration file `master` [499] line at some point, and such hand-edits do not reflect the current reality at the time of the update/upgrade, an update/upgrade will fail and the cluster may be left in an indeterminate state.

Warning

The best practice is to NOT change the `master` [499] line in the INI configuration file after installation.

There is still a window of opportunity for failure. The update will continue, passing the `CurrentTopologyCheck` test and potentially leaving the cluster in an indeterminate state if the `master` [499] option is set to a hostname that is not the current Primary or the current host.

10.2.4. 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
```

If you configured the `profile-script` [507] option in your INI file, then the environment will be configured with aliases and PATHS set, therefore you can also simply execute:

```
shell> tpm update
```

If you are making significant changes such as adding/removing nodes or you are changing hostnames, then it is recommended to execute the update from tjhe software staging path to ensure all metadata files are updated correctly, for example:

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> ./tools/tpm update --replace-release
```

This will read the `tungsten.ini` file and apply the settings. The tpm command will identify what services likely need to be restarted and will just restart those. You can manually restart the desired services if you are unsure if the new configuration has been applied.

Note

The tpm update command may cause a brief outage while restarting the connectors. This will occur if you are upgrading to a new version You can avoid that with:

```
shell> ./tools/tpm update --replace-release --no-connectors
```

Afterwards, the connectors must still be restarted on each server to pick up the new version or any changes, this can be done when convenient, by running:

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

10.3. Converting from Staging to INI

If you currently use the Staging installation method and wish to convert to using INI files, use the following procedure.

You can also try using the script in Section 10.3.1, “Using the `translatetoini.pl` Script”.

1. Place cluster[s] in *`MAINTENANCE`* Mode:

   ```
   shell> cctrl
   cctrl> set policy maintenance
   ```

2. Create the text file `/etc/tungsten/tungsten.ini` on each node. They should all be the same, therefore it is advised to create the file on one host and then copy it to all other hosts.

   ```
   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 `/etc/tungsten/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 vi:

   ```
   shell> vi /etc/tungsten/tungsten.ini
   :%s/tools\/^tpm configure /[/g
   ```

```
:%s/^--//g
:%s/\s*\\$//g
```

> **Important**
>
> In the above example, you MUST manually add the trailing square bracket] to the end of the defaults tag and to the end of every service name section. Just search for the opening square bracket [and make sure there is a matching closing square bracket for every one.

See Section 10.2.1, "Creating an INI file" for more information.

3. On every node, extract the software into `/opt/continuent/software/{extracted_dir}`

   > **Warning**
   >
   > Make sure you have the same release that is currently installed.

   ```
   shell> cd /opt/continuent/software
   shell> tar zxf tungsten-clustering-7.0.3-141.tar.gz
   ```

   > **Important**
   >
   > If this is an Multi-Site/Active-Active topology, make sure you extract both the clustering and replication packages.

4. On each node, change to the extracted directory and execute the tpm command:

   > **Warning**
   >
   > Execute this step on the Replicas first, then switch the Primary - this procedure will restart the Tungsten services so switch your Primary to avoid interruption of service. See Section 6.16.3, "Performing Maintenance on an Entire Dataservice" for more information.

   ```
   shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
   shell> ./tools/tpm update --replace-release
   ```

   This will read the `tungsten.ini` file and apply the settings. The tpm command will identify what services likely need to be restarted and will just restart those. You can manually restart the desired services if you are unsure if the new configuration has been applied.

   > **Note**
   >
   > The tpm update command may cause a brief outage while restarting the connectors. This will occur if you are upgrading to a new version You can avoid that with:
   >
   > ```
   > shell> ./tools/tpm update --replace-release --no-connectors
   > ```
   >
   > Afterwards, the connectors must still be restarted on each server to pick up the new version or any changes, this can be done when convenient, by running:
   >
   > ```
   > shell> tpm promote-connector
   > ```

5. If you have an Multi-Site/Active-Active topology, you must also update the cross-site replicators:

   On each node, change to the extracted replicator directory and execute the tpm command:

   ```
   shell> cd /opt/continuent/software/tungsten-replicator-7.0.3-141
   shell> ./tools/tpm update --replace-release
   ```

6. Once all steps have been completed and the cluster(s) are stable, take each cluster out of *`MAINTENANCE`* mode by setting the policy back to *`AUTOMATIC`*:

   ```
   shell> cctrl
   cctrl> set policy automatic
   ```

10.3.1. Using the `translatetoini.pl` Script

You can download a script from the documentation library, translatetoini.pl. You must have a copy of Perl installed to be able to execute the script.

To use the script, you can either run the script and paste in the staging output, or pipe the output from tpm reverse directly into the script. When supplying the staging output, you should supply the output from within the configured staging directory. For example:

```
shell> ./tools/tpm reverse|../translatetoini.pl
```

The script will create the file `tungsten.ini` in the current directory containing the converted output.

To change the destination, use the `--filename` option:

```
shell> ./tools/tpm reverse|../translatetoini.pl --filename=t.ini
```

You can also combine multiple staging configurations into a single INI conversion by appending to an existing INI file by adding the `--append` option:

```
shell> ./tools/tpm reverse|../translatetoini.pl --append
```

You should always check the INI file before using it for a live installation to ensure that all of the options and parameters have been identified and configured properly.

A training video is available on how to perform the staging to INI file conversion using the translatetoini.pl script:

Click here for a video of the INI conversion procedure, showing the full process from start to finish...

10.4. tpm Staging Configuration

> **Important**
>
> For backwards compatability, the staging deployment method is still supported however it will be deprecated in a future release and therefore it is not recommended for new deployments. Existing deployments should be converted when possible. See Section 10.3, "Converting from Staging to INI" for details on converting.

Before installing your hosts, you must provide the desired configuration. This will be done with one or more calls to tpm configure as seen in Chapter 2, *Deployment*. These calls place the given parameters into a staging configuration file that will be used during installation. This is done for dataservices, composite dataservices and replication services.

Instead of a subcommand, tpm configure accepts a service name or the word `defaults` as a subcommand. This identifies what you are configuring.

When configuring defaults, the defaults affect all configured services, with individual services able to override or set their own parameters.

```
shell> tpm configure [service_name|defaults] [tpm options] [service configuration options]
```

In addition to the Section 10.8, "tpm Configuration Options", the common options in Table 10.14, "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 Cluster, the tpm command will only use `$CONTINUENT_PROFILES`.

If you are running Tungsten Replicator, the tpm command will use `$REPLICATOR_PROFILES` if it is available, before using `$CONTINUENT_PROFILES`.

10.4.1. 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.

10.4.2. 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` [499], `--slaves` [514] and `--master` [499]
>
> 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` [499] and `--slaves` [514]; `--members` [499] will be calculated as the unique join of both values.
>
> If you specify `--master` [499] and `--members` [499]; `--slaves` [514] will be calculated as the unique difference of both values.

10.4.3. Configuring a single host

```
shell> ./tools/tpm configure alpha \
    --hosts=host3 \
    --backup-method=xtrabackup-incremental
```

This will apply the `--repl-backup-method` [475] option to just the host3 server. Multiple hosts may be given as a comma-separated list. The names used in the `--members` [499], `--slaves` [514], `--master` [499], `--connectors` [484] options should be used when calling `--hosts` [494]. These values will override any given in tpm configure defaults or tpm configure alpha.

10.4.4. Reviewing the current configuration

You may run the tpm reverse command to review the list of configuration options. This will run in the staging directory and in your installation directory. It is a good idea to run this command prior to installation and upgrades to validate the current settings.

```
shell> ./tools/tpm reverse
# Defaults for all data services and hosts
tools/tpm configure defaults \
--application-password=secret \
--application-port=3306 \
--application-user=app \
--replication-password=secret \
--replication-port=13306 \
--replication-user=tungsten \
--start-and-report=true \
--user=tungsten
# Options for the alpha data service
tools/tpm configure alpha \
--connectors=host1,host2,host3 \
--master=host1 \
--members=host1,host2,host3
```

The output includes all of the tpm configure commands necessary to rebuild the configuration. It includes all default, dataservice and host specific configuration settings. Review this output and make changes as needed until you are satisfied.

10.4.5. 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 10.1, “Processing Installs and Upgrades”. This will include the full set of `--members` [499], `--slaves` [514], `--master` [499], and `--connectors` [484].

10.4.5.1. 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 10.1, “Processing Installs and Upgrades”.

10.4.5.2. 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 10.1, “Processing Installs and Upgrades”.

10.4.6. Upgrades from a Staging Directory

This process must be run from the staging directory in order to run properly. Determine where the current software was installed from.

```
shell> tpm query staging
tungsten@staging-host:/opt/continuent/software/tungsten-clustering-7.0.3-141
```

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-clustering-7.0.3-141.tar.gz
shell> cd tungsten-clustering-7.0.3-141
```

Warning

Before performing and upgrade, please ensure that you have checked the Appendix B, *Prerequisites*, as software and system requirements may have changed between versions and releases.

Before any update, the current configuration must be known. If the `$CONTINUENT_PROFILES` or `$REPLICATOR_PROFILES` environment variables were used in the original deployment, these can be set to the directory location where the configuration was stored.

Alternatively, the update can be performed by fetching the existing configuration from the deployed directory by using the tpm fetch command:

```
shell> ./tools/tpm fetch --reset --directory=/opt/continuent \
    --hosts=host1,autodetect
```

This will load the configuration into the local staging directory. Review the current configuration before making any configuration changes or deploying the new software.

```
shell> ./tools/tpm reverse
```

This will output the current configuration of all services defined in the staging directory. You can then make changes using tpm configure before pushing out the upgrade. Run tpm reverse again before tpm update to confirm your changes were loaded correctly.

```
shell> ./tools/tpm configure service_name ...
shell> ./tools/tpm update --replace-release
```

Important

The use of `--replace-release` is not mandatory for minor configuration changes. however it is highly recommended when upgrading between versions.

Using this option will ensure that underlying metadata and property files are cleanly rebuilt, thus ensuring any new or deprecated properties between releases are correctly added/removed acordingly.

This will update the configuration file and then push the updates to all hosts. No additional arguments are needed for the tpm update command since the configuration has already been loaded.

Note

The tpm update command may cause a brief outage while restarting the connectors. This will occur if you are upgrading to a new version You can avoid that with:

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

The connectors must be updated separately on each server by running:

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

The tpm command will use connector graceful-stop 30 followed by connector start [334] when upgrading versions. If that command fails then a regular connector stop [335] is run. This behavior is also applied when using tools/tpm update --replace-release.

10.4.7. Configuration Changes from a Staging Directory

Where, and how, you make configuration changes depends on where you want the changes to be applied.

Making Configuration Changes to the Current Host

You may make changes to a specific host from the `/opt/continuent/tungsten` directory.

```
shell> ./tools/tpm update service_name --thl-log-retention=14d
```

This will update the local configuration with the new settings and restart the replicator. You can use the tpm help update command to see which components will be restarted.

```
shell> ./tools/tpm help update | grep thl-log-retention
--thl-log-retention How long do you want to keep THL files?
```

If you make changes in this way then you must be sure to run tpm fetch from your staging directory prior to any further changes. Skipping this step may result in you pushing an old configuration from the staging directory.

Making Configuration Changes to all hosts

This process must be run from the staging directory in order to run properly. Determine where the current software was installed from.

```
shell> tpm query staging
tungsten@staging-host:/opt/continuent/software/tungsten-clustering-7.0.3-141
```

This outputs the hostname and directory where the software was installed from. Make your way to that host and directory before proceeding.

```
shell> ./tools/tpm fetch --reset --directory=/opt/continuent \
    --hosts=host1,autodetect
```

This will load the configuration into the local staging directory. Review the current configuration before making any configuration changes or deploying the new software.

```
shell> ./tools/tpm reverse
```

This will output the current configuration of all services defined in the staging directory. You can then make changes using tpm configure before pushing out the upgrade. Run tpm reverse again before tpm update to confirm your changes were loaded correctly.

```
shell> ./tools/tpm configure service_name ...
shell> ./tools/tpm update
```

This will update the configuration file and then push the updates to all hosts. No additional arguments are needed for the tpm update command since the configuration has already been loaded.

Note

The tpm update command may cause a brief outage while restarting the connectors. This will occur if you are upgrading to a new version You can avoid that with:

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

The connectors must be updated separately on each server by running:

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

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 10.1. tpm Core Options

Option	Description
--force [422], -f [422]	Do not display confirmation prompts or stop the configure process for errors
--help [422], -h [422]	Displays help message
--info [422], -i [422]	Display info, notice, warning and error messages
--notice [422], -n [422]	Display notice, warning and error messages
--preview [422], -p [422]	Displays the help message and preview the effect of the command line options
--profile [422]	Sets name of config file
--quiet [422], -q [422]	Only display warning and error messages

Option	Description
--verbose [422], -v [422]	Display debug, info, notice, warning and error messages

– --force [422]

Forces the deployment process to complete even if there are warning or error messages that would normally cause the process to fail. Forcing the installation also ignores all confirmation prompts during installation and always attempts to complete the process.

– --help [422]

Displays the help message for tpm showing the current options, commands and version information.

– --info [422]

Changes the reporting level to include information, notice, warning and error messages. Information level messages include annotations of the current process and stage in the deployment, such as configuration or generating files and configurations. This shows slightly more information than the default, but less than the full debug level offered by --verbose [422].

– --notice [422]

Sets the output level to include notice, warning, and error messages. Notice level messages include information about further steps or actions that should be taken, or things that should be noted without indicating a failure or error with the configuration options select.

– --preview [422]
Displays the help message and preview the effect of the command line options.

– --profile [422]

Specify the name of the configuration file to be used, e.g. tpm profile /tmp/config.sh. This can be useful if you are performing multiple configurations or deployments from the same staging directory. The entire configuration and deployment information is stored in the file before installation is started. By specifying a different file you can have multiple deployments and configurations without requiring separate staging directories.

– --quiet [422]

Changes the error reporting level so that only warning and error messages are displayed. This mode can be useful in automated deployments as it provides output only when a warning or error exists. All other messages, including informational ones, are suppressed.

– --verbose [422]

Displays a much more detailed output of the status and progress of the deployment. In verbose mode, tpm annotates the entire process describing both what it is doing and all debug, warning and other messages in the output.

The tpm utility handles all operations from installation to updates and upgrades, and also provides a number of useful post-installation commands that can be used to simplify common administration operations.

The installation process starts in a staging directory. This is different from the installation directory where Tungsten Cluster will ultimately be placed but may be a sub-directory. In most cases we will install to /opt/continuent but use /opt/continuent/software as a staging directory. The release package should be unpacked in the staging directory before proceeding. See the Section B.2, “Staging Host Configuration” for instructions on selecting a staging directory.

Table 10.2. tpm Commands

Option	Description
ask	Ask tpm to provide values from the common configuration
configure	Configure a data service within the global configuration
connector	Open a connection to the configured connector using mysql
copy	Automates the act of copying the shared SSL keys generated during installation to other cluster nodes as part of the post-installation workflow.
delete-service	Delete a replication service or a composite datasource
diag	Obtain diagnostic information
fetch	Fetch configuration information from a running service

Option	Description
find-seqno	Assists with locating event information in the THL and producing a dsctl set command as output.
firewall	Display firewall information for the configured services
generate-haproxy-for-api	Generates a correctly formatted haproxy configuration block for use with Tungsten Dashboard v1
help	Show command help information
install	Install a data service based on the existing and runtime parameters
mysql	Open a connection to the configured MySQL server
policy	Display and optionally set the cluster policy.
post-process	Reset the THL for a host
promote-connector	Restart the connectors in the active configuration
purge-thl	Reset the THL for a host
query	Query the active configuration for information
report	Generate a security report for all available communication channels on a per-node basis
reverse	Show you the commands required to rebuild the configuration for the current directory.
uninstall	Uninstall software from host(s)
update	Update an existing configuration or software version
validate	Validate the current configuration
validate-update	Validate the current configuration and update

10.5.1. tpm ask Command

tpm ask can be used to query values from the common configuration

Usage:

```
tpm ask [args] [context] {value_or_function_name} [argument]
```

Table 10.3. tpm ask Arguments

Option	Description
--allstages	Display stages of the replicators for all roles
--api	Use the v2 API REST interface instead of the command line when possible
--debug, -d	Debug Mode.
--dsrole	Display local datasource role
--dsstate	Display local datasource state
--help, -h	
--info, -i	
--nodeinfo	Display state and role of local datasource and replicator
--path, -p	Pass full path to replicator executables e.g. /opt/replicator/tungsten/tungsten-replicator/bin
--quiet, -q	
--stages	Display stages of the local replicator
--test, -t	
--thl	Path and name of thl executable (Ignores --path if also supplied)
--trepctl	Path and name of trepctl executable (Ignores --path if also supplied)
--trrole	Display local replicator role

Option	Description
`--trstate`	Display local replicator state
`-v`	Verbose output.

`[context]` is optional and can be one of the following:

Table 10.4. tpm ask Context Options

Option	Description
`all`	Extract every available variable from the perl common object.
`common`	Extract the variable from the perl common object. This is the default if no context is supplied.
`function`	Extract the return value of the named function from within the perl_common object. You may specify an optional argument to the function.
`functions`	Extract the available function names from the perl common object.
`keys`	Extract all available variable names from the perl common object.

Examples:

```
shell> tpm ask keys
shell> tpm ask all
shell> tpm ask functions
shell> tpm ask services
shell> tpm ask isCAA
shell> tpm ask function managerEnabled
shell> tpm ask function cleanBool false
```

10.5.2. tpm check Command

The tpm check command provides simple ways to check an installation,

10.5.2.1. tpm check ini Command

10.5.2.2. tpm check ports Command

10.5.3. tpm configure Command

The tpm configure command is used to build a configuration when deploying via the staging method.

> **Important**
>
> For backwards compatability, the Staging deployment method is still supported however it will be deprecated in a future release and therefore it is not recommended for new deployments. Existing deployments should be converted when possible. See Section 10.3, “Converting from Staging to INI” for details on converting.

10.5.4. tpm connector Command

This will open a MySQL CLI connection to the local Tungsten Connector using the current values for `application-user` [474], `application-password` [473] and `application-port` [473].

```
shell> tpm connector
```

> **Warning**
>
> This command will fail if the mysql utility is not available or if the local server does not have a running Tungsten Connector.

> **Important**
>
> The MySQL 5.7/8.0 command-line client will now attempt to connect via SSL by default, which will fail on the Connector unless it is configured for SSL operations. You may add the `skip-ssl` option to bypass this issue. See Section 5.13.3, “Configuring Connector SSL” for more information about using SSL with the Connector.

10.5.4.1. tpm connector --hosts

Limits the connection to the list of specified hosts. For example:

```
shell> tpm connector --hosts host1,host2
```

Would limit the connection to the connector on one of the specified hosts. The hostname must be specified in the same form as it is in the configuration.

10.5.4.2. tpm connector --dataservice-name

Limit the command to the hosts in the specified dataservice. Mutliple dataservices can be specified by providing each dataservice separated by a comman.

```
shell> tpm connector --dataservice-name east
```

10.5.4.3. tpm connector --samples

Provides sample configuration information for various common development environments:

```
shell> tpm connector --samples
Bash -
mysql -hdb1 -P3306 -uapp_user -p'PASSWORD_OBFUSCATED'

Perl::dbi -
$dbh = DBI->connecti('DBI:mysql:host=db1;port=3306', 'app_user', 'PASSWORD_OBFUSCATED')

PHP::mysqli -
$dbh = new mysqli('db1', 'app_user', 'PASSWORD_OBFUSCATED', 'schema', '3306');

PHP::pdo -
$dbh = new PDO('mysql:host=db1;port=3306', 'app_user', 'PASSWORD_OBFUSCATED');

Python::mysql.connector -
dbh = mysql.connector.connect(user='app_user', password='PASSWORD_OBFUSCATED', host='db1', port=3306, database='schema')

Java::DriverManager -
dbh=DriverManager.getConnection("jdbc:mysql://db1:3306/schema", "app_user", 'PASSWORD_OBFUSCATED')

tpm connector -
/usr/bin/mysql --user=app_user --password='PASSWORD_OBFUSCATED' --host=db1 --port=3306
```

10.5.5. tpm copy Command

Automates the act of copying the shared SSL keys generated during installation to other cluster nodes as part of the post-installation work-flow.

This command should only be run on ONE node, and should only be needed once, right after initial installation has completed on all nodes, and before the Tungsten processes have been started.

> **Note**
>
> This tool should only be needed for INI-based installations, and requires SSH access between the nodes to function properly.

See the table below for a list of valid arguments:

Table 10.5. tpm copy Common Options

Option	Description
`--debug`, `-d`	Debug Mode.
`--help`, `-h`	
`--hosts`	Specify a comma-separated list of node names as seen in tpm reverse [default: all nodes in clusters as shown below]
`--i-am-sure`	Confirm you want the files copied and that the installs are done on all nodes
`--info`, `-i`	

Option	Description
--list	List the default nodes shown below (all valid nodes in all clusters)
--quiet, -q	
--timeout	Specify the ssh/scp Timeout as an integer in seconds (default: 3)
-v	Verbose output.

With no options, prompt to continue, then, by default, copy the needed security files from /opt/continuent/share/ to all nodes on all clusters, skipping the localhost. To bypass the interactive pause, you may specify i-am-sure to confirm that all nodes are done installing, and that none of the Tungsten processes have been started yet.

This tool uses OS commands ssh, scp and uptime to function.

The scp and ssh commands will be called with two options:

- -o StrictHostKeyChecking=no
- -o ConnectTimeout={timeout in seconds, default 3}

The files specified by:

- /opt/continuent/share/[jpt]*
- /opt/continuent/share/.[jpt]*

will be copied from the local host to the same directory on all hosts in the target list, excluding the local node.

The default security key location path (/opt/continuent/share/) may be changed (although NOT recommended) by using the tpm option security-directory [513]

10.5.6. tpm delete-service Command

The tpm delete-service command allows you to cleanly remove a dataservice from your cluster, or a single replication service from a stand-alone replicator installation.

The tpm delete-service command will know if it is being run from the Clustering software or the Replicator software, and will act accordingly

See the table below for a list of valid arguments:

Table 10.6. tpm delete-service Common Options

Option	Description
--api	Use the v2 API REST interface instead of the command line when possible
--auto, -A	Automatically execute any needed commands that it is possible to handle.
--debug, -d	Debug Mode.
--i-have-run-tpm-update	For Staging-method installations, pass this flag to confirm that the `tools/tpm update --replace-release `command has already been run from the staging directory
--help, -h	
--i-am-sure	Bypass the 'Are You Sure?' prompt when using --auto.
--info, -i	
--newini, -n	Pass the fullpath (and filename) to an INI file to be used at the 'Edit INI' step
-f	Pass the force flag to tpm.
--path, -p	Pass full path to replicator executables e.g. --path /opt/replicator/tungsten/tungsten-replicator/bin
--quiet, -q	
--test, -t	

Option	Description
`--thl`	Path and name of thl executable (Ignores --path if also supplied)
`--trepctl`	Path and name of trepctl executable (Ignores --path if also supplied)
`-v`	Verbose output.

A number of options determine the behavior of this command and these are outlined below.

Usage for tpm delete-service

```
shell> tpm delete-service [args] {service_name} [configuration_service_name]
```

`{service_name}` is the Replicator service name as seen in trepctl services and/or the Manager composite datasource name as seen via cctrl.

`[configuration_service_name]` is the Replicator service name as seen in tpm reverse, and only needed for standalone Replicator installations where the service name in the configuration is not the same as the actual replication service name, i.e. in cluster-extractor topologies where there is a cluster-alias employed.

The default behavior is to display the needed commands for the admin to execute manually.

To use this tool in a fully automated manner:

- specify `--auto`
- include `--i-am-sure` to bypass most interactive prompts
- include `--newini {filename}` otherwise you will be prompted to edit the INI file in the vi editor before proceeding

Use-cases

- Tungsten Replicator: i.e. for Standalone, Cluster-Extractor, Fan-In and Multi-Site/Active-Active topologies where there is a discrete Replicator running outside of a Cluster (needs Replicator workflow)
- Composite Active/Passive Clusters: remove a composite member cluster (needs Manager workflow)
- Composite Active/Active Clusters: remove a composite member cluster (needs Manager + Replicator workflow)

Workflows

1. Replicator Service

   ```
   shell> trepctl -all-services offline
   shell> rm tungsten-replicator/conf/static-SERVICE.properties
   shell> rm tungsten-replicator/conf/.static-SERVICE.properties.orig
   shell> replicator restart
   shell> trepctl -all-services online

   # Remove the various directories after the restart so that the replicator no longer has the dirs open :
   shell> rm -rf thl/SERVICE/
   shell> rm -rf relay/SERVICE/
   shell> trepctl -all-services online
   ```

2. Manager Service:

   ```
   shell> cctrl
   cctrl> set policy maintenance
   cctrl> use composite_parent
   cctrl> drop composite datasource SERVICE
   ```

   Remove *SERVICE* line(s) from `cluster-home/conf/dataservices.properties`

   - CAA & CAP: Remove lines matching `/^SERVICE=/`
   - CAA only: Remove lines matching `/^.+?_from_SERVICE=/`
   - If CAA, also follow Replicator Service workflow for `*_from_SERVICE`

   Finish by running an update and returning cluster to *AUTOMATIC*

   ```
   shell> tpm update
   or
   shell> manager restart
   shell> cctrl
   ```

```
cctrl> set policy automatic
```

> **Note**
>
> tpm uninstall should also be run as a final optional step if the service being deleted is the only service on any affected hosts.

10.5.7. tpm diag Command

The tpm diag command will create a TGZ file including log files, current dataservice status and a number of OS metrics.

```
shell> tpm diag
db1|Diagnostic information written to /home/tungsten/tungsten-diag-2025-03-04-14-25-56.tgz
```

Table 10.7. tpm diag Common Options

Option	Description
--allhosts, -a	Gather diags for all hosts listed in the configuration
--api	Use the v2 API REST interface instead of the command line when possible
-p, --pass, --password	Specify API auth password (default: not defined)
-us, --user	Specify API auth user (default: not defined)
--debug, -d	Displays debug-level status messages
--force, -f	Force hosts not listed in the configuration to be used, used with --hosts
--groups	Specify a comma-separated list of subroutine groups to include (NO spaces). Anything not listed will be skipped.
--help, -h	Displays a help message
--hosts	Specify a comma-separated list of hosts to act upon (NO spaces)
--include	Specify a comma-separated list of subroutines to include (NO spaces). Anything not listed will be skipped.
--info, -i	Displays info-level status messages
--list	Print all diagnostic gathering groups and associated subroutines for use with --include, --groups, --skip and --skipgroups
--local, -l	Limit to this node only when deployed using the Staging method, because by default all hosts will be gathered.
-L, --log-limit	Specify the quantity of log files to gather (default: unlimited).
--path, -p	Pass full path to replicator executables e.g. --path /opt/replicator/tungsten/tungsten-replicator/bin
--quiet, -q	Hides status output whenever possible
--skip	Specify a comma-separated list of subroutines to skip (NO spaces)
--skipgroups	Specify a comma-separated list of subroutine groups to skip (NO spaces)
--thl	Path and name of thl executable (Ignores --path if also supplied)
--trepctl	Path and name of trepctl executable (Ignores --path if also supplied)
-v	Displays verbose-level status messages.

The operation of tpm diag differs between installation types (Staging vs INI). This is outlined below:

The tpm diag command can be issued in two ways:

- For an INI based installation, the tpm diag command alone will ONLY obtain diagnostics from the local host on which the command is executed.

  For an Staging based installation, the tpm diag command alone will obtain diagnostics all hosts configured within the topology.

- Use of the --hosts option will obtain diagnostics from the specified host(s) only, e.g:

```
shell> tpm diag --hosts db1,db2
```

- The `-a` (or `--allhosts`) option will attempt to obtain diagnostics from all hosts in the cluster if ssh has been configured and the other hosts can be reached. The output of tpm diag will provide feedback detailing the hosts that were reached.

The structure of the created file will depend on the configured hosts, but will include all the logs for each accessible host configured in individual directories for each host.

Using additional options

It is possible to limit the amount of information gathered by tpm diag by optionally skipping individual gather subroutines, or skipping entire groups by using `--include` (v7+ only), `--skip`, `--group` (v7+ only), `--skipgroups` or `--skipsudo` (v7.1+ only)

To see a list of the various groups and subroutines for use with these options, execute with the `--list` option, for example:

Examples

```
shell> tpm diag --list

GROUP: SUBROUTINE(s)
cluster: cctrlClusterValidate cctrlClusterTopologyValidate cctrlHistoryFile cctrlLong cctrlPing cctrlStatus
general: confDirs logFiles miscFiles
mysql: etcMysql etcMycnf etcMyInclude etcMyIncludedirs getMysqlCommands mysqlErrorLog
os: getOSCommands etcHosts cpuInfo etcSystemRelease etcOSRelease usrlibOSRelease
replicator: thlInfo thlIndex trepctlPerf trepctlQuick trepctlStatus trepctlStatusJSON
tpm: etcTungsten tpmDiff tpmReverse tpmValidate tpmAskSummary
```

```
shell> tpm diag --skip=getOSCommands
```

```
shell> tpm diag --skipgroups=os
```

tungsten_send_diag

If the host you are running the diag from has external internet connectivity, you may also wish to consider using tungsten_send_diag. This will run tpm diag for you and automatically upload the resulting file to Continuent Support. For more information on using this, see Section 9.38, “The tungsten_send_diag Script”

10.5.8. 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` [494]

  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` [521]

  The username to be used when logging in to other hosts.

- `--directory` [489]

  The installation directory of the current Tungsten Cluster installation. If `autodetect` is specified, then tpm will look for the installation directory by checking any running Tungsten Cluster processes.

10.5.9. tpm firewall Command

The tpm firewall command displays port information required to configure a firewall. When used, the information shown is for the current host:

```
shell> tpm firewall
To db1
-----------------------------------------------------------------------------
From application servers      3306, 3307
From connector servers        8096, 11999, 12000, 13306
```

```
From database servers         2112, 7800, 8090, 9997, 10999, 11999, 12000, 13306
```

The information shows which ports, on which hosts, should be opened to enable communication.

10.5.10. tpm find-seqno Command

For convenience, a wrapper script tungsten_find_seqno is also available. In all the examples below, simply replace tpm find-seqno with tungsten_find_seqno or vice versa.

The tpm find-seqno assists with locating event information in the THL and producing a dsctl set command as output.

The tpm find-seqno command performs the following steps:

- Get the Replicator sequence number to search for from the CLI and validate
- Validate paths and commands
- Load all available service names and validate any specified service against that list
- Check if this Replicator is in the Primary role
- Locate the supplied seqno in the available THL
- Parse the THL found, if any
- Generate the dsctl command and display

Table 10.8. tungsten_find_seqno Options

Option	Description
`--debug`, `-d`	Enable additional debug output.
`--help`, `-h`	Show help text
`--path`, `-p`	Specify the full path to the executable directory where the thl and trepctl commands are located.
`--quiet`, `-q`	Prevent final message from being output.
`--service`, `-s`	Specify the service name to use with the various commands.
`--test`, `-t`	Test mode - do not perform the actual update on the files
`--thl`	Specify the full path to the thl command executable file.
`--trepctl`	Specify the full path to the trepctl command executable file.
`--verbose`, `-v`	Show verbose output

Below is a sample session:

```
shell> tpm find-seqno 4
dsctl set -reset -seqno 4 -epoch 2 -event-id "mysql-bin.000030:0000000000001981;-1" -source-id "db1"
```

10.5.11. tpm generate-haproxy-for-api Command

This read-only action will read all available INI files and dump out corresponding `haproxy.cfg` entries with properly incrementing ports; the composite parent will come first, followed by the composite children in alphabetical order.

See the table below for a list of valid arguments:

Table 10.9. tpm generate-haproxy-for-api Common Options

Option	Description
`--advanced`, `-A`	
`-c`	Use Connector hosts in the backend definitions.
`--debug`, `-d`	Debug Mode. Very Chatty, Verbose mode also implied.
`-f`	Include additional backend flags to backend host lines.
`--help`, `-h`	

Option	Description
`--port`	Provide the starting port number
`--quiet`, `-q`	
`--test`, `-t`	
`--verbose`, `-v`	Verbose output.
`--yes`, `-y`	

10.5.12. 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-clustering-7.0.3-141_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
...
```

10.5.13. tpm install Command

The tpm install command performs an installation based on the configuration held within the INI Configuration file.

The `/etc/tungsten/tungsten.ini` should first have been created and populated with the configuration appropriate to your needs.

After unpacking the software into `/opt/continuent/software` you would then run the tpm install command, for example:

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> ./tools/tpm install
```

or, if you use a different name/location for the ini file, you can specify the `--ini` option:

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> ./tools/tpm install --ini /my/directory/myconfig.ini
```

For staging deployments, the tpm install requires additional options to configure the deployment, 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
```

Or, alternatively use tpm configure prior to tpm install:

```
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
```

During installation, tpm checks for any host configuration problems and issues 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.

10.5.14. tpm mysql Command

This will open a MySQL CLI connection to the local MySQL server using the current values for `replication-user` [510], `replication-password` [510] and `replication-port` [510].

```
shell> tpm mysql
```

This command will fail if the mysql utility is not available or if the local server does not have a running database server.

10.5.15. tpm policy Command

The tpm policy command displays and optionally sets the cluster policy.

Table 10.10. tpm policy Options

Option	Description
`--all`	Show the policy for all cluster services.
`--automatic`, `--auto`, `-a`	Set the cluster policy to AUTOMATIC mode.
`--maintenance`, `--maint`, `-m`	Set the cluster policy to MAINTENANCE mode.
`--print`, `--display`, `--get`, `--list`, `--show`, `-d`, `-g`, `-l`, `-p`, `-s`	Displays the current cluster policy. Issuing with no arguments will also display the current policy.

```
shell> tpm policy
maintenance
shell> tpm policy -a
automatic
```

The tpm policy command was designed as an alternative for the cctrl set policy command to make it easier to set the cluster policy on the command line.

10.5.16. tpm post-process Command

For convenience, a wrapper script tungsten_post_process is also available. In all the examples below, simply replace tpm post-process with tungsten_post_process or vice versa.

> **Note**
>
> The tpm post-process command is automatically executed during install or update of Composite Active/Active and Composite Dynamic Active/Active topologies. Manual execution should only be necessary if changes wish to be applied without the nedd for running tpm update

The tpm post-process command assists with the graceful maintenance of the static cross-site replicator configuration files on disk.

Table 10.11. tpm-post-process Options

Option	Description
`--debug`, `-d`	Enable additional debug output.
`--help`, `-h`	Show help text
`--quiet`, `-q`	Prevent final message from being output.
`--test`, `-t`	Test mode - do not perform the actual update on the files
`--verbose`, `-v`	Show verbose output
`--yes`, `-y`	Bypass interactive confirmation prompt

The tpm post-process command performs the following steps:

- Locates, reads and parses the various INI configuration files.

  Below is the list of possible INI files and file match patterns:

  - `{$HOME}/tungsten.ini`
  - `/etc/tungsten/tungsten*.ini`

    Note

    Please note that all files in the `/etc/tungsten` directory starting with `tungsten` and ending in `.ini` will be used.

  - `/etc/tungsten.ini`
- Identifies cross-site services that are local to the node, as well as the main local service.

  Warning

  Only Replicator options and properties for specific entry types are currently supported.

  The supported stanzas are as follows:

  - `{service}.replicator`
  - `{service}_from_{service}`

  Below is an example INI file showing section identifiers only:

```
[defaults]
[defaults.replicator]
[east]
[east.replicator]
[west]
[east_from_west]
[west_from_east]
```

  Of the above example section identifiers, only the following would be used by tpm post-process:

  - `east.replicator`
  - `east_from_west`
  - `west_from_east`

  The remaining example section identifiers (`defaults`, `defaults.replicator`, `east` and `west`) would be ignored.
- Gathers configuration options and properties defined for each identified local service.
- Locates the associated configuration file on disk and gets the existing value for the option key.
- Interactively prompts for confirmation. This may be bypassed using the `-y` option to tpm post-process.
- Filter out any files that are not static replicator configurations.
- The tpm post-process command will then do one edit-in-place per ini option per local service.
- The script will check to make sure the value has been updated.
- Finally, there will be a summary message and helpful instructions about next steps. This may be bypassed using the `-q` option to tpm post-process.

Below is a sample session:

```
shell> tpm post-process
About to update the following config file(s):

/opt/continuent/tungsten/tungsten-replicator/conf/static-east_from_north.properties
/opt/continuent/tungsten/tungsten-replicator/conf/static-east_from_west.properties

Do you wish to continue? [y/N] y

Done!
```

```
tpm post-process updated 0 configuration options successfully and skipped 6

Pick one node and set Maintenance mode so the manager does not try to bring the replicator online when you don't want it to:

  echo "set policy maintenance" | cctrl

Then take the Replicator offline gracefully on one node:

  trepctl -all-services offline

Restart the Replicator:

  replicator restart

Bring all services online:

  trepctl -all-services online

Check the status:

  trepctl services

Repeat as needed.

When all nodes are done, pick one node and execute:

  echo "set policy automatic" | cctrl
```

10.5.17. tpm promote-connector Command

The tpm promote-connector command should be used after performing a tpm update with the `--no-connectors` [504] option.

When using the `--no-connectors` [504] option with, running connectors are not stopped and restarted with the latest configuration or application updates, which would otherwise interrupt active applications using the connector.

The tpm promote-connector stops and restarts the configured Connector services on all configured hosts using the currently active configuration, thus allowing you to elect to perform the restart at a convenient time to cause minimal disruption:

```
shell> tpm promote-connector

NOTE  >> Command successfully completed
```

The tpm command will use connector graceful-stop 30 followed by connector start [334] when upgrading versions. If that command fails then a regular connector stop [335] is run. This behavior is also applied when using tools/tpm update --replace-release.

10.5.18. tpm purge-thl Command

The tpm purge-thl command is designed to assist with identifying the safe removal of THL based on the following rules:

For convenience, a wrapper script tungsten_purge_thl is also available. In all the examples below, simply replace tpm purge-thl with tungsten_purge_thl or vice versa.

Table 10.12. tpm purge-thl Options

Option	Description
`--auto`, `-A`	Automatically execute any needed commands that it is possible to handle. Edits to the configuration are not possible at this time.
`--debug`, `-d`	Enable additional debug output.
`--help`, `-h`	Show help text
`--i-am-sure`	Bypass the "Are You Sure?" prompt when using --auto.
`--info`, `-i`	
`--path`, `-p`	Use to supply full path to replicator executables used for thl and trepctl if not in default location.
`--quiet`, `-q`	
`--test`, `-t`	
`--thl`	Use to supply full path to replicator executables used for thl if not in default location. Ignores --path if supplied
`-d`, `--debug`	Use to supply full path to replicator executables used for trepctl if not in default location. Ignores --path if supplied

Option	Description
`--verbose`, `-v`	

- Gather the last applied seqno from all Replica nodes and take the lowest one
- Find the current THL file which contains that seqno, then locate the previous one
- construct a thl purge command to remove thl thru the last seqno in the prev file

A replicator service name name may be optionally specified as the last argument, for example:

```
shell> tpm purge-thl -A alpha
```

The default behavior is to display the needed commands for the admin to execute manually.

10.5.19. tpm query Command

The tpm 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 modified-files — return the list of files modified since installation by tpm
- tpm query staging — return the staging directory from where Tungsten Cluster was installed
- tpm query topology — return the current topology
- tpm query usermap — return the list of users organized by type from the `user.map`
- tpm query values — return the list of configured values
- tpm query version — get the version of the current installation

10.5.19.1. tpm query config

Returns a list of all of the configuration values, both user-specified and implied within the current configuration. The information is returned in the form of a JSON value:

```
shell> tpm query config
{
"__system_defaults_will_be_overwritten__": {
...
"staging_directory": "/home/tungsten/tungsten-clustering-7.0.3-141",
"staging_host": "tr-ms1",
"staging_user": "tungsten"
}
```

10.5.19.2. tpm query dataservices

Returns the list of configured dataservices that have, or will be, installed:

```
shell> tpm query dataservices
alpha                          : PHYSICAL
```

10.5.19.3. 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-clustering-7.0.3-141_pid22729",
"dataservice_host_options": {
```

```
  "alpha": {
    "start": "true"
  }
...
"staging_directory": "/home/tungsten/tungsten-clustering-7.0.3-141",
"staging_host": "tr-ms1",
"staging_user": "tungsten"
}
```

10.5.19.4. 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.

10.5.19.5. tpm query staging

Returns the host and directory from which the current installation was created:

```
shell> tpm query staging
tungsten@host1:/home/tungsten/tungsten-clustering-7.0.3-141
```

This can be useful when the installation host and directory from which the original configuration was made need to be updated or modified.

For convenience, the following command will take you straight to the directory:

```
shell> cd `tpm query staging | cut -d: -f2`
```

10.5.19.6. tpm query topology

Returns the current topology and list of configured servers and roles in the form of a JSON object:

```
shell> tpm query topology
{
"host1": "slave",
"host2": "slave",
"host3": "master"
}
```

10.5.19.7. tpm query usermap

Returns a summarized list of the currently configured users in the `user.map`:

```
shell> tpm query usermap
# user.map Summary

# Configured users
app_user ******** alpha

# Script entries


# DirectRead users


# Host-based routing entries
```

10.5.19.8. tpm query version

Returns the version of the installed software:

```
shell>  tpm query version
7.0.3-141
```

10.5.20. tpm report Command

The purpose of tpm report is to provide easy access to all of the settings that pertain to a specific topic.

Usage:

```
tpm report [args]
```

Arguments:

Table 10.13. tpm report Common Options

Option	Description
--api	Use the v2 API REST interface instead of the command line when possible
--debug, -d	
--extra, -x	Provide additional details in the reports
--help, -h	
--info, -i	
--json	Display report as JSON, all other output will be suppressed
--list, -l	List reports by number
--password, -p	Use to specify the API auth password (default: not defined)
--path	Use to supply full path to replicator executables
--ports	When available, display the hostname and listener ports
--quiet, -q	
--report, --filter, -r	Limit display to the specified report number(s); Use a comma-separated numeric list with no spaces to specify multiple reports.
--ssl, -security	Display current security settings and values (default behavior when no topic is specified)
--test, -t	
--thl	Use to supply full path to thl executable (Ignores --path)
--trepctl	Use to supply full path of trepctl executable (Ignore --path)
--user, -u	Use to specify API auth User (default: not defined)
--verbose, -v	

The default (and only) topic is the security stance. More topics will be added over time.

Each topic contains a set of numbered reports. View the list of reports for any topic using --list (or -l). For example:

```
shell> tpm report --list
>>> Security Reports <<<

1. Application to Connector (mysql)
Communications from the client application to the Connector port

2. Connector to Database (mysql)
Communications from the Connector to the Database port

3. Connector to Manager (proprietary (routerGateway))
Communications from the Connector to the Manager

4. Manager to Manager (rmi/jmx, jgroups)
Communications from Manager to Manager

5. Manager to Database (mysql)
Communications from Manager to Database

6. Manager to Replicator (rmi/jmx)
Communications from Manager to Replicator

7. Replicator to Replicator (thl)
Communications from Replicator to Replicator

8. Replicator to Database (mysql)
Communications from Replicator to Database

9. connector Command to the Connector (rmi/jmx)
Communications from the connector cli command to the to local Connector process

10. cctrl Command to the Manager (rmi/jmx)
Communications from the cctrl cli command to the to Manager process

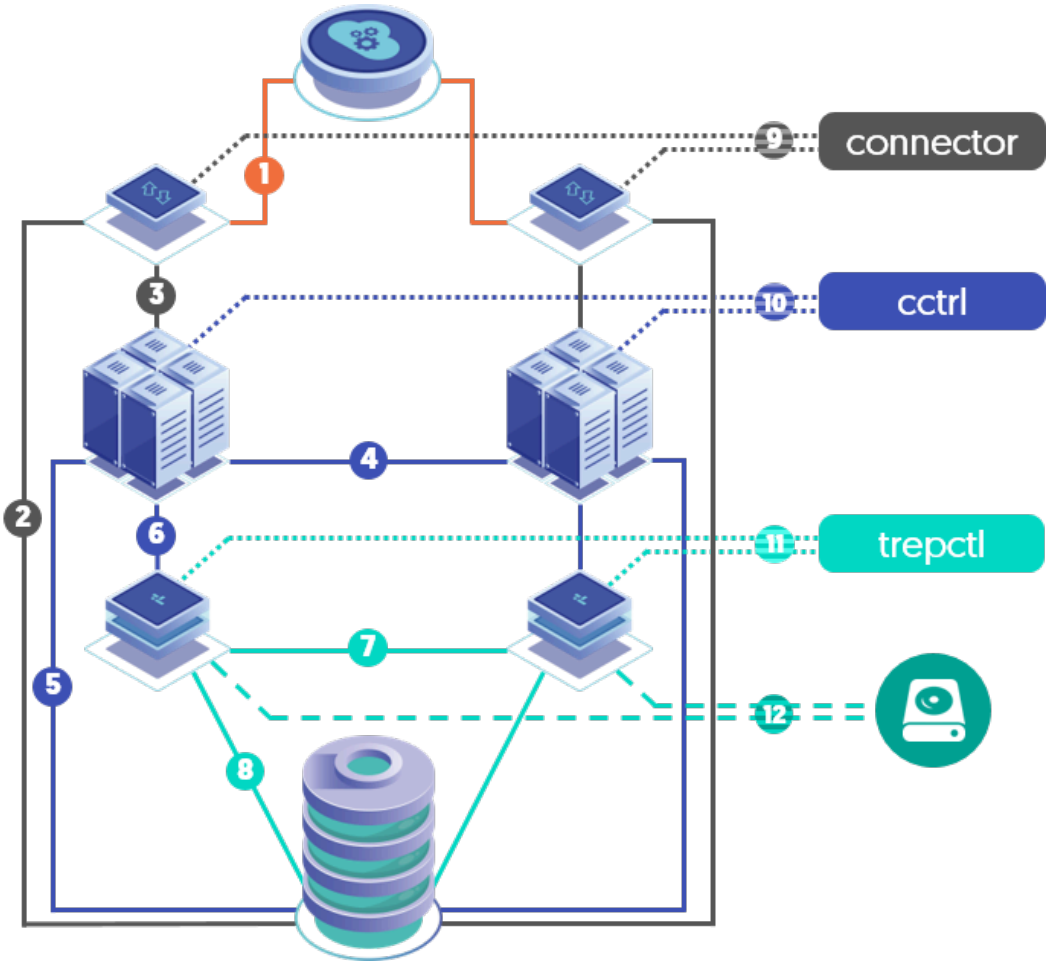
11. trepctl Command to the Replicator (rmi/jmx)
Communications from the trepctl cli command to the to Replicator process

12. Replicator On Disk THL (proprietary (thl encryption))
THL files on disk are encrypted
```

```
13. SSL-Specific `tpm` Options for the API (http,https)
Options used by tpm to enable or disable the API for each component
```

The following graphic provides a visual representation to the various communication channels within the cluster and how they are mapped to the various "levels" presented above.

Figure 10.1. Internals: Cluster Communication Channels



To display just a specific report, specify it using the `--report` option. For example:

```
shell> tpm report --report 1

TUNGSTEN SECURITY REPORT as of Thu May  5 20:37:17 UTC 2022

--------------------------------------------------------------------------------
>> Comm Channel 1:  Application to Connector <<
--------------------------------------------------------------------------------

=> CHANNEL INFORMATION:
Connector Bridge Mode: OFF
Application-to-Connector SSL is Enabled

=> TPM OPTIONS:
connector-ssl-capable = true
enable-connector-client-ssl = true
enable-connector-ssl = true
```

To display additional information in the report, use the `--extra` (or `-x`) option. For example:

```
shell> tpm report --extra --report 1

TUNGSTEN SECURITY REPORT as of Thu May  5 20:37:17 UTC 2022

--------------------------------------------------------------------------------
```

```
>> Comm Channel 1:  Application to Connector <<
--------------------------------------------------------------------------------

=> CHANNEL INFORMATION:
Connector Bridge Mode: OFF
Application-to-Connector SSL is Enabled (SSL.IN=true, Proxy Mode)

=> TPM OPTIONS:
connector-ssl-capable = true
enable-connector-client-ssl = true
enable-connector-ssl = true
```

To attempt to gather the information via the API, use `--api` on the command line. The tpm report command will fall back to the CLI tools if an API method is unavailable for a given report. For example:

```
shell> tpm report -x --report 1 --api

TUNGSTEN SECURITY REPORT as of Thu May  5 20:43:28 UTC 2022

--------------------------------------------------------------------------------
>> Comm Channel 1:  Application to Connector <<
--------------------------------------------------------------------------------

=> CHANNEL INFORMATION:
APIv2 Connector is SSL Capable: true
APIv2 Connector Requires SSL: true
APIv2 Connector Bridge Mode: false
WARN: No APIv2 solution available for Channel 1: Application to Connector - falling back to CLI
Application-to-Connector SSL is Enabled (SSL.IN=true, Proxy Mode)

=> TPM OPTIONS:
connector-ssl-capable = true
enable-connector-client-ssl = true
enable-connector-ssl = true
```

You may need to specify `--user` and `--password` for API authentication if not configured via your `/etc/tungsten/tungsten.ini` at installation time.

If both `--user` and `--password` are defined, tpm report will use them. If either or both `--user` and `--password` are missing, tpm report will attempt to derive the values from the configuration.

If you wish to output the report in machine-readable JSON-formatted text with other output suppressed, simply add the `--json` option. For example:

```
shell> tpm report -x --report 1 --api --json
{
 "1" : {
    "channelinformation" : [
       "APIv2 Connector is SSL Capable: true",
       "APIv2 Connector Requires SSL: true",
       "APIv2 Connector Bridge Mode: false",
       "WARN: No APIv2 solution available for Channel 1: Application to Connector - falling back to CLI",
       "Application-to-Connector SSL is Enabled (SSL.IN=true, Proxy Mode)"
    ],
    "metadata" : {
       "description" : "Communications from the client application to the Connector port",
       "protocol" : "mysql",
       "section" : "Application to Connector",
       "sslCapable" : 1
    },
    "tpmoptions" : {
       "connector-ssl-capable" : "true",
       "enable-connector-client-ssl" : "true",
       "enable-connector-ssl" : "true"
    }
 }
}
```

To simply display all reports, use:

```
shell> tpm report
shell> tpm report -x
shell> tpm report --json
shell> tpm report -x --json
```

10.5.21. 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. by default the command will return in a staging deployment format.

```
shell> tpm reverse --ini-format
```

```
# Installed from tungsten@db1:/opt/continuent/software/tungsten-clustering-7.0.3-141
# Configuration built from /etc/tungsten/tungsten.ini
# Defaults for all data services and hosts
[defaults]
application-password=secret
application-port=3306
application-readonly-port=3307
application-user=app_user
connector-rest-api-address=0.0.0.0
disable-security-controls=true
install-directory=/opt/continuent
manager-rest-api-address=0.0.0.0
'profile-script=~/.bash_profile'
replication-password=secret
replication-port=13306
replication-user=tungsten
replicator-rest-api-address=0.0.0.0
rest-api-admin-pass=secret
rest-api-admin-user=tungsten
user=tungsten
# Options for the alpha data service
[alpha]
connectors=db1,db2,db3
master=db1
members=db1,db2,db3
topology=clustered
```

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

10.5.22. tpm uninstall Command

The tpm uninstall command is used to remove the installation.

> **Warning**
>
> The uninstall command must be used with care. This is a destructive command and irreversible.

To uninstall the software, you need to issue the following command from the installed software staging directory on every host for INI installs, or from the staging host only for Staging Installs. Running the command on the staging hosts installed via the staging method, will cascade through all nodes in the topology.

```
shell> {STAGING_DIR}/tools/tpm uninstall --i-am-sure
```

10.5.23. 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 10.1, "Processing Installs and Upgrades". The behavior of tpm update is dependent on two factors.

1. Are you upgrading to a new version or applying configuration changes to the current version?
2. The installation method used during deployment.

> **Note**
>
> Check the output of tpm query staging to determine which method your current installation uses. The output for an installation from a staging directory will start with `# Installed from tungsten@staging-host:/opt/continuent/software/tungsten-clustering-7.0.3-141`.
>
> An installation based on an INI file will aslso include the line `# Configuration built from /etc/tungsten/tungsten.ini`

Upgrading to a new version

If a staging directory was used; see Section 10.4.6, "Upgrades from a Staging Directory".

If an INI file was used; see Section 10.2.3, "Upgrades with an INI File"

Applying configuration changes to the current version

If a staging directory was used; see Section 10.4.7, “Configuration Changes from a Staging Directory”.

If an INI file was used; see Section 10.2.4, “Configuration Changes with an INI file”.

> **Warning**
>
> During the update process, the cluster will be in *MAINTENANCE* mode. This is intentional to prevent unwanted failovers during the proces, however it is important to understand that should the primary faill for genuine reasons NOT associated with the upgrade, then failover will also not happen at that time.
>
> It is important to ensure clusters are returned to the *AUTOMATIC* state as soon as all Maintenance operations are complete and the cluster is stable.

Special Considerations for the Connector

The tpm command will use connector graceful-stop 30 followed by connector start [334] when upgrading versions. If that command fails then a regular connector stop [335] is run.

This behavior is also applied when using tools/tpm update --replace-release.

The tpm command will use connector reconfigure [334] when changing connector settings without a version upgrade.

The use of connector reconfigure [334] is disabled if any of the following properties are set:

```
--application-port
--application-readonly-port
--router-gateway-port
--router-jmx-port
--conn-java-mem-size
```

If connector reconfigure [334] can't be used, connector graceful-stop 30 and connector start [334] are used.

To avoid connectors being restarted as part of this process, you should also include the `--no-connectors` [504] option. You will then need to manually retsrat the connectors or use the tpm promote-connector when convenient.

10.5.24. tpm validate Command

The tpm validate command validates the current configuration before installation. The validation checks all prerequisites that apply before an installation, and assumes that the configured hosts are currently not configured for any Tungsten services, and no Tungsten services are currently running.

```
shell> {STAGING_DIR}/tools/tpm validate
.........
...
#####################################################################
# Validation failed
#####################################################################
...
```

The command can be run after creating the `/etc/tungsten/tungsten.ini` and before a tpm install to ensure that any prerequisite or configuration issues are addressed before installation occurs.

10.5.25. tpm validate-update Command

The tpm validate-update command checks whether the configured hosts are ready to be updated. By checking the prerequisites and configuration of the dataserver and hosts, the same checks as made by tpm during a tpm install operation. Since there may have been changes to the requirements or required configuration, this check can be useful before attempting an update.

Using tpm validate-update is different from tpm validate in that it checks the environment based on the updated configuration, including the status of any existing services.

```
shell> {STAGING_DIR}/tools/tpm validate-update
....
WARN  >> host1 >> The process limit is set to 7812, we suggest a value»
  of at least 8096. Add 'tungsten       -    nproc  8096' to your »
  /etc/security/limits.conf and restart Tungsten processes. (ProcessLimitCheck)

WARN  >> host2 >> The process limit is set to 7812, we suggest a value»
   of at least 8096. Add 'tungsten       -    nproc  8096' to your »
  /etc/security/limits.conf and restart Tungsten processes. (ProcessLimitCheck)
```

```
WARN  >> host3 >> The process limit is set to 7812, we suggest a value »
  of at least 8096. Add 'tungsten       -    nproc  8096' to your »
  /etc/security/limits.conf and restart Tungsten processes. (ProcessLimitCheck)
.WARN  >> host3 >> MyISAM tables exist within this instance - These »
  tables are not crash safe and may lead to data loss in a failover »
  (MySQLMyISAMCheck)


NOTE  >> Command successfully completed
```

Any problems noted should be addressed before you perform the update using tpm update.

10.6. tpm Common Options

tpm accepts these options along with those in Section 10.8, "tpm Configuration Options".

- In an INI file, without the double-dash prefix, i.e. `skip-validation-check=MySQLConnectorPermissionsCheck` [444]
- On the command-line, using a double-dash prefix, i.e. `--skip-validation-check=MySQLConnectorPermissionsCheck` [444]

Table 10.14. tpm Common Options

CmdLine Option	INI File Option	Description
`--enable-validation-check` [442]	`enable-validation-check` [442]	Enable a specific validation check, overriding any configured skipped checks
`--enable-validation-warnings` [442]	`enable-validation-warnings` [442]	Enable a specific validation warning, overriding any configured skipped warning
`--ini` [443]	`ini` [443]	Specify the location of the directory where INI files will be located, or specify a specific filename
`--net-ssh-option` [443]	`net-ssh-option` [443]	Set the Net::SSH option for remote system calls
`--property` [443], `--property=key+=value` [443], `--property=key=value` [443], `--property=key~=/match/replace/` [443]	`property` [443], `property=key+=value` [443], `property=key=value` [443], `property=key~=/match/replace/` [443]	Modify specific property values for the key in any file that the configure script touches.
`--remove-property` [443]	`remove-property` [443]	Remove the setting for a previously configured property
`--replace-release` [443]	`replace-release` [443]	Used when upgrading or making configuration changes. This option will rebuild the entire release and recreate all metadata configuration.
`--skip-validation-check` [444]	`skip-validation-check` [444]	Do not run the specified validation check.
`--skip-validation-warnings` [444]	`skip-validation-warnings` [444]	Do not display warnings for the specified validation check.

– `--enable-validation-check` [442]

Option	`--enable-validation-check` [442]
Config File Options	`enable-validation-check` [442]
Description	Enable a specific validation check, overriding any configured skipped checks
Value Type	string

The `enable-validation-check` [442] will specifically enable a given validation check if the check had previously been set to be ignored in a previous invocation of the configuration through tpm. If a check fails, installation is canceled.

Setting both `skip-validation-check` [444] and `enable-validation-check` [442] is equivalent to explicitly disabling the specified check.

– `--enable-validation-warnings` [442]

Option	`--enable-validation-warnings` [442]
Config File Options	`enable-validation-warnings` [442]
Description	Enable a specific validation warning, overriding any configured skipped warning
Value Type	string

The `enable-validation-warnings` [442] will specifically enable a given validation warning check if the check had previously been set to be ignored in a previous invocation of the configuration through tpm.

Setting both skip-validation-warnings [444] and enable-validation-warnings [442] is equivalent to explicitly disabling the specified check.

– --ini [443]

Option	--ini [443]
Config File Options	ini [443]
Description	Specify the location of the directory where INI files will be located, or specify a specific filename
Value Type	string
Default	/etc/tungsten/tungsten.ini

Specifies an alternative location, or file, for the INI files from the default.

– --net-ssh-option [443]

Option	--net-ssh-option [443]
Config File Options	net-ssh-option [443]
Description	Set the Net::SSH option for remote system calls
Value Type	string

Enables you to set a specific Net::SSH option. For example:

```
shell> tpm update ... --net-ssh-option=compression=zlib
```

– --property [443]

Option	--property [443]
Aliases	--property=key+=value [443], --property=key=value [443], --property=key~=/match/replace/ [443]
Config File Options	property [443], property=key+=value [443], property=key=value [443], property=key~=/match/replace/ [443]
Description	Modify specific property values for the key in any file that the configure script touches.
Value Type	string

The property [443] option enables you to explicitly set property values in the target files. A number of different models are supported:

- key=value

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

- key+=value

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

- key~=/match/replace/

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

– --remove-property [443]

Option	--remove-property [443]
Config File Options	remove-property [443]
Description	Remove the setting for a previously configured property
Value Type	string

Remove a previous explicit property setting. For example:

```
shell> tpm configure --remove-property=replicator.filter.pkey.addPkeyToInserts
```

– --replace-release [443]

Option	--replace-release [443]

Config File Options	replace-release [443]
Description	Used when upgrading or making configuration changes. This option will rebuild the entire release and recreate all metadata configuration.
Value Type	boolean

This property can only be used with tools/tpm update and can only be executed from within the software staging tree.

This option is highly recommended when upgrading between versions as it will ensure all metadata is correctly rebuilt for the version being installed.

It is not necessary when applying config changes, however it can be useful if properties are not being applied correctly, or if underlying metadata has become corrupt.

– --skip-validation-check [444]

Option	--skip-validation-check [444]
Config File Options	skip-validation-check [444]
Description	Do not run the specified validation check.
Value Type	string

The skip-validation-check [444] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

Warning

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

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

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

The check in this case is MySQLDefaultTableTypeCheck [455], and could be ignored using skip-validation-check=MySQLDefaultTableTypeCheck [444].

Values can be passed as a comma-separated list, or single skip-validation-check [444] entries for each check to be skipped.

Setting both skip-validation-check [444] and enable-validation-check [442] is equivalent to explicitly disabling the specified check.

– --skip-validation-warnings [444]

Option	--skip-validation-warnings [444]
Config File Options	skip-validation-warnings [444]
Description	Do not display warnings for the specified validation check.
Value Type	string

The skip-validation-warnings [444] disables a given validation check.

You can identify a given check by examining the warnings generated during configuration. For example, the Linux swappiness warning:

```
WARN >> centos >> Linux swappiness is currently set to 60, on restart it will be 60, »
 consider setting this to 10 or under to avoid swapping. (SwappinessCheck)
```

The check in this case is MySQLDefaultTableTypeCheck [455], and could be ignored using skip-validation-warnings=SwappinessCheck [444].

Setting both skip-validation-warnings [444] and enable-validation-warnings [442] is equivalent to explicitly disabling the specified warning.

10.7. tpm Validation Checks

During configuration and installation, tpm runs a number of configuration, operating system, datasource, and other validation checks to ensure that the correct environment, prerequisites and other settings will produce a valid, working, configuration.

All relevant checks are executed automatically unless specifically ignored (warnings) or disabled (checks) using the corresponding skip-validation-warnings [444] or skip-validation-check [444] options.

Table 10.15. tpm Validation Checks

Option	Description
BackupDirectoryWriteableCheck [448]	Checks that the configured backup directory is writeable
BackupDumpDirectoryWriteableCheck [448]	Checks the backup temp directory is writeable
BackupScriptAvailableCheck [448]	Checks that the configured backup script exists and can be executed
ClusterDiagnosticCheck [448]	
ClusterStatusCheck [448]	
CommitDirectoryCheck [448]	
ConfigurationStorageDirectoryCheck [449]	
ConfigureValidationCheck [449]	
ConfiguredDirectoryCheck [449]	
ConflictingReplicationServiceTHLPortsCheck [449]	
ConnectorChecks [449]	Ensures that the configured connector selection is valid
ConnectorDBVersionCheck [449]	
ConnectorListenerAddressCheck [449]	
ConnectorRWROAddressesCheck [449]	Ensure the RW and RO addresses are different
ConnectorSmartScaleAllowedCheck [449]	Confirms whether SmartScale is valid within the current configured parameters
ConnectorUserCheck [449]	
ConsistentReplicationCredentialsCheck [450]	
CurrentCommandCoordinatorCheck [450]	
CurrentConnectorCheck [450]	
CurrentReleaseDirectoryIsSymlink [450]	
CurrentTopologyCheck [450]	
CurrentVersionCheck [450]	
DatasourceBootScriptCheck [450]	
DifferentMasterSlaveCheck [450]	
DirectOracleServiceSIDCheck [450]	
EncryptionCheck [450]	
EncryptionKeystoreCheck [450]	
FileValidationCheck [451]	
FirewallCheck [451]	
GlobalHostAddressesCheck [451]	
GlobalHostOracleLibrariesFoundCheck [451]	
GlobalMatchingPingMethodCheck [451]	
GlobalRestartComponentsCheck [451]	
GroupValidationCheck [451]	
HdfsValidationCheck [451]	
HostLicensesCheck [451]	
HostOracleLibrariesFoundCheck [451]	
HostReplicatorServiceRunningCheck [451]	
HostSkippedChecks [452]	
HostnameCheck [452]	
HostsFileCheck [452]	
InstallServicesCheck [452]	
InstallationScriptCheck [452]	

Option	Description
InstallerMasterSlaveCheck [452]	Checks whether a Primary host has been defined for the configured service.
InstallingOverExistingInstallation [452]	
JavaUserTimezoneCheck [452]	
JavaVersionCheck [452]	
KeystoresCheck [452]	
KeystoresToCommitCheck [452]	
ManagerActiveWitnessConversionCheck [453]	
ManagerChecks [453]	
ManagerHeapThresholdCheck [453]	
ManagerListenerAddressCheck [453]	
ManagerPingMethodCheck [453]	
ManagerWitnessAvailableCheck [453]	
ManagerWitnessNeededCheck [453]	
MatchingHomeDirectoryCheck [453]	
MissingReplicationServiceConfigurationCheck [453]	
ModifiedConfigurationFilesCheck [453]	
MySQLAllowIntensiveChecks [453]	Enables searching MySQL INFORMATION_SCHEMA for validation checks
MySQLApplierLogsCheck [454]	
MySQLApplierPortCheck [454]	
MySQLApplierServerIDCheck [454]	
MySQLAvailableCheck [454]	Checks if MySQL is installed
MySQLBinaryLogsEnabledCheck [454]	Checks that binary logging has been enabled on MySQL
MySQLBinlogDoDbCheck [454]	
MySQLClientCheck [454]	Checks whether the MySQL client command tool is available
MySQLConfigFileCheck [454]	Checks the existence of a MySQL configuration file
MySQLConnectorBridgeModePermissionsCheck [454]	
MySQLConnectorPermissionsCheck [454]	
MySQLDefaultTableTypeCheck [455]	Checks the default table type for MySQL
MySQLDumpCheck [455]	Checks that the mysqldump command version matches the installed MySQL
MySQLGeneratedColumnCheck [455]	Checks whether MySQL virtual/generated columns are defined
MySQLInnoDBEnabledCheck [455]	
MySQLJsonDataTypeCheck [455]	
MySQLLoadDataInfilePermissionsCheck [455]	
MySQLLoginCheck [455]	Checks whether Tungsten Cluster can connect to MySQL using the configured credentials
MySQLMyISAMCheck [455]	Checks for the existence of MyISAM tables
MySQLNoMySQLReplicationCheck [456]	
MySQLPasswordSettingCheck [456]	
MySQLPermissionsCheck [456]	
MySQLReadableLogsCheck [456]	
MySQLSettingsCheck [456]	
MySQLSuperReadOnlyCheck [456]	Checks whether super_read_only has been enabled on MySQL
MySQLTriggerCheck [456]	

Option	Description
MySQLUnsupportedDataTypesCheck [456]	
MysqlConnectorCheck [456]	
MysqldumpAvailableCheck [457]	
MysqldumpSettingsCheck [457]	
NewDirectoryRequiredCheck [457]	
NtpdRunningCheck [457]	
OSCheck [457]	
OldServicesRunningCheck [457]	
OpenFilesLimitCheck [457]	
OpensslLibraryCheck [457]	
OracleLoginCheck [457]	
OraclePermissionsCheck [457]	
OracleRedoReaderMinerDirectoryCheck [457]	
OracleServiceSIDCheck [458]	
OracleVersionCheck [458]	
PGAvailableCheck [458]	
ParallelReplicationCheck [458]	
ParallelReplicationCountCheck [458]	
PgControlAvailableCheck [458]	
PgStandbyAvailableCheck [458]	
PgdumpAvailableCheck [458]	
PgdumpallAvailableCheck [458]	
PingSyntaxCheck [458]	
PortAvailabilityCheck [458]	
ProfileScriptCheck [459]	
RMIListenerAddressCheck [459]	
RelayDirectoryWriteableCheck [459]	Checks that the relay log directory can be written to
ReplicatorChecks [459]	
RestartComponentsCheck [459]	
RouterAffinityCheck [459]	
RouterBridgeModeDefaultCheck [459]	
RouterDelayBeforeOfflineCheck [459]	
RouterKeepAliveTimeoutCheck [459]	
RowBasedBinaryLoggingCheck [459]	Checks that Row-based binary logging has been enabled for heterogeneous deployments
RsyncAvailableCheck [460]	
RubyVersionCheck [460]	
SSHLoginCheck [460]	Checks connectivity to other hosts over SSH
ServiceTransferredLogStorageCheck [460]	
StartingStoppedServices [460]	
SudoCheck [460]	
SwappinessCheck [460]	Checks the swappiness OS configuration is within a recommended range
THLDirectoryWriteableCheck [461]	
THLListenerAddressCheck [461]	

Option	Description
THLSchemaChangeCheck [461]	Ensures that the existing THL format is compatible with the new release
THLStorageCheck [461]	Confirms the THL storage directory exists, is empty and writeable
THLStorageChecksum [461]	
TargetDirectoryDoesNotExist [461]	
TransferredLogStorageCheck [461]	
UpgradeSameProductCheck [461]	Ensures that the same product is being updated
VIPEnabledHostAllowsRootCommands [461]	
VIPEnabledHostArpPath [461]	
VIPEnabledHostIfconfigPath [462]	
VerticaUserGroupsCheck [462]	Checks that the Vertica user has the correct OS group membership
WhichAvailableCheck [462]	Checks the existence of a working which command
WriteableHomeDirectoryCheck [462]	Ensures the home directory can be written to
WriteableTempDirectoryCheck [462]	Ensures the temporary directory can be written to
XtrabackupAvailableCheck [462]	
XtrabackupDirectoryWriteableCheck [462]	
XtrabackupSettingsCheck [462]	

– BackupDirectoryWriteableCheck [448]

Option	BackupDirectoryWriteableCheck [448]
Description	Checks that the configured backup directory is writeable

Confirms that the directory defined in backup-directory [475] directory exists and can be written to.

– BackupDumpDirectoryWriteableCheck [448]

Option	BackupDumpDirectoryWriteableCheck [448]
Description	Checks the backup temp directory is writeable

Confirms that the directory defined in backup-dump-directory [475] directory exists and can be written to.

– BackupScriptAvailableCheck [448]

Option	BackupScriptAvailableCheck [448]
Description	Checks that the configured backup script exists and can be executed

Confirms that the script defined in backup-script [476] exists and is executable.

– ClusterDiagnosticCheck [448]

Option	ClusterDiagnosticCheck [448]
Description	

– ClusterStatusCheck [448]

Option	ClusterStatusCheck [448]
Description	

– CommitDirectoryCheck [448]

Option	CommitDirectoryCheck [448]

Description	

– ConfigurationStorageDirectoryCheck [449]

Option	ConfigurationStorageDirectoryCheck [449]
Description	

– ConfigureValidationCheck [449]

Option	ConfigureValidationCheck [449]
Description	

– ConfiguredDirectoryCheck [449]

Option	ConfiguredDirectoryCheck [449]
Description	

– ConflictingReplicationServiceTHLPortsCheck [449]

Option	ConflictingReplicationServiceTHLPortsCheck [449]
Description	

– ConnectorChecks [449]

Option	ConnectorChecks [449]
Description	Ensures that the configured connector selection is valid

Checks that the list of connectors and the corresponding list of data services is valid.

– ConnectorDBVersionCheck [449]

Option	ConnectorDBVersionCheck [449]
Description	

– ConnectorListenerAddressCheck [449]

Option	ConnectorListenerAddressCheck [449]
Description	

– ConnectorRWROAddressesCheck [449]

Option	ConnectorRWROAddressesCheck [449]
Description	Ensure the RW and RO addresses are different

For environments where the connector has been configured to use different hosts and ports for RW and RO operations, ensure that the settings are in fact different.

– ConnectorSmartScaleAllowedCheck [449]

Option	ConnectorSmartScaleAllowedCheck [449]
Description	Confirms whether SmartScale is valid within the current configured parameters

Checks that both SmartScale and Read/Write splitting have been enabled.

– ConnectorUserCheck [449]

Option	ConnectorUserCheck [449]

Description	

– ConsistentReplicationCredentialsCheck [450]

Option	ConsistentReplicationCredentialsCheck [450]
Description	

– CurrentCommandCoordinatorCheck [450]

Option	CurrentCommandCoordinatorCheck [450]
Description	

– CurrentConnectorCheck [450]

Option	CurrentConnectorCheck [450]
Description	

– CurrentReleaseDirectoryIsSymlink [450]

Option	CurrentReleaseDirectoryIsSymlink [450]
Description	

– CurrentTopologyCheck [450]

Option	CurrentTopologyCheck [450]
Description	

– CurrentVersionCheck [450]

Option	CurrentVersionCheck [450]
Description	

– DatasourceBootScriptCheck [450]

Option	DatasourceBootScriptCheck [450]
Description	

– DifferentMasterSlaveCheck [450]

Option	DifferentMasterSlaveCheck [450]
Description	

– DirectOracleServiceSIDCheck [450]

Option	DirectOracleServiceSIDCheck [450]
Description	

– EncryptionCheck [450]

Option	EncryptionCheck [450]
Description	

– EncryptionKeystoreCheck [450]

Option	EncryptionKeystoreCheck [450]

Description	

– FileValidationCheck [451]

Option	FileValidationCheck [451]
Description	

– FirewallCheck [451]

Option	FirewallCheck [451]
Description	

– GlobalHostAddressesCheck [451]

Option	GlobalHostAddressesCheck [451]
Description	

– GlobalHostOracleLibrariesFoundCheck [451]

Option	GlobalHostOracleLibrariesFoundCheck [451]
Description	

– GlobalMatchingPingMethodCheck [451]

Option	GlobalMatchingPingMethodCheck [451]
Description	

– GlobalRestartComponentsCheck [451]

Option	GlobalRestartComponentsCheck [451]
Description	

– GroupValidationCheck [451]

Option	GroupValidationCheck [451]
Description	

– HdfsValidationCheck [451]

Option	HdfsValidationCheck [451]
Description	

– HostLicensesCheck [451]

Option	HostLicensesCheck [451]
Description	

– HostOracleLibrariesFoundCheck [451]

Option	HostOracleLibrariesFoundCheck [451]
Description	

– HostReplicatorServiceRunningCheck [451]

Option	HostReplicatorServiceRunningCheck [451]

Description	

– HostSkippedChecks [452]

Option	HostSkippedChecks [452]
Description	

– HostnameCheck [452]

Option	HostnameCheck [452]
Description	

– HostsFileCheck [452]

Option	HostsFileCheck [452]
Description	

– InstallServicesCheck [452]

Option	InstallServicesCheck [452]
Description	

– InstallationScriptCheck [452]

Option	InstallationScriptCheck [452]
Description	

– InstallerMasterSlaveCheck [452]

Option	InstallerMasterSlaveCheck [452]
Description	Checks whether a Primary host has been defined for the configured service.

– InstallingOverExistingInstallation [452]

Option	InstallingOverExistingInstallation [452]
Description	

– JavaUserTimezoneCheck [452]

Option	JavaUserTimezoneCheck [452]
Description	

– JavaVersionCheck [452]

Option	JavaVersionCheck [452]
Description	

– KeystoresCheck [452]

Option	KeystoresCheck [452]
Description	

– KeystoresToCommitCheck [452]

Option	KeystoresToCommitCheck [452]

Description	

– ManagerActiveWitnessConversionCheck [453]

Option	ManagerActiveWitnessConversionCheck [453]
Description	

– ManagerChecks [453]

Option	ManagerChecks [453]
Description	

– ManagerHeapThresholdCheck [453]

Option	ManagerHeapThresholdCheck [453]
Description	

– ManagerListenerAddressCheck [453]

Option	ManagerListenerAddressCheck [453]
Description	

– ManagerPingMethodCheck [453]

Option	ManagerPingMethodCheck [453]
Description	

– ManagerWitnessAvailableCheck [453]

Option	ManagerWitnessAvailableCheck [453]
Description	

– ManagerWitnessNeededCheck [453]

Option	ManagerWitnessNeededCheck [453]
Description	

– MatchingHomeDirectoryCheck [453]

Option	MatchingHomeDirectoryCheck [453]
Description	

– MissingReplicationServiceConfigurationCheck [453]

Option	MissingReplicationServiceConfigurationCheck [453]
Description	

– ModifiedConfigurationFilesCheck [453]

Option	ModifiedConfigurationFilesCheck [453]
Description	

– MySQLAllowIntensiveChecks [453]

Option	MySQLAllowIntensiveChecks [453]

Description	Enables searching MySQL INFORMATION_SCHEMA for validation checks

Enables tpm to make use of the MySQL `INFORMATION_SCHEMA` to perform various validation checks. These include, but are not limited to:

- Tables not configured to use transactional tables
- Unsupported datatypes in MySQL tables

– `MySQLApplierLogsCheck` [454]

Option	`MySQLApplierLogsCheck` [454]
Description	

– `MySQLApplierPortCheck` [454]

Option	`MySQLApplierPortCheck` [454]
Description	

– `MySQLApplierServerIDCheck` [454]

Option	`MySQLApplierServerIDCheck` [454]
Description	

– `MySQLAvailableCheck` [454]

Option	`MySQLAvailableCheck` [454]
Description	Checks if MySQL is installed

– `MySQLBinaryLogsEnabledCheck` [454]

Option	`MySQLBinaryLogsEnabledCheck` [454]
Description	Checks that binary logging has been enabled on MySQL

Examines the `log_bin` variable has been defined within the running MySQL server. Binary logging must be enabled for replication to work.

– `MySQLBinlogDoDbCheck` [454]

Option	`MySQLBinlogDoDbCheck` [454]
Description	

– `MySQLClientCheck` [454]

Option	`MySQLClientCheck` [454]
Description	Checks whether the MySQL client command tool is available

– `MySQLConfigFileCheck` [454]

Option	`MySQLConfigFileCheck` [454]
Description	Checks the existence of a MySQL configuration file

– `MySQLConnectorBridgeModePermissionsCheck` [454]

Option	`MySQLConnectorBridgeModePermissionsCheck` [454]
Description	

– `MySQLConnectorPermissionsCheck` [454]

Option	`MySQLConnectorPermissionsCheck` [454]

Description	

– MySQLDefaultTableTypeCheck [455]

Option	MySQLDefaultTableTypeCheck [455]
Description	Checks the default table type for MySQL

Checks that the default table type configured for MySQL is a compatible transactional storage engine such as InnoDB

– MySQLDumpCheck [455]

Option	MySQLDumpCheck [455]
Description	Checks that the mysqldump command version matches the installed MySQL

Checks whether the mysqldump command within the configured PATH matches the version of MySQL being configured as a source or target. A mismatch could indicate that multiple MySQL versions are installed.

A mismatch could create invalid or corrupt backups. Either correct your PATH or use preferred-path [506] to point to the correct MySQL installation.

– MySQLGeneratedColumnCheck [455]

Option	MySQLGeneratedColumnCheck [455]
Description	Checks whether MySQL virtual/generated columns are defined

Checks, whether any tables contain generated or virtual columns. The test is only executed on MySQL 5.7+ and only if mysql-allow-intensive-checks [502] has been enabled.

– MySQLInnoDBEnabledCheck [455]

Option	MySQLInnoDBEnabledCheck [455]
Description	

– MySQLJsonDataTypeCheck [455]

Option	MySQLJsonDataTypeCheck [455]
Description	

Checks, whether any tables contain JSON columns. The test is only executed on MySQL 5.7+ and only if mysql-allow-intensive-checks [502] has been enabled.

– MySQLLoadDataInfilePermissionsCheck [455]

Option	MySQLLoadDataInfilePermissionsCheck [455]
Description	

– MySQLLoginCheck [455]

Option	MySQLLoginCheck [455]
Description	Checks whether Tungsten Cluster can connect to MySQL using the configured credentials

– MySQLMyISAMCheck [455]

Option	MySQLMyISAMCheck [455]
Description	Checks for the existence of MyISAM tables

Checks for the existence of MyISAM tables within the database. Use of MyISAM tables is not supported since MyISAM is not transactionally consistent. This can cause problems for both extraction and applying data.

In order to check for the existence of MyISAM tables, tpm uses two techniques:

- Looking for `.MYD` files within the MySQL directory, which are the files which contains MyISAM data. tpm must be able to read and see the contents of the MySQL data directory. If the configured user does not already have access, you can use the `root-command-prefix=true` [492] option to grant root access to access the filesystem.
- Using the MySQL `INFORMATION_SCHEMA` to look for tables defined with the MyISAM engine. For this option to work, intensive checks must have been enabled using `mysql-allow-intensive-checks` [502].

If neither of these methods is available, the check will fail and installation will stop.

– `MySQLNoMySQLReplicationCheck` [456]

Option	`MySQLNoMySQLReplicationCheck` [456]
Description	

– `MySQLPasswordSettingCheck` [456]

Option	`MySQLPasswordSettingCheck` [456]
Description	

– `MySQLPermissionsCheck` [456]

Option	`MySQLPermissionsCheck` [456]
Description	

– `MySQLReadableLogsCheck` [456]

Option	`MySQLReadableLogsCheck` [456]
Description	

– `MySQLSettingsCheck` [456]

Option	`MySQLSettingsCheck` [456]
Description	

– `MySQLSuperReadOnlyCheck` [456]

Option	`MySQLSuperReadOnlyCheck` [456]
Description	Checks whether super_read_only has been enabled on MySQL

Checks whether the `super_read_only` variable within MySQL has been enabled. If enabled, replication will not work. The check will test both the running server and the configuration file to determine whether the value has been enabled.

– `MySQLTriggerCheck` [456]

Option	`MySQLTriggerCheck` [456]
Description	

– `MySQLUnsupportedDataTypesCheck` [456]

Option	`MySQLUnsupportedDataTypesCheck` [456]
Description	

– `MysqlConnectorCheck` [456]

Option	`MysqlConnectorCheck` [456]

Description	

– MysqldumpAvailableCheck [457]

Option	MysqldumpAvailableCheck [457]
Description	

– MysqldumpSettingsCheck [457]

Option	MysqldumpSettingsCheck [457]
Description	

– NewDirectoryRequiredCheck [457]

Option	NewDirectoryRequiredCheck [457]
Description	

– NtpdRunningCheck [457]

Option	NtpdRunningCheck [457]
Description	

– OSCheck [457]

Option	OSCheck [457]
Description	

– OldServicesRunningCheck [457]

Option	OldServicesRunningCheck [457]
Description	

– OpenFilesLimitCheck [457]

Option	OpenFilesLimitCheck [457]
Description	

– OpensslLibraryCheck [457]

Option	OpensslLibraryCheck [457]
Description	

– OracleLoginCheck [457]

Option	OracleLoginCheck [457]
Description	

– OraclePermissionsCheck [457]

Option	OraclePermissionsCheck [457]
Description	

– OracleRedoReaderMinerDirectoryCheck [457]

Option	OracleRedoReaderMinerDirectoryCheck [457]

Description	

– OracleServiceSIDCheck [458]

Option	OracleServiceSIDCheck [458]
Description	

– OracleVersionCheck [458]

Option	OracleVersionCheck [458]
Description	

– PGAvailableCheck [458]

Option	PGAvailableCheck [458]
Description	

– ParallelReplicationCheck [458]

Option	ParallelReplicationCheck [458]
Description	

– ParallelReplicationCountCheck [458]

Option	ParallelReplicationCountCheck [458]
Description	

– PgControlAvailableCheck [458]

Option	PgControlAvailableCheck [458]
Description	

– PgStandbyAvailableCheck [458]

Option	PgStandbyAvailableCheck [458]
Description	

– PgdumpAvailableCheck [458]

Option	PgdumpAvailableCheck [458]
Description	

– PgdumpallAvailableCheck [458]

Option	PgdumpallAvailableCheck [458]
Description	

– PingSyntaxCheck [458]

Option	PingSyntaxCheck [458]
Description	

– PortAvailabilityCheck [458]

Option	PortAvailabilityCheck [458]

Description	

– ProfileScriptCheck [459]

Option	ProfileScriptCheck [459]
Description	

– RMIListenerAddressCheck [459]

Option	RMIListenerAddressCheck [459]
Description	

– RelayDirectoryWriteableCheck [459]

Option	RelayDirectoryWriteableCheck [459]
Description	Checks that the relay log directory can be written to

Confirms that the directory defined in relay-log-dir directory exists and can be written to.

– ReplicatorChecks [459]

Option	ReplicatorChecks [459]
Description	

– RestartComponentsCheck [459]

Option	RestartComponentsCheck [459]
Description	

– RouterAffinityCheck [459]

Option	RouterAffinityCheck [459]
Description	

– RouterBridgeModeDefaultCheck [459]

Option	RouterBridgeModeDefaultCheck [459]
Description	

– RouterDelayBeforeOfflineCheck [459]

Option	RouterDelayBeforeOfflineCheck [459]
Description	

– RouterKeepAliveTimeoutCheck [459]

Option	RouterKeepAliveTimeoutCheck [459]
Description	

– RowBasedBinaryLoggingCheck [459]

Option	RowBasedBinaryLoggingCheck [459]
Description	Checks that Row-based binary logging has been enabled for heterogeneous deployments

For heterogeneous deployments, row-based binary logging must have been enabled. For all services where heterogeneous support has been enabled, for example due to `enable-heterogeneous-service` [491] or `enable-batch-service`, row-based logging within MySQL must have been switched on. The test looks for the value of `binlog_format=ROW`.

– `RsyncAvailableCheck` [460]

Option	`RsyncAvailableCheck` [460]
Description	

– `RubyVersionCheck` [460]

Option	`RubyVersionCheck` [460]
Description	

– `SSHLoginCheck` [460]

Option	`SSHLoginCheck` [460]
Description	Checks connectivity to other hosts over SSH

Checks to confirm the SSH logins to other hosts in the cluster work, without requiring a password, and without returning additional rows of information when directly, remotely, running a command.

In the event of the check failing, the following items should be checked:

- Confirm that it is possible to SSH to the remote site using the username provided, and without requiring a password. For example:

```
host1-shell> ssh tungsten@host2
Last login: Wed Aug 9 09:55:23 2017 from fe80::1042:8aee:61da:a20%en0
host2-shell>
```

- Remove any remote messages returned when the user logs in. This includes the output from the *`Banner`* argument within `/etc/ssh/sshd_config`, or text or files output by the users shell login script or profile.

- Ensure that your remote shell has not been configured to output text or a message when a logout is attempted, for example by using:

```
shell> trap "echo logout" 0
```

– `ServiceTransferredLogStorageCheck` [460]

Option	`ServiceTransferredLogStorageCheck` [460]
Description	

– `StartingStoppedServices` [460]

Option	`StartingStoppedServices` [460]
Description	

– `SudoCheck` [460]

Option	`SudoCheck` [460]
Description	

– `SwappinessCheck` [460]

Option	`SwappinessCheck` [460]
Description	Checks the swappiness OS configuration is within a recommended range

Checks whether the Linux swappiness parameter has been set to a value of 10 or less, both in the current setting and when the system reboots. A value greater than 10 may allow for running programs to be swapped out, which will affect the performance of the Tungsten Clustering when running. Change the value in `sysctl.conf`.

– THLDirectoryWriteableCheck [461]

Option	THLDirectoryWriteableCheck [461]
Description	

– THLListenerAddressCheck [461]

Option	THLListenerAddressCheck [461]
Description	

– THLSchemaChangeCheck [461]

Option	THLSchemaChangeCheck [461]
Description	Ensures that the existing THL format is compatible with the new release

Checks that the format of the current THL is compatible with the schema and format of the new software. A difference may mean that the THL needs to be reset before installation can continue.

– THLStorageCheck [461]

Option	THLStorageCheck [461]
Description	Confirms the THL storage directory exists, is empty and writeable

Confirms that the directory configured for THL storage using log-dir directory exists, is writeable, and is empty.

– THLStorageChecksum [461]

Option	THLStorageChecksum [461]
Description	

– TargetDirectoryDoesNotExist [461]

Option	TargetDirectoryDoesNotExist [461]
Description	

– TransferredLogStorageCheck [461]

Option	TransferredLogStorageCheck [461]
Description	

– UpgradeSameProductCheck [461]

Option	UpgradeSameProductCheck [461]
Description	Ensures that the same product is being updated

Updates must occur with the same product, for example, Tungsten Replicator to Tungsten Replicator. It is not possible to update replicator to cluster, or cluster to replicator.

– VIPEnabledHostAllowsRootCommands [461]

Option	VIPEnabledHostAllowsRootCommands [461]
Description	

– VIPEnabledHostArpPath [461]

Option	VIPEnabledHostArpPath [461]

Description	

– VIPEnabledHostIfconfigPath [462]

Option	VIPEnabledHostIfconfigPath [462]
Description	

– VerticaUserGroupsCheck [462]

Option	VerticaUserGroupsCheck [462]
Description	Checks that the Vertica user has the correct OS group membership

Checks whether the user running Vertica is a member of the tungsten user's primary group. Without this setting, the CSV files generated by the replicator would not be readable by Vertica when importing them into the database during batchloading.

– WhichAvailableCheck [462]

Option	WhichAvailableCheck [462]
Description	Checks the existence of a working which command

Checks the existence of a working which command.

– WriteableHomeDirectoryCheck [462]

Option	WriteableHomeDirectoryCheck [462]
Description	Ensures the home directory can be written to

Checks that the home directory for the configured user can be written to.

– WriteableTempDirectoryCheck [462]

Option	WriteableTempDirectoryCheck [462]
Description	Ensures the temporary directory can be written to

The temporary directory is used during installation to store a variety of information. This check ensures that the directory is writeable, and that files can be created and deleted correctly.

– XtrabackupAvailableCheck [462]

Option	XtrabackupAvailableCheck [462]
Description	

– XtrabackupDirectoryWriteableCheck [462]

Option	XtrabackupDirectoryWriteableCheck [462]
Description	

– XtrabackupSettingsCheck [462]

Option	XtrabackupSettingsCheck [462]
Description	

10.8. tpm Configuration Options

tpm supports a large range of configuration options, which can be specified either:

- In an INI file, without the double-dash prefix, i.e. repl-thl-log-retention=3d [520]
- On the command-line, using a double-dash prefix, i.e. --repl-thl-log-retention=3d [520]

A full list of all the available options supported is provided in Table 10.16, "tpm Configuration Options".

Table 10.16. tpm Configuration Options

CmdLine Option	INI File Option	Description
--application-password [473], --connector-password [473]	application-password [473], connector-password [473]	Database password for the connector
--application-port [473], --connector-listen-port [473]	application-port [473], connector-listen-port [473]	Port for the connector to listen on
--application-readonly-port [474], --connector-readonly-listen-port [474]	application-readonly-port [474], connector-readonly-listen-port [474]	Port for the connector to listen for read-only connections on
--application-user [474], --connector-user [474]	application-user [474], connector-user [474]	Database username for the connector
--auto-enable [474], --repl-auto-enable [474]	auto-enable [474], repl-auto-enable [474]	Auto-enable services after start-up
--auto-recovery-delay-interval [474], --repl-auto-recovery-delay-interval [474]	auto-recovery-delay-interval [474], repl-auto-recovery-delay-interval [474]	Delay (in seconds) between going OFFLINE and attempting to go ONLINE
--auto-recovery-max-attempts [474], --repl-auto-recovery-max-attempts [474]	auto-recovery-max-attempts [474], repl-auto-recovery-max-attempts [474]	Maximum number of attempts at automatic recovery
--auto-recovery-reset-interval [474], --repl-auto-recovery-reset-interval [474]	auto-recovery-reset-interval [474], repl-auto-recovery-reset-interval [474]	Delay (in seconds) before autorecovery is deemed to have succeeded
--backup-directory [475], --repl-backup-directory [475]	backup-directory [475], repl-backup-directory [475]	Permanent backup storage directory
--backup-dump-directory [475], --repl-backup-dump-directory [475]	backup-dump-directory [475], repl-backup-dump-directory [475]	Backup temporary dump directory
--backup-method [475], --repl-backup-method [475]	backup-method [475], repl-backup-method [475]	Database backup method
--backup-online [475], --repl-backup-online [475]	backup-online [475], repl-backup-online [475]	Does the backup script support backing up a datasource while it is ONLINE
--backup-options [476]	backup-options [476]	Space separated list of options to pass directly to the underlying backup binary in use.
--backup-retention [476], --repl-backup-retention [476]	backup-retention [476], repl-backup-retention [476]	Number of backups to retain
--backup-script [476], --repl-backup-script [476]	backup-script [476], repl-backup-script [476]	What is the path to the backup script
--batch-enabled [476]	batch-enabled [476]	Should the replicator service use a batch applier
--batch-load-language [476]	batch-load-language [476]	Which script language to use for batch loading
--batch-load-template [476]	batch-load-template [476]	Value for the loadBatchTemplate property
--property=bridgeServerToClient-ForcedCloseTimeout [443]	property=bridgeServerToClientForced-CloseTimeout [443]	Adjust the timeout for Bridge Mode to force close connections (supplied in ms)
--buffer-size [477], --repl-buffer-size [477]	buffer-size [477], repl-buffer-size [477]	Replicator queue size between stages (min 1)
--cctrl-column-width [477]	cctrl-column-width [477]	Sets the minimum column width for service names within cctrl
--channels [477], --repl-channels [477]	channels [477], repl-channels [477]	Number of replication channels to use for parallel apply.
--cluster-slave-auto-recovery-delay-interval [477], --cluster-slave-repl-auto-recovery-delay-interval [477]	cluster-slave-auto-recovery-delay-interval [477], cluster-slave-repl-auto-recovery-delay-interval [477]	Default value for --auto-recovery-delay-interval when --topology=cluster-slave
--cluster-slave-auto-recovery-max-attempts [477], --clus-	cluster-slave-auto-recovery-max-attempts [477], cluster-slave-	Default value for --auto-recovery-max-attempts when --topology=cluster-slave

CmdLine Option	INI File Option	Description
ter-slave-repl-auto-recovery-max-attempts [477]	repl-auto-recovery-max-attempts [477]	
--cluster-slave-auto-recovery-reset-interval [478], --cluster-slave-repl-auto-recovery-reset-interval [478]	cluster-slave-auto-recovery-reset-interval [478], cluster-slave-repl-auto-recovery-reset-interval [478]	Default value for --auto-recovery-reset-interval when --topology=cluster-slave
--composite-datasources [478], --dataservice-composite-datasources [478]	composite-datasources [478], dataservice-composite-datasources [478]	Data services that should be added to this composite data service
--config-file [478]	config-file [478]	Display help information for content of the config file
--conn-java-enable-concurrent-gc [478]	conn-java-enable-concurrent-gc [478]	Connector Java uses concurrent garbage collection
--conn-java-mem-size [478]	conn-java-mem-size [478]	Connector Java heap memory size used to buffer data between clients and databases
--conn-round-robin-include-master [478]	conn-round-robin-include-master [478]	Should the Connector include the Primary in round-robin load balancing
--connector-affinity [478]	connector-affinity [478]	The default affinity for all connections
--connector-allow-cross-site-reconnects-for-reads [479]	connector-allow-cross-site-reconnects-for-reads [479]	Whether or not to reject reconnections that follow a read operation (RO_RELAXED or SmartScale after a read)
--connector-allow-cross-site-reconnects-for-writes [479]	connector-allow-cross-site-reconnects-for-writes [479]	Whether or not to reject reconnections that follow a write operation (RW_STRICT connection or SmartScale after a write)
--connector-autoreconnect [479]	connector-autoreconnect [479]	Enable auto-reconnect in the connector
--connector-autoreconnect-killed-connections [479]	connector-autoreconnect-killed-connections [479]	Enable autoreconnect for connections killed within the connector
--connector-bridge-mode [480], --enable-connector-bridge-mode [480]	connector-bridge-mode [480], enable-connector-bridge-mode [480]	Enable the Tungsten Connector bridge mode
--connector-db-version [480]	connector-db-version [480]	Server version as advertised by the Connector (proxy-mode only). When not defined or set to autodetect, the version will be set to the same as the primary server. You can override it here, preferably keeping the actual underlying server version with "-tungsten" to indicate client applications that the connection goes through the connector. Note that this option only applies to proxy mode, since bridge mode will always forward the version of the connected node, primary or replica depending on the bridge mode setting.
--connector-default-schema [480], --connector-forced-schema [480]	connector-default-schema [480], connector-forced-schema [480]	Default schema for the connector to use
--connector-delete-user-map [480]	connector-delete-user-map [480]	Overwrite an existing user.map file
--connector-disable-connection-warnings [480]	connector-disable-connection-warnings [480]	Hide Connector warnings in log files
--connector-disconnect-timeout [480]	connector-disconnect-timeout [480]	Time (in seconds) to wait for active connection to disconnect before forcing them closed [default: 5]
--connector-drop-after-max-connections [480]	connector-drop-after-max-connections [480]	Instantly drop connections that arrive after --connector-max-connections has been reached
--connector-keepalive-timeout [481]	connector-keepalive-timeout [481]	Delay (in ms) after which a manager connection is considered broken by the connector if no keep-alive command was received. Value must be positive and lower than 300000 (5 min) or the connector will refuse to start. Make sure the manager has mgr-monitor-interval value set greater than this, it controls the frequency at which keep-alive commands are sent.
--connector-listen-interface [481]	connector-listen-interface [481]	Listen interface to use for the connector
--connector-manager-use-ssl [481]	connector-manager-use-ssl [481]	Manager<>Connector connections can be encrypted with TLS by setting this to true or with the --disable-security-controls=false flags. Note that when enabled, managers still accept unencrypted connector connections in order to allow upgrades from older versions or unencrypted setups

CmdLine Option	INI File Option	Description
--connector-max-connections [481]	connector-max-connections [481]	The maximum number of connections the connector should allow at any time
--connector-max-slave-latency [481], --connector-max-applied-latency [481]	connector-max-applied-latency [481], connector-max-slave-latency [481]	The maximum applied latency for replica connections. Disabled by default.
--connector-readonly [481], --enable-connector-readonly [481]	connector-readonly [481], enable-connector-readonly [481]	Enable the Tungsten Connector read-only mode
--connector-relative-slave-status [482]	connector-relative-slave-status [482]	Determines whether the reported latency is relative (true) or absolute (false). Default is the value of dataservice-use-relative-latency [485] which itself has the default of false.
--connector-reset-when-affinity-back [482]	connector-reset-when-affinity-back [482]	Forces reconnection of clients to datasources with the configured affinity when they become available
--connector-rest-api [482]	connector-rest-api [482]	Enable (default) or Disable APIv2
--connector-rest-api-address [482]	connector-rest-api-address [482]	Address for the API to bind too.
--connector-rest-api-authentication [482]	connector-rest-api-authentication [482]	Enforce authentication for the API.
--connector-rest-api-port [482]	connector-rest-api-port [482]	Port for the Connector API.
--connector-rest-api-ssl [483]	connector-rest-api-ssl [483]	Enable SSL for the API.
--connector-ro-addresses [483]	connector-ro-addresses [483]	Connector addresses that should receive a r/o connection
--connector-rw-addresses [483]	connector-rw-addresses [483]	Connector addresses that should receive a r/w connection
--connector-rwsplitting [483]	connector-rwsplitting [483]	Enable DirectReads R/W splitting in the connector
--connector-server-ssl-ciphers [483]	connector-server-ssl-ciphers [483]	Configures ciphers the connector uses for SSL communications to the dbatabase server. Defaults to allow all cipher suites supported by the running JVM.
--connector-server-ssl-protocols [483]	connector-server-ssl-protocols [483]	Configures protocols the connector uses for SSL communications to the dbatabase server. Default value when not specified will be TLSv1,TLSv1.1,TLSv1.2
--connector-smartscale [483]	connector-smartscale [483]	Enable SmartScale R/W splitting in the connector
--connector-smartscale-session-id [484]	connector-smartscale-session-id [484]	The default session ID to use with smart scale
--connector-ssl-capable [484]	connector-ssl-capable [484]	Defines whether connector ports are advertised as being capable of SSL
--connectors [484], --dataservice-connectors [484]	connectors [484], dataservice-connectors [484]	Hostnames for the dataservice connectors
--consistency-policy [484], --repl-consistency-policy [484]	consistency-policy [484], repl-consistency-policy [484]	Should the replicator stop or warn if a consistency check fails?
--dataservice-name [484]	dataservice-name [484]	Limit the command to the hosts in this dataservice Multiple data services may be specified by providing a comma separated list
--dataservice-relay-enabled [485]	dataservice-relay-enabled [485]	Make this dataservice a replica of another
--dataservice-schema [485]	dataservice-schema [485]	The db schema to hold dataservice details
--dataservice-thl-port [485]	dataservice-thl-port [485]	Port to use for THL operations
--dataservice-use-relative-latency [485], --use-relative-latency [485]	dataservice-use-relative-latency [485], use-relative-latency [485]	Enable the cluster to operate on relative latency
--dataservice-vip-enabled [485]	dataservice-vip-enabled [485]	Is VIP management enabled?
--dataservice-vip-ipaddress [485]	dataservice-vip-ipaddress [485]	VIP IP address
--dataservice-vip-netmask [485]	dataservice-vip-netmask [485]	VIP netmask
--datasource-boot-script [485], --repl-datasource-boot-script [485]	datasource-boot-script [485], repl-datasource-boot-script [485]	Database start script
--datasource-enable-ssl [486], --repl-datasource-enable-ssl [486]	datasource-enable-ssl [486], repl-datasource-enable-ssl [486]	Enable SSL connection to DBMS server

CmdLine Option	INI File Option	Description
--datasource-log-directory [486], --repl-datasource-log-directory [486]	datasource-log-directory [486], repl-datasource-log-directory [486]	Primary log directory
--datasource-log-pattern [486], --repl-datasource-log-pattern [486]	datasource-log-pattern [486], repl-datasource-log-pattern [486]	Primary log filename pattern
--datasource-mysql-conf [486], --repl-datasource-mysql-conf [486]	datasource-mysql-conf [486], repl-datasource-mysql-conf [486]	MySQL config file
--datasource-mysql-data-directory [486], --repl-datasource-mysql-data-directory [486]	datasource-mysql-data-directory [486], repl-datasource-mysql-data-directory [486]	MySQL data directory
--datasource-mysql-ibdata-directory [486], --repl-datasource-mysql-ibdata-directory [486]	datasource-mysql-ibdata-directory [486], repl-datasource-mysql-ibdata-directory [486]	MySQL InnoDB data directory
--datasource-mysql-iblog-directory [487], --repl-datasource-mysql-iblog-directory [487]	datasource-mysql-iblog-directory [487], repl-datasource-mysql-iblog-directory [487]	MySQL InnoDB log directory
--datasource-mysql-ssl-ca [487], --repl-datasource-mysql-ssl-ca [487]	datasource-mysql-ssl-ca [487], repl-datasource-mysql-ssl-ca [487]	MySQL SSL CA file
--datasource-mysql-ssl-cert [487], --repl-datasource-mysql-ssl-cert [487]	datasource-mysql-ssl-cert [487], repl-datasource-mysql-ssl-cert [487]	MySQL SSL certificate file
--datasource-mysql-ssl-key [487], --repl-datasource-mysql-ssl-key [487]	datasource-mysql-ssl-key [487], repl-datasource-mysql-ssl-key [487]	MySQL SSL key file
--datasource-systemctl-service [487], --repl-datasource-systemctl-service [487]	datasource-systemctl-service [487], repl-datasource-systemctl-service [487]	Database systemctl script
--datasource-type [487], --repl-datasource-type [487]	datasource-type [487], repl-datasource-type [487]	Database type
--delete [488]	delete [488]	Delete the named data service from the configuration Data Service options
--deploy-current-package [488]	deploy-current-package [488]	Deploy the current Tungsten package
--deploy-package-uri [488]	deploy-package-uri [488]	URL for the Tungsten package to deploy
--direct-datasource-log-directory [488], --repl-direct-datasource-log-directory [488]	direct-datasource-log-directory [488], repl-direct-datasource-log-directory [488]	Primary log directory
--direct-datasource-log-pattern [488], --repl-direct-datasource-log-pattern [488]	direct-datasource-log-pattern [488], repl-direct-datasource-log-pattern [488]	Primary log filename pattern
--direct-datasource-type [488], --repl-direct-datasource-type [488]	direct-datasource-type [488], repl-direct-datasource-type [488]	Database type
--direct-replication-host [489], --direct-datasource-host [489], --repl-direct-datasource-host [489]	direct-datasource-host [489], direct-replication-host [489], repl-direct-datasource-host [489]	Database server hostname
--direct-replication-password [489], --direct-datasource-password [489], --repl-direct-datasource-password [489]	direct-datasource-password [489], direct-replication-password [489], repl-direct-datasource-password [489]	Password for datasource connection
--direct-replication-port [489], --direct-datasource-port [489], --repl-direct-datasource-port [489]	direct-datasource-port [489], direct-replication-port [489], repl-direct-datasource-port [489]	Database server port
--direct-replication-user [489], --direct-datasource-user [489], --repl-direct-datasource-user [489]	direct-datasource-user [489], direct-replication-user [489], repl-direct-datasource-user [489]	Database login for Tungsten

CmdLine Option	INI File Option	Description
--directory [489]	directory [489]	Set the directory of an existing installation used during fetching an existing configuration
--disable-relay-logs [489], --repl-disable-relay-logs [489]	disable-relay-logs [489], repl-disable-relay-logs [489]	Disable the use of relay-logs?
--disable-security-controls [490]	disable-security-controls [490]	Disables all forms of security, including SSL, TLS and authentication
--disable-slave-extractor [490], --repl-disable-slave-extractor [490]	disable-slave-extractor [490], repl-disable-slave-extractor [490]	Should replica servers support the primary role?
--drop-static-columns-in-updates [490]	drop-static-columns-in-updates [490]	This will modify UPDATE transactions in row-based replication and eliminate any columns that were not modified.
--enable-active-witnesses [490], --active-witnesses [490]	active-witnesses [490], enable-active-witnesses [490]	Enable active witness hosts
--enable-connector-client-ssl [490], --connector-client-ssl [490]	connector-client-ssl [490], enable-connector-client-ssl [490]	Enable SSL encryption of traffic from the client to the connector
--enable-connector-server-ssl [490], --connector-server-ssl [490]	connector-server-ssl [490], enable-connector-server-ssl [490]	Enable SSL encryption of traffic from the connector to the database
--enable-connector-ssl [491], --connector-ssl [491]	connector-ssl [491], enable-connector-ssl [491]	Enable SSL encryption of connector traffic to the database
--enable-heterogeneous-master [491]	enable-heterogeneous-master [491]	Enable heterogeneous operation for the primary
--enable-heterogeneous-service [491]	enable-heterogeneous-service [491]	Enable heterogeneous operation
--enable-heterogeneous-slave [491]	enable-heterogeneous-slave [491]	Enable heterogeneous operation for the replica
--enable-jgroups-ssl [492], --jgroups-ssl [492]	enable-jgroups-ssl [492], jgroups-ssl [492]	Enable SSL encryption of JGroups communication on this host
--enable-rmi-authentication [492], --rmi-authentication [492]	enable-rmi-authentication [492], rmi-authentication [492]	Enable RMI authentication for the services running on this host
--enable-rmi-ssl [492], --rmi-ssl [492]	enable-rmi-ssl [492], rmi-ssl [492]	Enable SSL encryption of RMI communication on this host
--enable-slave-thl-listener [492], --repl-enable-slave-thl-listener [492]	enable-slave-thl-listener [492], repl-enable-slave-thl-listener [492]	Should this service allow THL connections?
--enable-sudo-access [492], --root-command-prefix [492]	enable-sudo-access [492], root-command-prefix [492]	Run root commands using sudo
--enable-thl-ssl [493], --repl-enable-thl-ssl [493], --thl-ssl [493]	enable-thl-ssl [493], repl-enable-thl-ssl [493], thl-ssl [493]	Enable SSL encryption of THL communication for this service
--executable-prefix [493]	executable-prefix [493]	Adds a prefix to command aliases
--file-protection-level [493]	file-protection-level [493]	Protection level for Continuent files
--file-protection-umask [493]	file-protection-umask [493]	Protection umask for Continuent files
--host-name [493]	host-name [493]	DNS hostname
--hosts [494]	hosts [494]	Limit the command to the hosts listed You must use the hostname as it appears in the configuration.
--hub [494], --dataservice-hub-host [494]	dataservice-hub-host [494], hub [494]	What is the hub host for this all-masters dataservice?
--hub-service [494], --dataservice-hub-service [494]	dataservice-hub-service [494], hub-service [494]	The data service to use for the hub of a star topology
--install [494]	install [494]	Install service start scripts
--install-directory [494], --home-directory [494]	home-directory [494], install-directory [494]	Installation directory

CmdLine Option	INI File Option	Description
--java-connector-keystore-password [494]	java-connector-keystore-password [494]	Set the password for unlocking the tungsten_connector_keystore.jks file in the security directory. Specific to connector<->mysql communication only.
--java-connector-keystore-path [494]	java-connector-keystore-path [494]	Local path to the Java Connector Keystore file. Specific to connector<->mysql communication only.
--java-connector-truststore-password [495]	java-connector-truststore-password [495]	Set the password for unlocking the tungsten_connector_truststore.jks file in the security directory. Specific to connector<->mysql communication only.
--java-connector-truststore-path [495]	java-connector-truststore-path [495]	Local path to the Java Connector Truststore file. Specific to connector<->mysql communication only.
--java-enable-concurrent-gc [495], --repl-java-enable-concurrent-gc [495]	java-enable-concurrent-gc [495], repl-java-enable-concurrent-gc [495]	Replicator Java uses concurrent garbage collection
--java-external-lib-dir [495], --repl-java-external-lib-dir [495]	java-external-lib-dir [495], repl-java-external-lib-dir [495]	Directory for 3rd party Jar files required by replicator
--java-file-encoding [495], --repl-java-file-encoding [495]	java-file-encoding [495], repl-java-file-encoding [495]	Java platform charset (esp. for heterogeneous replication)
--java-jgroups-key [495]	java-jgroups-key [495]	The alias to use for the JGroups TLS key in the keystore.
--java-jgroups-keystore-path [495]	java-jgroups-keystore-path [495]	Local path to the JGroups Java Keystore file.
--java-jmxremote-access-path [496]	java-jmxremote-access-path [496]	Local path to the Java JMX Remote Access file.
--java-keystore-password [496]	java-keystore-password [496]	Set the password for unlocking the tungsten_keystore.jks file in the security directory. Specific for intra cluster communication.
--java-keystore-path [496]	java-keystore-path [496]	Local path to the Java Keystore file. Specific for intra cluster communication. NOTE: When java-keystore-path is passed to tpm, the keystore must contain both tls and mysql certs when appropriate. tpm will NOT add mysql cert nor generate tls cert when this flag is found, so both certs must be manually imported already.
--java-mem-size [496], --repl-java-mem-size [496]	java-mem-size [496], repl-java-mem-size [496]	Replicator Java heap memory size in Mb (min 128)
--java-passwordstore-path [496]	java-passwordstore-path [496]	Local path to the Java Password Store file.
--java-thl-keystore-path [496]	java-thl-keystore-path [496]	The alias to use for storing the keystore associated with THL encryption.
--java-thl-truststore-path [496]	java-thl-truststore-path [496]	The alias to use for storing the truststore associated with THL encryption.
--java-tls-alias [497]	java-tls-alias [497]	The alias to use for the TLS key/certificate in the keystore and truststore.
--java-tls-key-lifetime [497]	java-tls-key-lifetime [497]	Lifetime for the Java TLS key
--java-tls-keystore-path [497]	java-tls-keystore-path [497]	The keystore holding a certificate to use for all Continuent TLS encryption.
--java-truststore-password [497]	java-truststore-password [497]	The password for unlocking the tungsten_truststore.jks file in the security directory
--java-truststore-path [497]	java-truststore-path [497]	Local path to the Java Truststore file.
--java-user-timezone [497], --repl-java-user-timezone [497]	java-user-timezone [497], repl-java-user-timezone [497]	Java VM Timezone (esp. for cross-site replication)
--log [497]	log [497]	Write all messages, visible and hidden, to this file. You may specify a filename, 'pid' or 'timestamp'.
--log-slave-updates [497]	log-slave-updates [497]	Should replicas log updates to binlog
--manager-replicator-offline-timeout [498]	manager-replicator-offline-timeout [498]	Timeout (in seconds) for the manager to wait until the replicator goes offline.
--manager-rest-api [498]	manager-rest-api [498]	Enable (default) or Disable APIv2
--manager-rest-api-address [498]	manager-rest-api-address [498]	Address for the API to bind too.

CmdLine Option	INI File Option	Description
--manager-rest-api-authentication [498]	manager-rest-api-authentication [498]	Enforce authentication for the API.
--manager-rest-api-port [498]	manager-rest-api-port [498]	Port for the manager API.
--manager-rest-api-ssl [498]	manager-rest-api-ssl [498]	Enable SSL for the API.
--master [499], --dataservice-master-host [499], --masters [499], --relay [499]	dataservice-master-host [499], master [499], masters [499], relay [499]	Hostname of the primary (or relay) host within this service
--master-preferred-role [499], --repl-master-preferred-role [499]	master-preferred-role [499], repl-master-preferred-role [499]	Preferred role for primary THL when connecting as a replica
--master-services [499], --dataservice-master-services [499]	dataservice-master-services [499], master-services [499]	Data service names that should be used on each Primary
--master-thl-host [499]	master-thl-host [499]	Primary THL Hostname
--master-thl-port [499]	master-thl-port [499]	Primary THL Port
--members [499], --dataservice-hosts [499]	dataservice-hosts [499], members [499]	Hostnames for the dataservice members
--metadata-directory [499], --repl-metadata-directory [499]	metadata-directory [499], repl-metadata-directory [499]	Replicator metadata directory
--mgr-api [500]	mgr-api [500]	Enable the Manager API
--mgr-api-address [500]	mgr-api-address [500]	Address for the Manager API
--mgr-api-full-access [500]	mgr-api-full-access [500]	Enable all Manager API commands. Only the status command will be enabled without it.
--mgr-api-port [500]	mgr-api-port [500]	Port for the Manager API
--mgr-group-communication-port [500]	mgr-group-communication-port [500]	Port to use for manager group communication
--mgr-heap-threshold [500]	mgr-heap-threshold [500]	Java memory usage (MB) that will force a Manager restart
--mgr-java-enable-concurrent-gc [500]	mgr-java-enable-concurrent-gc [500]	Manager Java uses concurrent garbage collection
--mgr-java-mem-size [501]	mgr-java-mem-size [501]	Manager Java heap memory size in Mb (min 128)
--mgr-listen-interface [501]	mgr-listen-interface [501]	Listen interface to use for the manager
--mgr-monitor-interval [501]	mgr-monitor-interval [501]	Frequency, in milliseconds, at which managers send keep-alive commands to connectors.
--mgr-ping-method [501]	mgr-ping-method [501]	Mechanism to use when identifying the liveness of other data-sources (ping, echo)
--mgr-policy-mode [501]	mgr-policy-mode [501]	Manager policy mode
--mgr-rmi-port [501]	mgr-rmi-port [501]	Port to use for the manager RMI server
--mgr-rmi-remote-port [501]	mgr-rmi-remote-port [501]	Port to use for calling the remote manager RMI server
--mgr-ro-slave [502]	mgr-ro-slave [502]	Make replicas read-only
--mgr-vip-arp-path [502]	mgr-vip-arp-path [502]	Path to the arp binary
--mgr-vip-device [502]	mgr-vip-device [502]	VIP network device
--mgr-vip-ifconfig-path [502]	mgr-vip-ifconfig-path [502]	Path to the ifconfig binary
--mgr-wait-for-members [502]	mgr-wait-for-members [502]	Wait for all datasources to be available before completing installation
--mon-db-query-timeout [502]	mon-db-query-timeout [502]	Defines the timeout (in secs) for the managers mysql_checker_query.sql call
--mysql-allow-intensive-checks [502]	mysql-allow-intensive-checks [502]	For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility.
--mysql-driver [503]	mysql-driver [503]	MySQL Driver Vendor
--mysql-enable-ansiquotes [503], --repl-mysql-enable-ansiquotes [503]	mysql-enable-ansiquotes [503], repl-mysql-enable-ansiquotes [503]	Enables ANSI_QUOTES mode for incoming events?

CmdLine Option	INI File Option	Description
--mysql-enable-enumtostring [503], --repl-mysql-enable-enum-tostring [503]	mysql-enable-enumtostring [503], repl-mysql-enable-enum-tostring [503]	Enable a filter to convert ENUM values to strings
--mysql-enable-noonlykey-words [503], --repl-mysql-en-able-noonlykeywords [503]	mysql-enable-noonlykeywords [503], repl-mysql-enable-noonlykey-words [503]	Enables a filter to translate DELETE FROM ONLY to DELETE FROM and UPDATE ON-LY to UPDATE.
--mysql-enable-settostring [503], --repl-mysql-enable-set-tostring [503]	mysql-enable-settostring [503], re-pl-mysql-enable-settostring [503]	Enable a filter to convert SET types to strings
--mysql-ro-slave [503], --re-pl-mysql-ro-slave [503]	mysql-ro-slave [503], repl-mysql-ro-slave [503]	Replicas are read-only?
--mysql-server-id [503], --re-pl-mysql-server-id [503]	mysql-server-id [503], repl-mysql-server-id [503]	Explicitly set the MySQL server ID
--mysql-use-bytes-for-string [504], --repl-mysql-use-bytes-for-string [504]	mysql-use-bytes-for-string [504], repl-mysql-use-bytes-for-string [504]	Transfer strings as their byte representation?
--mysql-xtrabackup-dir [504], --re-pl-mysql-xtrabackup-dir [504]	mysql-xtrabackup-dir [504], re-pl-mysql-xtrabackup-dir [504]	Directory to use for storing xtrabackup full & incremental backups
--native-slave-takeover [504], --repl-native-slave-takeover [504]	native-slave-takeover [504], re-pl-native-slave-takeover [504]	Takeover native replication
--no-connectors [504]	no-connectors [504]	When issued during an update, connectors will not be restarted. Restart of the connectors will then need to be performed mannually for updates to take affect.
--no-deployment [504]	no-deployment [504]	Skip deployment steps that create the install directory
--no-validation [504]	no-validation [504]	Skip validation checks that run on each host
--optimize-row-events [505]	optimize-row-events [505]	Enables or disables optimized row updates. Enabled by default.
--optimize-row-events-limit-delete-rows [505]	optimize-row-events-limit-delete-rows [505]	Limits the number of deletes grouped per event when optimize-row-events is enabled. Note that deletes can only be optimized for tables with single column PK's
--optimize-row-events-limit-in-sert-rows [505]	optimize-row-events-limit-in-sert-rows [505]	Limits the number of inserts grouped per event when optimize-row-events is enabled.
--policy-relay-from-slave [505]	policy-relay-from-slave [505]	Cross-site replicators within a Composite Active/Active deployment can be configured to point to Replicas by default, and to prefer Replicas over Primaries during operation. In a standard deployment, cross-site replicators work via Primaries at each cluster site to read the remote information. To configure the service to use Replicas in preference to Primaries, use the policy-relay-from-slave=true [505] option to tpm. Both Primaries and Replicas remain in the list of possible hosts, so if no Replicas are available during a switch or failover event, then a Primary will be used.
--prefer-ip-stack [505]	prefer-ip-stack [505]	Switch between IPv4 and IPv6 support.
--preferred-path [506]	preferred-path [506]	Additional command path
--prefetch-enabled [506]	prefetch-enabled [506]	Should the replicator service be setup as a prefetch applier
--prefetch-max-time-ahead [506]	prefetch-max-time-ahead [506]	Maximum number of seconds that the prefetch applier can get in front of the standard applier
--prefetch-min-time-ahead [506]	prefetch-min-time-ahead [506]	Minimum number of seconds that the prefetch applier must be in front of the standard applier
--prefetch-schema [506]	prefetch-schema [506]	Schema to watch for timing prefetch progress
--prefetch-sleep-time [506]	prefetch-sleep-time [506]	How long to wait when the prefetch applier gets too far ahead
--privileged-master [507]	privileged-master [507]	Does the login for the Primary database service have superuser privileges
--privileged-slave [507]	privileged-slave [507]	Does the login for the Replica database service have superuser privileges
--profile-script [507]	profile-script [507]	Append commands to include env.sh in this profile script

CmdLine Option	INI File Option	Description
--protect-configuration-files [507]	protect-configuration-files [507]	When enabled, configuration files are protected to be only readable and updatable by the configured user
--relay-directory [508], --repl-relay-directory [508]	relay-directory [508], repl-relay-directory [508]	Directory for logs transferred from the Primary
--relay-enabled [508]	relay-enabled [508]	Should the replicator service be setup as a relay.
--relay-source [508], --dataservice-relay-source [508], --master-dataservice [508]	dataservice-relay-source [508], master-dataservice [508], relay-source [508]	Dataservice name to use as a relay source
--repl-allow-bidi-unsafe [508]	repl-allow-bidi-unsafe [508]	Allow unsafe SQL from remote service
--repl-store-thl-compressed [508]	repl-store-thl-compressed [508]	Enable (true) or disable (false) compression of THL on disk.
--repl-store-thl-encrypted [509]	repl-store-thl-encrypted [509]	Enable (true) or disable (false) encryption of THL on disk.
--repl-svc-extractor-multi-frag-service-detection [509]	repl-svc-extractor-multi-frag-service-detection [509]	Force extraction to read ahead to last fragment to detect service name
--repl-thl-client-serialization [509]	repl-thl-client-serialization [509]	Enable THL compression on downstream Replicator Appliers.
--repl-thl-server-serialization [509]	repl-thl-server-serialization [509]	Comma Separated list of THL compression protocols to enable on the thl-server (Extractor).
--replace-tls-certificate [509]	replace-tls-certificate [509]	Replace the TLS certificate
--replica-tx-commit-level [510]	replica-tx-commit-level [510]	If set, determines the value of the underlying database property innodb_flush_log_at_trx_commit when a node becomes a replica.
--replication-host [510], --datasource-host [510], --repl-datasource-host [510]	datasource-host [510], repl-datasource-host [510], replication-host [510]	Hostname of the datasource
--replication-password [510], --datasource-password [510], --repl-datasource-password [510]	datasource-password [510], repl-datasource-password [510], replication-password [510]	Database password
--replication-port [510], --datasource-port [510], --repl-datasource-port [510]	datasource-port [510], repl-datasource-port [510], replication-port [510]	Database network port
--replication-user [510], --datasource-user [510], --repl-datasource-user [510]	datasource-user [510], repl-datasource-user [510], replication-user [510]	User for database connection
replicator.stage.q-to-dbms.blockCommitPolicy [511]	replicator.stage.q-to-dbms.blockCommitPolicy [511]	Sets the commit policy of the replicator.
replicator.extractor.dbms.transaction_frag_size [511]	replicator.extractor.dbms.transaction_frag_size [511]	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.
--replicator-rest-api [511]	replicator-rest-api [511]	Enable (default) or Disable APIv2
--replicator-rest-api-address [511]	replicator-rest-api-address [511]	Address for the API to bind too.
--replicator-rest-api-authentication [511]	replicator-rest-api-authentication [511]	Enforce authentication for the API.
--replicator-rest-api-port [512]	replicator-rest-api-port [512]	Port for the Replicator API.
--replicator-rest-api-ssl [512]	replicator-rest-api-ssl [512]	Enable SSL for the API.
--reset [512]	reset [512]	Clear the current configuration before processing any arguments
--rest-api-admin-pass [512], --rest-api-admin-password [512]	rest-api-admin-pass [512], rest-api-admin-password [512]	Specify the initial Admin User Password for API access. rest-api-admin-password alias only available from version 7.1.2 onwards.
--rest-api-admin-user [512]	rest-api-admin-user [512]	Specify the initial Admin Username for API access
--rmi-port [512], --repl-rmi-port [512]	repl-rmi-port [512], rmi-port [512]	Replication RMI listen port
--rmi-user [512]	rmi-user [512]	The username for RMI authentication
--role [513], --repl-role [513]	repl-role [513], role [513]	What is the replication role for this service?

CmdLine Option	INI File Option	Description
--router-gateway-port [513]	router-gateway-port [513]	The router gateway port
--router-jmx-port [513]	router-jmx-port [513]	The router jmx port
--security-directory [513]	security-directory [513]	Storage directory for the Java security/encryption files
--property=server.listen.address [443]	property=server.listen.address [443]	IP Address to force the connector to be bound too.
--service-alias [513], --dataservice-service-alias [513]	dataservice-service-alias [513], service-alias [513]	Replication alias of this dataservice
--service-name [514]	service-name [514]	Set the service name
--service-type [514], --repl-service-type [514]	repl-service-type [514], service-type [514]	What is the replication service type?
--skip-statemap [514]	skip-statemap [514]	Do not copy the cluster-home/conf/statemap.properties from the previous install
--slaves [514], --dataservice-slaves [514], --members [499]	dataservice-slaves [514], members [499], slaves [514]	What are the Replicas for this dataservice?
--start [514]	start [514]	Start the services after configuration
--start-and-report [514]	start-and-report [514]	Start the services and report out the status after configuration
--svc-allow-any-remote-service [514], --repl-svc-allow-any-remote-service [514]	repl-svc-allow-any-remote-service [514], svc-allow-any-remote-service [514]	Replicate from any service
--svc-applier-block-commit-interval [515], --repl-svc-applier-block-commit-interval [515]	repl-svc-applier-block-commit-interval [515], svc-applier-block-commit-interval [515]	Minimum interval between commits
--svc-applier-block-commit-size [515], --repl-svc-applier-block-commit-size [515]	repl-svc-applier-block-commit-size [515], svc-applier-block-commit-size [515]	Applier block commit size (min 1)
--svc-applier-filters [515], --repl-svc-applier-filters [515]	repl-svc-applier-filters [515], svc-applier-filters [515]	Replication service applier filters
--svc-applier-last-applied-write-interval [515]	svc-applier-last-applied-write-interval [515]	Interval (in seconds) to store the last known applied seqno and latency
--svc-extractor-filters [515], --repl-svc-extractor-filters [515]	repl-svc-extractor-filters [515], svc-extractor-filters [515]	Replication service extractor filters
--svc-fail-on-zero-row-update [515], --repl-svc-fail-on-zero-row-update [515]	repl-svc-fail-on-zero-row-update [515], svc-fail-on-zero-row-update [515]	How should the replicator behave when a Row-Based Replication UPDATE or DELETE does not affect any rows.
--svc-parallelization-type [516], --repl-svc-parallelization-type [516]	repl-svc-parallelization-type [516], svc-parallelization-type [516]	Method for implementing parallel apply
--svc-remote-filters [516], --repl-svc-remote-filters [516]	repl-svc-remote-filters [516], svc-remote-filters [516]	Replication service remote download filters
--svc-reposition-on-source-id-change [516], --repl-svc-reposition-on-source-id-change [516]	repl-svc-reposition-on-source-id-change [516], svc-reposition-on-source-id-change [516]	The Primary will come ONLINE from the current position if the stored source_id does not match the value in the static properties
--svc-shard-default-db [516], --repl-svc-shard-default-db [516]	repl-svc-shard-default-db [516], svc-shard-default-db [516]	Mode for setting the shard ID from the default db
--svc-systemd-config-connector [516]	svc-systemd-config-connector [516]	Used to provide custom systemd configuration that will be used in place of the default generated on.
--svc-systemd-config-manager [517]	svc-systemd-config-manager [517]	Used to provide custom systemd configuration that will be used in place of the default generated on.
--svc-systemd-config-replicator [517]	svc-systemd-config-replicator [517]	Used to provide custom systemd configuration that will be used in place of the default generated on.
--svc-table-engine [518], --repl-svc-table-engine [518]	repl-svc-table-engine [518], svc-table-engine [518]	Replication service table engine

CmdLine Option	INI File Option	Description
--svc-thl-filters [518], --repl-svc-thl-filters [518]	repl-svc-thl-filters [518], svc-thl-filters [518]	Replication service THL filters
--target-dataservice [518], --slave-dataservice [518]	slave-dataservice [518], target-dataservice [518]	Dataservice to use to determine the value of host configuration
--temp-directory [518]	temp-directory [518]	Temporary Directory
--template-file [518]	template-file [518]	Display the keys that may be used in configuration template files
--template-search-path [518]	template-search-path [518]	Adds a new template search path for configuration file generation
--thl-directory [519], --repl-thl-directory [519]	repl-thl-directory [519], thl-directory [519]	Replicator log directory
--thl-do-checksum [519], --repl-thl-do-checksum [519]	repl-thl-do-checksum [519], thl-do-checksum [519]	Execute checksum operations on THL log files
--thl-interface [519], --repl-thl-interface [519]	repl-thl-interface [519], thl-interface [519]	Listen interface to use for THL operations
--thl-log-connection-timeout [519], --repl-thl-log-connection-timeout [519]	repl-thl-log-connection-timeout [519], thl-log-connection-timeout [519]	Number of seconds to wait for a connection to the THL log
--thl-log-file-size [519], --repl-thl-log-file-size [519]	repl-thl-log-file-size [519], thl-log-file-size [519]	File size in bytes for THL disk logs
--thl-log-fsync [520], --repl-thl-log-fsync [520]	repl-thl-log-fsync [520], thl-log-fsync [520]	Fsync THL records on commit. More reliable operation but adds latency to replication when using low-performance storage
--thl-log-retention [520], --repl-thl-log-retention [520]	repl-thl-log-retention [520], thl-log-retention [520]	How long do you want to keep THL files.
--thl-port [520], --repl-thl-port [520]	repl-thl-port [520], thl-port [520]	Port to use for THL Operations
--thl-protocol [520], --repl-thl-protocol [520]	repl-thl-protocol [520], thl-protocol [520]	Protocol to use for THL communication with this service
--topology [520], --dataservice-topology [520]	dataservice-topology [520], topology [520]	Replication topology for the dataservice.
--track-schema-changes [521]	track-schema-changes [521]	This will enable filters that track DDL statements and write the resulting change to files on Replica hosts. The feature is intended for use in some batch deployments.
--user [521]	user [521]	System User
--witnesses [521], --dataservice-witnesses [521]	dataservice-witnesses [521], witnesses [521]	Witness hosts for the dataservice

10.8.1. A tpm Options

--application-password

Option	--application-password [473]
Aliases	--connector-password [473]
Config File Options	application-password [473], connector-password [473]
Description	Database password for the connector
Value Type	string

--application-port

Option	--application-port [473]
Aliases	--connector-listen-port [473]
Config File Options	application-port [473], connector-listen-port [473]
Description	Port for the connector to listen on
Value Type	string
Default	9999

--application-readonly-port

Option	--application-readonly-port [474]
Aliases	--connector-readonly-listen-port [474]
Config File Options	application-readonly-port [474], connector-readonly-listen-port [474]
Description	Port for the connector to listen for read-only connections on
Value Type	string

--application-user

Option	--application-user [474]
Aliases	--connector-user [474]
Config File Options	application-user [474], connector-user [474]
Description	Database username for the connector
Value Type	string

--auto-enable

Option	--auto-enable [474]	
Aliases	--repl-auto-enable [474]	
Config File Options	auto-enable [474], repl-auto-enable [474]	
Description	Auto-enable services after start-up	
Value Type	boolean	
Valid Values	false	
	true	

--auto-recovery-delay-interval

Option	--auto-recovery-delay-interval [474]
Aliases	--repl-auto-recovery-delay-interval [474]
Config File Options	auto-recovery-delay-interval [474], repl-auto-recovery-delay-interval [474]
Description	Delay (in seconds) between going OFFLINE and attempting to go ONLINE
Value Type	numeric
Default	5

The delay between the replicator identifying that autorecovery is needed, and autorecovery being attempted. For busy MySQL installations, larger numbers may be needed to allow time for MySQL servers to restart or recover from their failure.

--auto-recovery-max-attempts

Option	--auto-recovery-max-attempts [474]
Aliases	--repl-auto-recovery-max-attempts [474]
Config File Options	auto-recovery-max-attempts [474], repl-auto-recovery-max-attempts [474]
Description	Maximum number of attempts at automatic recovery
Value Type	numeric
Default	0

Specifies the number of attempts the replicator will make to go back online. When the number of attempts has been reached, the replicator will remain in the OFFLINE 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 [474]
Aliases	--repl-auto-recovery-reset-interval [474]
Config File Options	auto-recovery-reset-interval [474], repl-auto-recovery-reset-interval [474]
Description	Delay (in seconds) before autorecovery is deemed to have succeeded
Value Type	numeric
Default	5

The time in ONLINE state that indicates to the replicator that the autorecovery procedure has succeeded. For servers with very large transactions, this value should be increased to allow the transaction to be successfully applied.

10.8.2. B tpm Options

--backup-directory

Option	--backup-directory [475]
Aliases	--repl-backup-directory [475]
Config File Options	backup-directory [475], repl-backup-directory [475]
Description	Permanent backup storage directory
Value Type	string
Default	{home directory}/backups

--backup-dump-directory

Option	--backup-dump-directory [475]
Aliases	--repl-backup-dump-directory [475]
Config File Options	backup-dump-directory [475], repl-backup-dump-directory [475]
Description	Backup temporary dump directory
Value Type	string

--backup-method

Option	--backup-method [475]	
Aliases	--repl-backup-method [475]	
Config File Options	backup-method [475], repl-backup-method [475]	
Description	Database backup method	
Value Type	string	
Valid Values	ebs-snapshot	
	file-copy-snapshot	
	mariabackup	Use mariabackup (Available from v7.0.0 only)
	mariabackup-incremental	Use mariabackup (Available from v7.0.0 only)
	mysqldump	Use mysqldump
	none	
	script	Use a custom script
	xtrabackup	Use Percona XtraBackup
	xtrabackup-full	Use Percona XtraBackup Full
	xtrabackup-incremental	Use Percona XtraBackup Incremental

The default, if not supplied, will be dependant on the enviroment. During installation tpm will detect which tools are available, favouring xtrabackup-full (or mariabackup-full). If not found, then mysqldump will be the default.

--backup-online

Option	--backup-online [475]

Aliases	--repl-backup-online [475]	
Config File Options	backup-online [475], repl-backup-online [475]	
Description	Does the backup script support backing up a datasource while it is ONLINE	
Value Type	boolean	
Valid Values	false	
	true	

--backup-options

Option	--backup-options [476]
Config File Options	backup-options [476]
Description	Space separated list of options to pass directly to the underlying backup binary in use.
Value Type	string

Options passed are not validated by Tungsten. They are specific to the backup binary you choose to use and therefore you must ensure accuracy in syntax.

--backup-retention

Option	--backup-retention [476]
Aliases	--repl-backup-retention [476]
Config File Options	backup-retention [476], repl-backup-retention [476]
Description	Number of backups to retain
Value Type	numeric

--backup-script

Option	--backup-script [476]
Aliases	--repl-backup-script [476]
Config File Options	backup-script [476], repl-backup-script [476]
Description	What is the path to the backup script
Value Type	filename

--batch-enabled

Option	--batch-enabled [476]	
Config File Options	batch-enabled [476]	
Description	Should the replicator service use a batch applier	
Value Type	boolean	
Default	false	
Valid Values	true	

--batch-load-language

Option	--batch-load-language [476]	
Config File Options	batch-load-language [476]	
Description	Which script language to use for batch loading	
Value Type	string	
Valid Values	js	JavaScript
	sql	SQL

--batch-load-template

Option	--batch-load-template [476]

Config File Options	batch-load-template [476]
Description	Value for the loadBatchTemplate property
Value Type	string

--property=bridgeServerToClientForcedCloseTimeout

Option	--property=bridgeServerToClientForcedCloseTimeout [443]
Config File Options	property=bridgeServerToClientForcedCloseTimeout [443]
Description	Adjust the timeout for Bridge Mode to force close connections (supplied in ms)
Value Type	string
Default	50

--buffer-size

Option	--buffer-size [477]
Aliases	--repl-buffer-size [477]
Config File Options	buffer-size [477], repl-buffer-size [477]
Description	Replicator queue size between stages (min 1)
Value Type	numeric
Default	10

10.8.3. C tpm Options

--cctrl-column-width

Option	--cctrl-column-width [477]
Config File Options	cctrl-column-width [477]
Description	Sets the minimum column width for service names within cctrl

Sets the minimum column width for service names within cctrl

--channels

Option	--channels [477]
Aliases	--repl-channels [477]
Config File Options	channels [477], repl-channels [477]
Description	Number of replication channels to use for parallel apply.
Value Type	numeric
Default	1

--cluster-slave-auto-recovery-delay-interval

Option	--cluster-slave-auto-recovery-delay-interval [477]
Aliases	--cluster-slave-repl-auto-recovery-delay-interval [477]
Config File Options	cluster-slave-auto-recovery-delay-interval [477], cluster-slave-repl-auto-recovery-delay-interval [477]
Description	Default value for --auto-recovery-delay-interval when --topology=cluster-slave
Value Type	string

--cluster-slave-auto-recovery-max-attempts

Option	--cluster-slave-auto-recovery-max-attempts [477]
Aliases	--cluster-slave-repl-auto-recovery-max-attempts [477]
Config File Options	cluster-slave-auto-recovery-max-attempts [477], cluster-slave-repl-auto-recovery-max-attempts [477]
Description	Default value for --auto-recovery-max-attempts when --topology=cluster-slave

Value Type	string

--cluster-slave-auto-recovery-reset-interval

Option	`--cluster-slave-auto-recovery-reset-interval` [478]
Aliases	`--cluster-slave-repl-auto-recovery-reset-interval` [478]
Config File Options	`cluster-slave-auto-recovery-reset-interval` [478], `cluster-slave-repl-auto-recovery-reset-interval` [478]
Description	Default value for --auto-recovery-reset-interval when --topology=cluster-slave
Value Type	string

--composite-datasources

Option	`--composite-datasources` [478]
Aliases	`--dataservice-composite-datasources` [478]
Config File Options	`composite-datasources` [478], `dataservice-composite-datasources` [478]
Description	Data services that should be added to this composite data service
Value Type	string

--config-file

Option	`--config-file` [478]
Config File Options	`config-file` [478]
Description	Display help information for content of the config file
Value Type	string

--conn-java-enable-concurrent-gc

Option	`--conn-java-enable-concurrent-gc` [478]
Config File Options	`conn-java-enable-concurrent-gc` [478]
Description	Connector Java uses concurrent garbage collection
Value Type	string

--conn-java-mem-size

Option	`--conn-java-mem-size` [478]	
Config File Options	`conn-java-mem-size` [478]	
Description	Connector Java heap memory size used to buffer data between clients and databases	
Value Type	numeric	
Valid Values	256	

The Connector allocates memory for each concurrent client connection, and may use up to the size of the configured MySQL `max_allowed_packet`. With multiple connections, the heap size should be configured to at least the combination of the number of concurrent connections multiplied by the maximum packet size.

--conn-round-robin-include-master

Option	`--conn-round-robin-include-master` [478]	
Config File Options	`conn-round-robin-include-master` [478]	
Description	Should the Connector include the Primary in round-robin load balancing	
Value Type	boolean	
Default	false	
Valid Values	false	
	true	

--connector-affinity

Option	--connector-affinity [478]
Config File Options	connector-affinity [478]
Description	The default affinity for all connections
Value Type	string

--connector-allow-cross-site-reconnects-for-reads

Option	--connector-allow-cross-site-reconnects-for-reads [479]	
Config File Options	connector-allow-cross-site-reconnects-for-reads [479]	
Description	Whether or not to reject reconnections that follow a read operation (RO_RELAXED or SmartScale after a read)	
Value Type	boolean	
Default	true	
Valid Values	false	
	true	

--connector-allow-cross-site-reconnects-for-writes

Option	--connector-allow-cross-site-reconnects-for-writes [479]	
Config File Options	connector-allow-cross-site-reconnects-for-writes [479]	
Description	Whether or not to reject reconnections that follow a write operation (RW_STRICT connection or SmartScale after a write)	
Value Type	boolean	
Default	false	
Valid Values	false	
	true	

--connector-autoreconnect

Option	--connector-autoreconnect [479]	
Config File Options	connector-autoreconnect [479]	
Description	Enable auto-reconnect in the connector	
Value Type	boolean	
Default	true	
Valid Values	false	
	true	

--connector-autoreconnect-killed-connections

Option	--connector-autoreconnect-killed-connections [479]	
Config File Options	connector-autoreconnect-killed-connections [479]	
Description	Enable autoreconnect for connections killed within the connector	
Value Type	boolean	
Default	true	
Valid Values	false	
	true	

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 [479]. 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	--connector-bridge-mode [480]	
Aliases	--enable-connector-bridge-mode [480]	
Config File Options	connector-bridge-mode [480], enable-connector-bridge-mode [480]	
Description	Enable the Tungsten Connector bridge mode	
Value Type	boolean	
Default	true	
Valid Values	false	
	true	

--connector-db-version

Option	--connector-db-version [480]
Config File Options	connector-db-version [480]
Description	Server version as advertised by the Connector (proxy-mode only). When not defined or set to autodetect, the version will be set to the same as the primary server. You can override it here, preferably keeping the actual underlying server version with "-tungsten" to indicate client applications that the connection goes through the connector. Note that this option only applies to proxy mode, since bridge mode will always forward the version of the connected node, primary or replica depending on the bridge mode setting.
Value Type	string
Default	autodetect

--connector-default-schema

Option	--connector-default-schema [480]
Aliases	--connector-forced-schema [480]
Config File Options	connector-default-schema [480], connector-forced-schema [480]
Description	Default schema for the connector to use
Value Type	string

--connector-delete-user-map

Option	--connector-delete-user-map [480]
Config File Options	connector-delete-user-map [480]
Description	Overwrite an existing user.map file
Value Type	boolean

--connector-disable-connection-warnings

Option	--connector-disable-connection-warnings [480]
Config File Options	connector-disable-connection-warnings [480]
Description	Hide Connector warnings in log files
Value Type	boolean

--connector-disconnect-timeout

Option	--connector-disconnect-timeout [480]
Config File Options	connector-disconnect-timeout [480]
Description	Time (in seconds) to wait for active connection to disconnect before forcing them closed [default: 5]
Value Type	numeric
Default	5

--connector-drop-after-max-connections

Option	--connector-drop-after-max-connections [480]
Config File Options	connector-drop-after-max-connections [480]
Description	Instantly drop connections that arrive after --connector-max-connections has been reached
Value Type	boolean

--connector-keepalive-timeout

Option	--connector-keepalive-timeout [481]
Config File Options	connector-keepalive-timeout [481]
Description	Delay (in ms) after which a manager connection is considered broken by the connector if no keep-alive command was received. Value must be positive and lower than 300000 (5 min) or the connector will refuse to start. Make sure the manager has mgr-monitor-interval value set greater than this, it controls the frequency at which keep-alive commands are sent.
Value Type	numeric
Default	30000

--connector-listen-interface

Option	--connector-listen-interface [481]
Config File Options	connector-listen-interface [481]
Description	Listen interface to use for the connector
Value Type	string

--connector-manager-use-ssl

Option	--connector-manager-use-ssl [481]	
Config File Options	connector-manager-use-ssl [481]	
Description	Manager<>Connector connections can be encrypted with TLS by setting this to true or with the --disable-security-controls=false flags. Note that when enabled, managers still accept unencrypted connector connections in order to allow upgrades from older versions or unencrypted setups	
Value Type	boolean	
Valid Values	false	Disabled by default in v6 or earlier
	true	Enabled by default in v7 onwards

--connector-max-connections

Option	--connector-max-connections [481]
Config File Options	connector-max-connections [481]
Description	The maximum number of connections the connector should allow at any time
Value Type	numeric

--connector-max-slave-latency

Option	--connector-max-slave-latency [481]
Aliases	--connector-max-applied-latency [481]
Config File Options	connector-max-applied-latency [481], connector-max-slave-latency [481]
Description	The maximum applied latency for replica connections. Disabled by default.
Value Type	numeric
Default	-1

--connector-readonly

Option	--connector-readonly [481]
Aliases	--enable-connector-readonly [481]

Config File Options	connector-readonly [481], enable-connector-readonly [481]
Description	Enable the Tungsten Connector read-only mode
Value Type	boolean

--connector-relative-slave-status

Option	--connector-relative-slave-status [482]
Config File Options	connector-relative-slave-status [482]
Description	Determines whether the reported latency is relative (true) or absolute (false). Default is the value of dataservice-use-relative-latency [485] which itself has the default of false.
Value Type	boolean

--connector-reset-when-affinity-back

Option	--connector-reset-when-affinity-back [482]	
Config File Options	connector-reset-when-affinity-back [482]	
Description	Forces reconnection of clients to datasources with the configured affinity when they become available	
Default	false	
Valid Values	false	Allow connections to remain open even when correct datasource becomes available
	true	Disconnect clients when datasource becomes available

When a site goes offline, connections to this site will be forced closed. Those connections will reconnect, as long as the site stays offline, they will be connected to remote site.

You can now enable an option so that when the site comes back online, the connector will disconnect all these connections that couldn't get to their preferred site so that they will then reconnect to the expected site with the appropriate affinity.

When not enabled, connections will continue to use the server originally configured until they disconnect through normal attribution. This is the default option.

--connector-rest-api

Option	--connector-rest-api [482]	
Config File Options	connector-rest-api [482]	
Description	Enable (default) or Disable APIv2	
Default	true	
Valid Values	false	
	true	

--connector-rest-api-address

Option	--connector-rest-api-address [482]
Config File Options	connector-rest-api-address [482]
Description	Address for the API to bind too.
Default	127.0.0.1

--connector-rest-api-authentication

Option	--connector-rest-api-authentication [482]
Config File Options	connector-rest-api-authentication [482]
Description	Enforce authentication for the API.
Default	true

--connector-rest-api-port

Option	--connector-rest-api-port [482]

Config File Options	connector-rest-api-port [482]
Description	Port for the Connector API.
Default	8096

--connector-rest-api-ssl

Option	--connector-rest-api-ssl [483]
Config File Options	connector-rest-api-ssl [483]
Description	Enable SSL for the API.
Default	true

--connector-ro-addresses

Option	--connector-ro-addresses [483]
Config File Options	connector-ro-addresses [483]
Description	Connector addresses that should receive a r/o connection
Value Type	string

--connector-rw-addresses

Option	--connector-rw-addresses [483]
Config File Options	connector-rw-addresses [483]
Description	Connector addresses that should receive a r/w connection
Value Type	string

--connector-rwsplitting

Option	--connector-rwsplitting [483]
Config File Options	connector-rwsplitting [483]
Description	Enable DirectReads R/W splitting in the connector
Value Type	string

--connector-server-ssl-ciphers

Option	--connector-server-ssl-ciphers [483]
Config File Options	connector-server-ssl-ciphers [483]
Description	Configures ciphers the connector uses for SSL communications to the dbatabase server. Defaults to allow all cipher suites supported by the running JVM.
Value Type	string

--connector-server-ssl-protocols

Option	--connector-server-ssl-protocols [483]
Config File Options	connector-server-ssl-protocols [483]
Description	Configures protocols the connector uses for SSL communications to the dbatabase server. Default value when not specified will be TLSv1,TLSv1.1,TLSv1.2
Value Type	string
Default	TLSv1,TLSv1.1,TLSv1.2

--connector-smartscale

Option	--connector-smartscale [483]
Config File Options	connector-smartscale [483]
Description	Enable SmartScale R/W splitting in the connector
Value Type	boolean

Default	false	
Valid Values	false	
	true	

--connector-smartscale-sessionid

Option	`--connector-smartscale-sessionid` [484]	
Config File Options	`connector-smartscale-sessionid` [484]	
Description	The default session ID to use with smart scale	
Value Type	string	
Default	DATABASE	
Valid Values	CONNECTION	
	DATABASE	
	PROVIDED_IN_DBNAME	
	USER	

--connector-ssl-capable

Option	`--connector-ssl-capable` [484]	
Config File Options	`connector-ssl-capable` [484]	
Description	Defines whether connector ports are advertised as being capable of SSL	
Value Type	boolean	
Default	true	
Valid Values	false	Do not advertise connector ports as SSL capable
	true	Advertise connector ports as SSL capable

When SSL is enabled, the Connector automatically advertises the ports and itself as SSL capable. With some clients, this triggers them to use SSL even if SSL has not been configured. This causes the connections to fail and not operate correctly.

--connectors

Option	`--connectors` [484]
Aliases	`--dataservice-connectors` [484]
Config File Options	`connectors` [484], `dataservice-connectors` [484]
Description	Hostnames for the dataservice connectors
Value Type	string

--consistency-policy

Option	`--consistency-policy` [484]
Aliases	`--repl-consistency-policy` [484]
Config File Options	`consistency-policy` [484], `repl-consistency-policy` [484]
Description	Should the replicator stop or warn if a consistency check fails?
Value Type	string

10.8.4. D tpm Options

--dataservice-name

Option	`--dataservice-name` [484]
Config File Options	`dataservice-name` [484]
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 [485]
Config File Options	dataservice-relay-enabled [485]
Description	Make this dataservice a replica of another
Value Type	string

--dataservice-schema

Option	--dataservice-schema [485]
Config File Options	dataservice-schema [485]
Description	The db schema to hold dataservice details
Value Type	string

--dataservice-thl-port

Option	--dataservice-thl-port [485]
Config File Options	dataservice-thl-port [485]
Description	Port to use for THL operations
Value Type	numeric
Default	2112

--dataservice-use-relative-latency

Option	--dataservice-use-relative-latency [485]
Aliases	--use-relative-latency [485]
Config File Options	dataservice-use-relative-latency [485], use-relative-latency [485]
Description	Enable the cluster to operate on relative latency
Value Type	boolean

--dataservice-vip-enabled

Option	--dataservice-vip-enabled [485]
Config File Options	dataservice-vip-enabled [485]
Description	Is VIP management enabled?
Value Type	boolean

--dataservice-vip-ipaddress

Option	--dataservice-vip-ipaddress [485]
Config File Options	dataservice-vip-ipaddress [485]
Description	VIP IP address
Value Type	string

--dataservice-vip-netmask

Option	--dataservice-vip-netmask [485]
Config File Options	dataservice-vip-netmask [485]
Description	VIP netmask
Value Type	string

--datasource-boot-script

Option	--datasource-boot-script [485]
Aliases	--repl-datasource-boot-script [485]
Config File Options	datasource-boot-script [485], repl-datasource-boot-script [485]
Description	Database start script
Value Type	string

--datasource-enable-ssl

Option	--datasource-enable-ssl [486]
Aliases	--repl-datasource-enable-ssl [486]
Config File Options	datasource-enable-ssl [486], repl-datasource-enable-ssl [486]
Description	Enable SSL connection to DBMS server
Value Type	boolean

--datasource-log-directory

Option	--datasource-log-directory [486]
Aliases	--repl-datasource-log-directory [486]
Config File Options	datasource-log-directory [486], repl-datasource-log-directory [486]
Description	Primary log directory
Value Type	string

--datasource-log-pattern

Option	--datasource-log-pattern [486]
Aliases	--repl-datasource-log-pattern [486]
Config File Options	datasource-log-pattern [486], repl-datasource-log-pattern [486]
Description	Primary log filename pattern
Value Type	string

--datasource-mysql-conf

Option	--datasource-mysql-conf [486]
Aliases	--repl-datasource-mysql-conf [486]
Config File Options	datasource-mysql-conf [486], repl-datasource-mysql-conf [486]
Description	MySQL config file
Value Type	string

--datasource-mysql-data-directory

Option	--datasource-mysql-data-directory [486]
Aliases	--repl-datasource-mysql-data-directory [486]
Config File Options	datasource-mysql-data-directory [486], repl-datasource-mysql-data-directory [486]
Description	MySQL data directory
Value Type	string

--datasource-mysql-ibdata-directory

Option	--datasource-mysql-ibdata-directory [486]
Aliases	--repl-datasource-mysql-ibdata-directory [486]
Config File Options	datasource-mysql-ibdata-directory [486], repl-datasource-mysql-ibdata-directory [486]
Description	MySQL InnoDB data directory
Value Type	string

--datasource-mysql-iblog-directory

Option	--datasource-mysql-iblog-directory [487]
Aliases	--repl-datasource-mysql-iblog-directory [487]
Config File Options	datasource-mysql-iblog-directory [487], repl-datasource-mysql-iblog-directory [487]
Description	MySQL InnoDB log directory
Value Type	string

--datasource-mysql-ssl-ca

Option	--datasource-mysql-ssl-ca [487]
Aliases	--repl-datasource-mysql-ssl-ca [487]
Config File Options	datasource-mysql-ssl-ca [487], repl-datasource-mysql-ssl-ca [487]
Description	MySQL SSL CA file
Value Type	string

--datasource-mysql-ssl-cert

Option	--datasource-mysql-ssl-cert [487]
Aliases	--repl-datasource-mysql-ssl-cert [487]
Config File Options	datasource-mysql-ssl-cert [487], repl-datasource-mysql-ssl-cert [487]
Description	MySQL SSL certificate file
Value Type	string

--datasource-mysql-ssl-key

Option	--datasource-mysql-ssl-key [487]
Aliases	--repl-datasource-mysql-ssl-key [487]
Config File Options	datasource-mysql-ssl-key [487], repl-datasource-mysql-ssl-key [487]
Description	MySQL SSL key file
Value Type	string

--datasource-systemctl-service

Option	--datasource-systemctl-service [487]
Aliases	--repl-datasource-systemctl-service [487]
Config File Options	datasource-systemctl-service [487], repl-datasource-systemctl-service [487]
Description	Database systemctl script
Value Type	string

Specifies the command name or full path of the command that should be used to control the database service, including startup, shutdown and restart. This is used by the Tungsten to control the underlying database service. By default, this will be configured to the service according to your environment if it has been found during installation. For example, the services command, or /etc/init.d/mysql.

--datasource-type

Option	--datasource-type [487]	
Aliases	--repl-datasource-type [487]	
Config File Options	datasource-type [487], repl-datasource-type [487]	
Description	Database type	
Value Type	string	
Default	mysql	
Valid Values	file	File
	hdfs	HDFS (Hadoop)

	kafka	Kafka
	mongodb	MongoDB
	mysql	MySQL
	oracle	Oracle
	postgres	PostgreSQL
	vertica	Vertica

--delete

Option	`--delete` [488]
Config File Options	`delete` [488]
Description	Delete the named data service from the configuration Data Service options
Value Type	string

--deploy-current-package

Option	`--deploy-current-package` [488]
Config File Options	`deploy-current-package` [488]
Description	Deploy the current Tungsten package
Value Type	string

--deploy-package-uri

Option	`--deploy-package-uri` [488]
Config File Options	`deploy-package-uri` [488]
Description	URL for the Tungsten package to deploy
Value Type	string

--direct-datasource-log-directory

Option	`--direct-datasource-log-directory` [488]
Aliases	`--repl-direct-datasource-log-directory` [488]
Config File Options	`direct-datasource-log-directory` [488], `repl-direct-datasource-log-directory` [488]
Description	Primary log directory
Value Type	string

--direct-datasource-log-pattern

Option	`--direct-datasource-log-pattern` [488]
Aliases	`--repl-direct-datasource-log-pattern` [488]
Config File Options	`direct-datasource-log-pattern` [488], `repl-direct-datasource-log-pattern` [488]
Description	Primary log filename pattern
Value Type	string

--direct-datasource-type

Option	`--direct-datasource-type` [488]	
Aliases	`--repl-direct-datasource-type` [488]	
Config File Options	`direct-datasource-type` [488], `repl-direct-datasource-type` [488]	
Description	Database type	
Value Type	string	
Default	mysql	
Valid Values	file	File

	hdfs	HDFS (Hadoop)
	mongodb	MongoDB
	mysql	
	mysql	MySQL
	oracle	Oracle
	vertica	Vertica

--direct-replication-host

Option	`--direct-replication-host` [489]
Aliases	`--direct-datasource-host` [489], `--repl-direct-datasource-host` [489]
Config File Options	`direct-datasource-host` [489], `direct-replication-host` [489], `repl-direct-datasource-host` [489]
Description	Database server hostname
Value Type	string

--direct-replication-password

Option	`--direct-replication-password` [489]
Aliases	`--direct-datasource-password` [489], `--repl-direct-datasource-password` [489]
Config File Options	`direct-datasource-password` [489], `direct-replication-password` [489], `repl-direct-datasource-password` [489]
Description	Password for datasource connection
Value Type	string

--direct-replication-port

Option	`--direct-replication-port` [489]
Aliases	`--direct-datasource-port` [489], `--repl-direct-datasource-port` [489]
Config File Options	`direct-datasource-port` [489], `direct-replication-port` [489], `repl-direct-datasource-port` [489]
Description	Database server port
Value Type	string

--direct-replication-user

Option	`--direct-replication-user` [489]
Aliases	`--direct-datasource-user` [489], `--repl-direct-datasource-user` [489]
Config File Options	`direct-datasource-user` [489], `direct-replication-user` [489], `repl-direct-datasource-user` [489]
Description	Database login for Tungsten
Value Type	string

--directory

Option	`--directory` [489]
Config File Options	`directory` [489]
Description	Set the directory of an existing installation used during fetching an existing configuration

Set the directory of an existing installation used during fetching an existing configuration

--disable-relay-logs

Option	`--disable-relay-logs` [489]
Aliases	`--repl-disable-relay-logs` [489]
Config File Options	`disable-relay-logs` [489], `repl-disable-relay-logs` [489]
Description	Disable the use of relay-logs?
Value Type	boolean

Default	true	
Valid Values	false	
	true	

--disable-security-controls

Option	`--disable-security-controls` [490]	
Config File Options	`disable-security-controls` [490]	
Description	Disables all forms of security, including SSL, TLS and authentication	
Value Type	string	
Default	false	
Valid Values	true	Disables full security

--disable-slave-extractor

Option	`--disable-slave-extractor` [490]	
Aliases	`--repl-disable-slave-extractor` [490]	
Config File Options	`disable-slave-extractor` [490], `repl-disable-slave-extractor` [490]	
Description	Should replica servers support the primary role?	
Value Type	string	

--drop-static-columns-in-updates

Option	`--drop-static-columns-in-updates` [490]	
Config File Options	`drop-static-columns-in-updates` [490]	
Description	This will modify UPDATE transactions in row-based replication and eliminate any columns that were not modified.	
Value Type	boolean	
Default	false	
Valid Values	false	
	true	

10.8.5. E tpm Options

--enable-active-witnesses

Option	`--enable-active-witnesses` [490]	
Aliases	`--active-witnesses` [490]	
Config File Options	`active-witnesses` [490], `enable-active-witnesses` [490]	
Description	Enable active witness hosts	
Value Type	boolean	
Default	false	
Valid Values	false	
	true	

--enable-connector-client-ssl

Option	`--enable-connector-client-ssl` [490]	
Aliases	`--connector-client-ssl` [490]	
Config File Options	`connector-client-ssl` [490], `enable-connector-client-ssl` [490]	
Description	Enable SSL encryption of traffic from the client to the connector	
Value Type	boolean	

--enable-connector-server-ssl

Option	--enable-connector-server-ssl [490]
Aliases	--connector-server-ssl [490]
Config File Options	connector-server-ssl [490], enable-connector-server-ssl [490]
Description	Enable SSL encryption of traffic from the connector to the database
Value Type	boolean

--enable-connector-ssl

Option	--enable-connector-ssl [491]
Aliases	--connector-ssl [491]
Config File Options	connector-ssl [491], enable-connector-ssl [491]
Description	Enable SSL encryption of connector traffic to the database
Value Type	boolean

--enable-heterogeneous-master

Option	--enable-heterogeneous-master [491]	
Config File Options	enable-heterogeneous-master [491]	
Description	Enable heterogeneous operation for the primary	
Value Type	boolean	
Default	false	
Valid Values	false	
	true	

--enable-heterogeneous-service

Option	--enable-heterogeneous-service [491]	
Config File Options	enable-heterogeneous-service [491]	
Description	Enable heterogeneous operation	
Value Type	boolean	
Default	false	
Valid Values	false	
	true	

- On a Primary
  - mysql-use-bytes-for-string [504] is set to false.
  - colnames filter is enabled (in the binlog-to-q stage to add column names to the THL information.
  - pkey filter is enabled (in the binlog-to-q and q-to-dbms stage), with the *addPkeyToInserts* and *addColumnsToDeletes* filter options set to false.
  - enumtostring filter is enabled (in the q-to-thl stage), to translate ENUM values to their string equivalents.
  - settostring filter is enabled (in the q-to-thl stage), to translate SET values to their string equivalents.
- On a Replica
  - mysql-use-bytes-for-string [504] is set to true.
  - pkey filter is enabled (q-to-dbms stage).

--enable-heterogeneous-slave

Option	--enable-heterogeneous-slave [491]
Config File Options	enable-heterogeneous-slave [491]
Description	Enable heterogeneous operation for the replica

Value Type	boolean	
Default	false	
Valid Values	false	
	true	

--enable-jgroups-ssl

Option	`--enable-jgroups-ssl` [492]
Aliases	`--jgroups-ssl` [492]
Config File Options	`enable-jgroups-ssl` [492], `jgroups-ssl` [492]
Description	Enable SSL encryption of JGroups communication on this host
Value Type	boolean

--enable-rmi-authentication

Option	`--enable-rmi-authentication` [492]
Aliases	`--rmi-authentication` [492]
Config File Options	`enable-rmi-authentication` [492], `rmi-authentication` [492]
Description	Enable RMI authentication for the services running on this host
Value Type	boolean

--enable-rmi-ssl

Option	`--enable-rmi-ssl` [492]
Aliases	`--rmi-ssl` [492]
Config File Options	`enable-rmi-ssl` [492], `rmi-ssl` [492]
Description	Enable SSL encryption of RMI communication on this host
Value Type	boolean

--enable-slave-thl-listener

Option	`--enable-slave-thl-listener` [492]
Aliases	`--repl-enable-slave-thl-listener` [492]
Config File Options	`enable-slave-thl-listener` [492], `repl-enable-slave-thl-listener` [492]
Description	Should this service allow THL connections?
Value Type	boolean

--enable-sudo-access

Option	`--enable-sudo-access` [492]	
Aliases	`--root-command-prefix` [492]	
Config File Options	`enable-sudo-access` [492], `root-command-prefix` [492]	
Description	Run root commands using sudo	
Value Type	boolean	
Valid Values	false	
	true	

The default behavior of this property is different in certain installs.

For a cluster node install that INCLUDES the manager process, AND where the install OS user is NOT root, the default will be `true`

For a Replicator only or Connector only install, the default will be `false`

When set to true, the property has the following effect:

- During staging tpm installs, if the tungsten user is different from the ssh user on remote hosts

- All startup scripts when using systemctl: replicator, connector, manager will call systemctl prefixed with sudo: for example: `sudo -n systemctl start treplicator`
- tprovision script (tps.pl) requires sudo access for mysql and xtrabackup calls
- replicator backup script for xtrabackup and other backup utilities
- check_tungsten.sh utility to call xinetd
- tmonitor starts exporter service with sudo
- manager to restart mysql service when found stopped
- tpm diagnostic operation (tpm diag)

--enable-thl-ssl

Option	`--enable-thl-ssl` [493]
Aliases	`--repl-enable-thl-ssl` [493], `--thl-ssl` [493]
Config File Options	`enable-thl-ssl` [493], `repl-enable-thl-ssl` [493], `thl-ssl` [493]
Description	Enable SSL encryption of THL communication for this service
Value Type	boolean

--executable-prefix

Option	`--executable-prefix` [493]
Config File Options	`executable-prefix` [493]
Description	Adds a prefix to command aliases
Value Type	string

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

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

10.8.6. F tpm Options

--file-protection-level

Option	`--file-protection-level` [493]
Config File Options	`file-protection-level` [493]
Description	Protection level for Continuent files
Value Type	string

--file-protection-umask

Option	`--file-protection-umask` [493]
Config File Options	`file-protection-umask` [493]
Description	Protection umask for Continuent files
Value Type	string

10.8.7. H tpm Options

--host-name

Option	`--host-name` [493]
Config File Options	`host-name` [493]
Description	DNS hostname

Value Type	string

--hosts

Option	`--hosts` [494]
Config File Options	`hosts` [494]
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` [494]
Aliases	`--dataservice-hub-host` [494]
Config File Options	`dataservice-hub-host` [494], `hub` [494]
Description	What is the hub host for this all-masters dataservice?
Value Type	string

--hub-service

Option	`--hub-service` [494]
Aliases	`--dataservice-hub-service` [494]
Config File Options	`dataservice-hub-service` [494], `hub-service` [494]
Description	The data service to use for the hub of a star topology
Value Type	string

10.8.8. I tpm Options

--install

Option	`--install` [494]
Config File Options	`install` [494]
Description	Install service start scripts
Value Type	string

--install-directory

Option	`--install-directory` [494]
Aliases	`--home-directory` [494]
Config File Options	`home-directory` [494], `install-directory` [494]
Description	Installation directory
Value Type	string

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

10.8.9. J tpm Options

--java-connector-keystore-password

Option	`--java-connector-keystore-password` [494]
Config File Options	`java-connector-keystore-password` [494]
Description	Set the password for unlocking the tungsten_connector_keystore.jks file in the security directory. Specific to connector<->mysql communication only.
Value Type	string

--java-connector-keystore-path

Option	--java-connector-keystore-path [494]
Config File Options	java-connector-keystore-path [494]
Description	Local path to the Java Connector Keystore file. Specific to connector<->mysql communication only.
Value Type	filename

--java-connector-truststore-password

Option	--java-connector-truststore-password [495]
Config File Options	java-connector-truststore-password [495]
Description	Set the password for unlocking the tungsten_connector_truststore.jks file in the security directory. Specific to connector<->mysql communication only.
Value Type	string

--java-connector-truststore-path

Option	--java-connector-truststore-path [495]
Config File Options	java-connector-truststore-path [495]
Description	Local path to the Java Connector Truststore file. Specific to connector<->mysql communication only.
Value Type	filename

--java-enable-concurrent-gc

Option	--java-enable-concurrent-gc [495]
Aliases	--repl-java-enable-concurrent-gc [495]
Config File Options	java-enable-concurrent-gc [495], repl-java-enable-concurrent-gc [495]
Description	Replicator Java uses concurrent garbage collection
Value Type	boolean

--java-external-lib-dir

Option	--java-external-lib-dir [495]
Aliases	--repl-java-external-lib-dir [495]
Config File Options	java-external-lib-dir [495], repl-java-external-lib-dir [495]
Description	Directory for 3rd party Jar files required by replicator
Value Type	string

--java-file-encoding

Option	--java-file-encoding [495]
Aliases	--repl-java-file-encoding [495]
Config File Options	java-file-encoding [495], repl-java-file-encoding [495]
Description	Java platform charset (esp. for heterogeneous replication)
Value Type	string

--java-jgroups-key

Option	--java-jgroups-key [495]
Config File Options	java-jgroups-key [495]
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 [495]

Config File Options	java-jgroups-keystore-path [495]
Description	Local path to the JGroups Java Keystore file.
Value Type	filename

--java-jmxremote-access-path

Option	--java-jmxremote-access-path [496]
Config File Options	java-jmxremote-access-path [496]
Description	Local path to the Java JMX Remote Access file.
Value Type	filename

--java-keystore-password

Option	--java-keystore-password [496]
Config File Options	java-keystore-password [496]
Description	Set the password for unlocking the tungsten_keystore.jks file in the security directory. Specific for intra cluster communication.
Value Type	string

--java-keystore-path

Option	--java-keystore-path [496]
Config File Options	java-keystore-path [496]
Description	Local path to the Java Keystore file. Specific for intra cluster communication. NOTE: When java-keystore-path is passed to tpm, the keystore must contain both tls and mysql certs when appropriate. tpm will NOT add mysql cert nor generate tls cert when this flag is found, so both certs must be manually imported already.
Value Type	filename

--java-mem-size

Option	--java-mem-size [496]
Aliases	--repl-java-mem-size [496]
Config File Options	java-mem-size [496], repl-java-mem-size [496]
Description	Replicator Java heap memory size in Mb (min 128)
Value Type	numeric

--java-passwordstore-path

Option	--java-passwordstore-path [496]
Config File Options	java-passwordstore-path [496]
Description	Local path to the Java Password Store file.
Value Type	filename

--java-thl-keystore-path

Option	--java-thl-keystore-path [496]
Config File Options	java-thl-keystore-path [496]
Description	The alias to use for storing the keystore associated with THL encryption.
Value Type	string

--java-thl-truststore-path

Option	--java-thl-truststore-path [496]
Config File Options	java-thl-truststore-path [496]
Description	The alias to use for storing the truststore associated with THL encryption.

Value Type	string

--java-tls-alias

Option	`--java-tls-alias` [497]
Config File Options	`java-tls-alias` [497]
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` [497]
Config File Options	`java-tls-key-lifetime` [497]
Description	Lifetime for the Java TLS key
Value Type	numeric

--java-tls-keystore-path

Option	`--java-tls-keystore-path` [497]
Config File Options	`java-tls-keystore-path` [497]
Description	The keystore holding a certificate to use for all Continuent TLS encryption.
Value Type	string

--java-truststore-password

Option	`--java-truststore-password` [497]
Config File Options	`java-truststore-password` [497]
Description	The password for unlocking the tungsten_truststore.jks file in the security directory
Value Type	string

--java-truststore-path

Option	`--java-truststore-path` [497]
Config File Options	`java-truststore-path` [497]
Description	Local path to the Java Truststore file.
Value Type	filename

--java-user-timezone

Option	`--java-user-timezone` [497]
Aliases	`--repl-java-user-timezone` [497]
Config File Options	`java-user-timezone` [497], `repl-java-user-timezone` [497]
Description	Java VM Timezone (esp. for cross-site replication)
Value Type	numeric

10.8.10. L tpm Options

--log

Option	`--log` [497]
Config File Options	`log` [497]
Description	Write all messages, visible and hidden, to this file. You may specify a filename, 'pid' or 'timestamp'.
Value Type	numeric

--log-slave-updates

Option	--log-slave-updates [497]	
Config File Options	log-slave-updates [497]	
Description	Should replicas log updates to binlog	
Value Type	boolean	
Default	false	
Valid Values	false	
	true	

10.8.11. M tpm Options

--manager-replicator-offline-timeout

Option	--manager-replicator-offline-timeout [498]
Config File Options	manager-replicator-offline-timeout [498]
Description	Timeout (in seconds) for the manager to wait until the replicator goes offline.
Value Type	string
Default	180

--manager-rest-api

Option	--manager-rest-api [498]	
Config File Options	manager-rest-api [498]	
Description	Enable (default) or Disable APIv2	
Default	true	
Valid Values	false	
	true	

--manager-rest-api-address

Option	--manager-rest-api-address [498]
Config File Options	manager-rest-api-address [498]
Description	Address for the API to bind too.
Default	127.0.0.1

--manager-rest-api-authentication

Option	--manager-rest-api-authentication [498]
Config File Options	manager-rest-api-authentication [498]
Description	Enforce authentication for the API.
Default	true

--manager-rest-api-port

Option	--manager-rest-api-port [498]
Config File Options	manager-rest-api-port [498]
Description	Port for the manager API.
Default	8090

--manager-rest-api-ssl

Option	--manager-rest-api-ssl [498]
Config File Options	manager-rest-api-ssl [498]
Description	Enable SSL for the API.

Default	true

--master

Option	`--master` [499]
Aliases	`--dataservice-master-host` [499], `--masters` [499], `--relay` [499]
Config File Options	`dataservice-master-host` [499], `master` [499], `masters` [499], `relay` [499]
Description	Hostname of the primary (or relay) host within this service
Value Type	string

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

--master-preferred-role

Option	`--master-preferred-role` [499]	
Aliases	`--repl-master-preferred-role` [499]	
Config File Options	`master-preferred-role` [499], `repl-master-preferred-role` [499]	
Description	Preferred role for primary THL when connecting as a replica	
Value Type	string	
Valid Values	master	Primary role
	slave	Replica role

--master-services

Option	`--master-services` [499]
Aliases	`--dataservice-master-services` [499]
Config File Options	`dataservice-master-services` [499], `master-services` [499]
Description	Data service names that should be used on each Primary
Value Type	string

--master-thl-host

Option	`--master-thl-host` [499]
Config File Options	`master-thl-host` [499]
Description	Primary THL Hostname
Value Type	string

--master-thl-port

Option	`--master-thl-port` [499]
Config File Options	`master-thl-port` [499]
Description	Primary THL Port
Value Type	string

--members

Option	`--members` [499]
Aliases	`--dataservice-hosts` [499]
Config File Options	`dataservice-hosts` [499], `members` [499]
Description	Hostnames for the dataservice members
Value Type	string

--metadata-directory

Option	`--metadata-directory` [499]

Aliases	--repl-metadata-directory [499]
Config File Options	metadata-directory [499], repl-metadata-directory [499]
Description	Replicator metadata directory
Value Type	string

--mgr-api

Option	--mgr-api [500]
Config File Options	mgr-api [500]
Description	Enable the Manager API
Value Type	boolean

--mgr-api-address

Option	--mgr-api-address [500]
Config File Options	mgr-api-address [500]
Description	Address for the Manager API
Value Type	string

--mgr-api-full-access

Option	--mgr-api-full-access [500]
Config File Options	mgr-api-full-access [500]
Description	Enable all Manager API commands. Only the status command will be enabled without it.
Value Type	boolean

--mgr-api-port

Option	--mgr-api-port [500]
Config File Options	mgr-api-port [500]
Description	Port for the Manager API
Value Type	string

--mgr-group-communication-port

Option	--mgr-group-communication-port [500]
Config File Options	mgr-group-communication-port [500]
Description	Port to use for manager group communication
Value Type	numeric
Default	7800

--mgr-heap-threshold

Option	--mgr-heap-threshold [500]
Config File Options	mgr-heap-threshold [500]
Description	Java memory usage (MB) that will force a Manager restart
Value Type	numeric
Default	200

--mgr-java-enable-concurrent-gc

Option	--mgr-java-enable-concurrent-gc [500]
Config File Options	mgr-java-enable-concurrent-gc [500]
Description	Manager Java uses concurrent garbage collection

Value Type	string

--mgr-java-mem-size

Option	`--mgr-java-mem-size` [501]
Config File Options	`mgr-java-mem-size` [501]
Description	Manager Java heap memory size in Mb (min 128)
Value Type	numeric
Default	250

--mgr-listen-interface

Option	`--mgr-listen-interface` [501]
Config File Options	`mgr-listen-interface` [501]
Description	Listen interface to use for the manager
Value Type	string

--mgr-monitor-interval

Option	`--mgr-monitor-interval` [501]
Config File Options	`mgr-monitor-interval` [501]
Description	Frequency, in milliseconds, at which managers send keep-alive commands to connectors.
Value Type	numeric
Default	3000

--mgr-ping-method

Option	`--mgr-ping-method` [501]
Config File Options	`mgr-ping-method` [501]
Description	Mechanism to use when identifying the liveness of other datasources (ping, echo)
Value Type	string

--mgr-policy-mode

Option	`--mgr-policy-mode` [501]	
Config File Options	`mgr-policy-mode` [501]	
Description	Manager policy mode	
Value Type	string	
Valid Values	automatic	*`automatic`* policy mode
	maintenance	*`maintenance`* policy mode
	manual	*`manual`* policy mode

--mgr-rmi-port

Option	`--mgr-rmi-port` [501]
Config File Options	`mgr-rmi-port` [501]
Description	Port to use for the manager RMI server
Value Type	string

--mgr-rmi-remote-port

Option	`--mgr-rmi-remote-port` [501]
Config File Options	`mgr-rmi-remote-port` [501]
Description	Port to use for calling the remote manager RMI server

Value Type	string

--mgr-ro-slave

Option	--mgr-ro-slave [502]
Config File Options	mgr-ro-slave [502]
Description	Make replicas read-only
Value Type	boolean

--mgr-vip-arp-path

Option	--mgr-vip-arp-path [502]
Config File Options	mgr-vip-arp-path [502]
Description	Path to the arp binary
Value Type	filename

--mgr-vip-device

Option	--mgr-vip-device [502]
Config File Options	mgr-vip-device [502]
Description	VIP network device
Value Type	string

--mgr-vip-ifconfig-path

Option	--mgr-vip-ifconfig-path [502]
Config File Options	mgr-vip-ifconfig-path [502]
Description	Path to the ifconfig binary
Value Type	filename

--mgr-wait-for-members

Option	--mgr-wait-for-members [502]
Config File Options	mgr-wait-for-members [502]
Description	Wait for all datasources to be available before completing installation
Value Type	boolean

--mon-db-query-timeout

Option	--mon-db-query-timeout [502]
Config File Options	mon-db-query-timeout [502]
Description	Defines the timeout (in secs) for the managers mysql_checker_query.sql call
Value Type	numeric
Default	5

--mysql-allow-intensive-checks

Option	--mysql-allow-intensive-checks [502]	
Config File Options	mysql-allow-intensive-checks [502]	
Description	For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility.	
Value Type	boolean	
Default	false	
Valid Values	false	
	true	

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

--mysql-driver

Option	`--mysql-driver` [503]
Config File Options	`mysql-driver` [503]
Description	MySQL Driver Vendor
Value Type	string
Default	drizzle

--mysql-enable-ansiquotes

Option	`--mysql-enable-ansiquotes` [503]
Aliases	`--repl-mysql-enable-ansiquotes` [503]
Config File Options	`mysql-enable-ansiquotes` [503], `repl-mysql-enable-ansiquotes` [503]
Description	Enables ANSI_QUOTES mode for incoming events?
Value Type	boolean

--mysql-enable-enumtostring

Option	`--mysql-enable-enumtostring` [503]
Aliases	`--repl-mysql-enable-enumtostring` [503]
Config File Options	`mysql-enable-enumtostring` [503], `repl-mysql-enable-enumtostring` [503]
Description	Enable a filter to convert ENUM values to strings
Value Type	boolean

--mysql-enable-noonlykeywords

Option	`--mysql-enable-noonlykeywords` [503]
Aliases	`--repl-mysql-enable-noonlykeywords` [503]
Config File Options	`mysql-enable-noonlykeywords` [503], `repl-mysql-enable-noonlykeywords` [503]
Description	Enables a filter to translate `DELETE FROM ONLY` to `DELETE FROM` and `UPDATE ONLY` to `UPDATE`.
Value Type	boolean

--mysql-enable-settostring

Option	`--mysql-enable-settostring` [503]
Aliases	`--repl-mysql-enable-settostring` [503]
Config File Options	`mysql-enable-settostring` [503], `repl-mysql-enable-settostring` [503]
Description	Enable a filter to convert SET types to strings
Value Type	boolean

--mysql-ro-slave

Option	`--mysql-ro-slave` [503]
Aliases	`--repl-mysql-ro-slave` [503]
Config File Options	`mysql-ro-slave` [503], `repl-mysql-ro-slave` [503]
Description	Replicas are read-only?
Value Type	boolean

--mysql-server-id

Option	`--mysql-server-id` [503]
Aliases	`--repl-mysql-server-id` [503]

Config File Options	mysql-server-id [503], repl-mysql-server-id [503]
Description	Explicitly set the MySQL server ID
Value Type	numeric

Setting this option explicitly sets the server-id information normally located in the MySQL configuration (my.cnf). This is useful in situations where there may be multiple MySQL installations and the server ID needs to be identified to prevent collisions when reading from the same Primary.

--mysql-use-bytes-for-string

Option	--mysql-use-bytes-for-string [504]
Aliases	--repl-mysql-use-bytes-for-string [504]
Config File Options	mysql-use-bytes-for-string [504], repl-mysql-use-bytes-for-string [504]
Description	Transfer strings as their byte representation?
Value Type	boolean

--mysql-xtrabackup-dir

Option	--mysql-xtrabackup-dir [504]
Aliases	--repl-mysql-xtrabackup-dir [504]
Config File Options	mysql-xtrabackup-dir [504], repl-mysql-xtrabackup-dir [504]
Description	Directory to use for storing xtrabackup full & incremental backups
Value Type	string

10.8.12. N tpm Options

--native-slave-takeover

Option	--native-slave-takeover [504]
Aliases	--repl-native-slave-takeover [504]
Config File Options	native-slave-takeover [504], repl-native-slave-takeover [504]
Description	Takeover native replication
Value Type	boolean

--no-connectors

Option	--no-connectors [504]
Config File Options	no-connectors [504]
Description	When issued during an update, connectors will not be restarted. Restart of the connectors will then need to be performed mannually for updates to take affect.
Value Type	string

--no-deployment

Option	--no-deployment [504]
Config File Options	no-deployment [504]
Description	Skip deployment steps that create the install directory
Value Type	string

--no-validation

Option	--no-validation [504]
Config File Options	no-validation [504]
Description	Skip validation checks that run on each host
Value Type	string

10.8.13. O tpm Options

--optimize-row-events

Option	--optimize-row-events [505]	
Config File Options	optimize-row-events [505]	
Description	Enables or disables optimized row updates. Enabled by default.	
Value Type	boolean	
Default	true	
Valid Values	false	Disable

Bundles multiple row-based events into a single INSERT or DELETE statement. This increases the throughput of large batches of row-based events.

--optimize-row-events-limit-delete-rows

Option	--optimize-row-events-limit-delete-rows [505]
Config File Options	optimize-row-events-limit-delete-rows [505]
Description	Limits the number of deletes grouped per event when optimize-row-events is enabled. Note that deletes can only be optimized for tables with single column PK's
Value Type	boolean
Default	100

--optimize-row-events-limit-insert-rows

Option	--optimize-row-events-limit-insert-rows [505]
Config File Options	optimize-row-events-limit-insert-rows [505]
Description	Limits the number of inserts grouped per event when optimize-row-events is enabled.
Value Type	boolean
Default	50

10.8.14. P tpm Options

--policy-relay-from-slave

Option	--policy-relay-from-slave [505]	
Config File Options	policy-relay-from-slave [505]	
Description	Cross-site replicators within a Composite Active/Active deployment can be configured to point to Replicas by default, and to prefer Replicas over Primaries during operation. In a standard deployment, cross-site replicators work via Primaries at each cluster site to read the remote information. To configure the service to use Replicas in preference to Primaries, use the policy-relay-from-slave=true [505] option to tpm. Both Primaries and Replicas remain in the list of possible hosts, so if no Replicas are available during a switch or failover event, then a Primary will be used.	
Value Type	boolean	
Default	false	
Valid Values	false	
	true	

Cross-site replicators within a Composite Active/Active deployment can be configured to point to Replicas by default, and to prefer Replicas over Primaries during operation. In a standard deployment, cross-site replicators work via Primaries at each cluster site to read the remote information. To configure the service to use Replicas in preference to Primaries, use the policy-relay-from-slave=true [505] option to tpm. Both Primaries and Replicas remain in the list of possible hosts, so if no Replicas are available during a switch or failover event, then a Primary will be used.

--prefer-ip-stack

Option	--prefer-ip-stack [505]

Config File Options	prefer-ip-stack [505]	
Description	Switch between IPv4 and IPv6 support.	
Value Type	string	
Default	"4"	
Valid Values	"4"	
	"6"	

--preferred-path

Option	--preferred-path [506]
Config File Options	preferred-path [506]
Description	Additional command path
Value Type	filename

Specifies one or more additional directories that will be added before the current PATH environment variable when external commands are run from within the backup environment. This affects all external tools used by Tungsten Cluster, including MySQL, Ruby, Java, and back-up/restore tools such as Percona Xtrabackup.

One or more paths can be specified by separating each directory with a colon. For example:

```
shell> tpm ... --preferred-path=/usr/local/bin:/opt/bin:/opt/percona/bin
```

The --preferred-path [506] 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 [506]
Config File Options	prefetch-enabled [506]
Description	Should the replicator service be setup as a prefetch applier
Value Type	boolean

--prefetch-max-time-ahead

Option	--prefetch-max-time-ahead [506]
Config File Options	prefetch-max-time-ahead [506]
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 [506]
Config File Options	prefetch-min-time-ahead [506]
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 [506]
Config File Options	prefetch-schema [506]
Description	Schema to watch for timing prefetch progress
Value Type	string
Default	tungsten_

--prefetch-sleep-time

Option	--prefetch-sleep-time [506]

Config File Options	prefetch-sleep-time [506]
Description	How long to wait when the prefetch applier gets too far ahead
Value Type	string

--privileged-master

Option	--privileged-master [507]
Config File Options	privileged-master [507]
Description	Does the login for the Primary database service have superuser privileges
Value Type	boolean

--privileged-slave

Option	--privileged-slave [507]
Config File Options	privileged-slave [507]
Description	Does the login for the Replica database service have superuser privileges
Value Type	booleam

--profile-script

Option	--profile-script [507]
Config File Options	profile-script [507]
Description	Append commands to include env.sh in this profile script
Value Type	string

--protect-configuration-files

Option	--protect-configuration-files [507]	
Config File Options	protect-configuration-files [507]	
Description	When enabled, configuration files are protected to be only readable and updatable by the configured user	
Value Type	string	
Valid Values	false	Make configuration files readable by any user
	true	

When enabled (default), the configuration that contain user, password and other information are configured so that they are only readable by the configured user. For example:

```
shell> ls -al /opt/continuent/tungsten/tungsten-replicator/conf/
total 148
drwxr-xr-x 2 tungsten mysql 4096 May 14 14:32 ./
drwxr-xr-x 11 tungsten mysql 4096 May 14 14:32 ../
-rw-r--r-- 1 tungsten mysql 33 May 14 14:32 dynamic-alpha.role
-rw-r--r-- 1 tungsten mysql 5059 May 14 14:32 log4j.properties
-rw-r--r-- 1 tungsten mysql 3488 May 14 14:32 log4j-thl.properties
-rw-r--r-- 1 tungsten mysql 972 May 14 14:32 mysql-java-charsets.properties
-rw-r--r-- 1 tungsten mysql 420 May 14 14:32 replicator.service.properties
-rw-r----- 1 tungsten mysql 1590 May 14 14:35 services.properties
-rw-r----- 1 tungsten mysql 1590 May 14 14:35 .services.properties.orig
-rw-r--r-- 1 tungsten mysql 896 May 14 14:32 shard.list
-rw-r----- 1 tungsten mysql 43842 May 14 14:35 static-alpha.properties
-rw-r----- 1 tungsten mysql 43842 May 14 14:35 .static-alpha.properties.orig
-rw-r----- 1 tungsten mysql 5667 May 14 14:35 wrapper.conf
-rw-r----- 1 tungsten mysql 5667 May 14 14:35 .wrapper.conf.orig
```

When disabled, the files are readable by all users:

```
shell> ll /opt/continuent/tungsten/tungsten-replicator/conf/
total 148
drwxr-xr-x 2 tungsten mysql 4096 May 14 14:32 ./
drwxr-xr-x 11 tungsten mysql 4096 May 14 14:32 ../
-rw-r--r-- 1 tungsten mysql 33 May 14 14:32 dynamic-alpha.role
-rw-r--r-- 1 tungsten mysql 5059 May 14 14:32 log4j.properties
-rw-r--r-- 1 tungsten mysql 3488 May 14 14:32 log4j-thl.properties
-rw-r--r-- 1 tungsten mysql 972 May 14 14:32 mysql-java-charsets.properties
```

```
-rw-r--r-- 1 tungsten mysql 420 May 14 14:32 replicator.service.properties
-rw-r--r-- 1 tungsten mysql 1590 May 14 14:32 services.properties
-rw-r--r-- 1 tungsten mysql 1590 May 14 14:32 .services.properties.orig
-rw-r--r-- 1 tungsten mysql 896 May 14 14:32 shard.list
-rw-r--r-- 1 tungsten mysql 43842 May 14 14:32 static-alpha.properties
-rw-r--r-- 1 tungsten mysql 43842 May 14 14:32 .static-alpha.properties.orig
-rw-r--r-- 1 tungsten mysql 5667 May 14 14:32 wrapper.conf
-rw-r--r-- 1 tungsten mysql 5667 May 14 14:32 .wrapper.conf.orig
```

10.8.15. R tpm Options

--relay-directory

Option	`--relay-directory` [508]
Aliases	`--repl-relay-directory` [508]
Config File Options	`relay-directory` [508], `repl-relay-directory` [508]
Description	Directory for logs transferred from the Primary
Value Type	string
Default	{home directory}/relay

--relay-enabled

Option	`--relay-enabled` [508]
Config File Options	`relay-enabled` [508]
Description	Should the replicator service be setup as a relay.
Value Type	boolean

--relay-source

Option	`--relay-source` [508]
Aliases	`--dataservice-relay-source` [508], `--master-dataservice` [508]
Config File Options	`dataservice-relay-source` [508], `master-dataservice` [508], `relay-source` [508]
Description	Dataservice name to use as a relay source
Value Type	string

--repl-allow-bidi-unsafe

Option	`--repl-allow-bidi-unsafe` [508]	
Config File Options	`repl-allow-bidi-unsafe` [508]	
Description	Allow unsafe SQL from remote service	
Value Type	boolean	
Default	false	
Valid Values	false	
	true	

--repl-store-thl-compressed

Option	`--repl-store-thl-compressed` [508]	
Config File Options	`repl-store-thl-compressed` [508]	
Description	Enable (true) or disable (false) compression of THL on disk.	
Value Type	boolean	
Default	false	
Valid Values	false	Disabled
	true	Enabled

Enable (true) or disable (false) compression of THL on disk.

--repl-store-thl-encrypted

Option	--repl-store-thl-encrypted [509]	
Config File Options	repl-store-thl-encrypted [509]	
Description	Enable (true) or disable (false) encryption of THL on disk.	
Value Type	boolean	
Default	false	
Valid Values	false	Disabled
	true	Enabled

Enable (true) or disable (false) encryption of THL on disk.

--repl-svc-extractor-multi-frag-service-detection

Option	--repl-svc-extractor-multi-frag-service-detection [509]	
Config File Options	repl-svc-extractor-multi-frag-service-detection [509]	
Description	Force extraction to read ahead to last fragment to detect service name	
Value Type	boolean	
Default	false	
Valid Values	true	Enabled

When working with multi-active topologies and unprivileged database access (Such as Aurora) the replicator will need to read ahead to the last fragment to ensure multi-fragmented events are correctly flagged by the BidiRemoteSlaveFilter. Disabled by default (false). Set to true to enable.

--repl-thl-client-serialization

Option	--repl-thl-client-serialization [509]	
Config File Options	repl-thl-client-serialization [509]	
Description	Enable THL compression on downstream Replicator Appliers.	
Value Type	string	
Default	LEGACY	
Valid Values	DEFLATE	
	JAVA	
	PROTOBUF	

--repl-thl-server-serialization

Option	--repl-thl-server-serialization [509]	
Config File Options	repl-thl-server-serialization [509]	
Description	Comma Separated list of THL compression protocols to enable on the thl-server (Extractor).	
Value Type	string	
Default	LEGACY,JAVA,PROTOBUF,DEFALTE	
Valid Values	DEFLATE	
	JAVA	
	PROTOBUF	

--replace-tls-certificate

Option	--replace-tls-certificate [509]
Config File Options	replace-tls-certificate [509]
Description	Replace the TLS certificate

Replace the TLS certificate

--replica-tx-commit-level

Option	--replica-tx-commit-level [510]
Config File Options	replica-tx-commit-level [510]
Description	If set, determines the value of the underlying database property innodb_flush_log_at_trx_commit when a node becomes a replica.
Value Type	string

If set, determines the value of the underlying database property `innodb_flush_log_at_trx_commit` when a node becomes a replica.

It will be reset to 1 when the node is promoted to a primary.

Setting a value of 2 can improve the performance of writes during the apply stage of replication.

--replication-host

Option	--replication-host [510]
Aliases	--datasource-host [510], --repl-datasource-host [510]
Config File Options	datasource-host [510], repl-datasource-host [510], replication-host [510]
Description	Hostname of the datasource
Value Type	string

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

--replication-password

Option	--replication-password [510]
Aliases	--datasource-password [510], --repl-datasource-password [510]
Config File Options	datasource-password [510], repl-datasource-password [510], replication-password [510]
Description	Database password
Value Type	string

The password to be used when connecting to the database using the corresponding --replication-user [510].

--replication-port

Option	--replication-port [510]	
Aliases	--datasource-port [510], --repl-datasource-port [510]	
Config File Options	datasource-port [510], repl-datasource-port [510], replication-port [510]	
Description	Database network port	
Value Type	string	
Valid Values	1521	Oracle Default
	27017	Kafka Default
	27017	MongoDB Default
	3306	MySQL Default
	5432	PostgreSQL Default
	5433	Vertica Default
	5439	Redshift Default
	8020	HDFS Default

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

--replication-user

Option	--replication-user [510]

Aliases	`--datasource-user` [510], `--repl-datasource-user` [510]
Config File Options	`datasource-user` [510], `repl-datasource-user` [510], `replication-user` [510]
Description	User for database connection
Value Type	string

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

replicator.stage.q-to-dbms.blockCommitPolicy

Option	`replicator.stage.q-to-dbms.blockCommitPolicy` [511]	
Config File Options	`replicator.stage.q-to-dbms.blockCommitPolicy` [511]	
Description	Sets the commit policy of the replicator.	
Value Type	numeric	
Valid Values	lax	lax
	strict	strict

When set to the default (`strict`) the replicator will commit blocks on service name changes, multiple fragments in a transaction, or unsafe_for_block_commit

This property must be prefixed with `property=`:

```
property=replicator.stage.q-to-dbms.blockCommitPolicy
```

replicator.extractor.dbms.transaction_frag_size

Option	`replicator.extractor.dbms.transaction_frag_size` [511]
Config File Options	`replicator.extractor.dbms.transaction_frag_size` [511]
Description	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.
Value Type	numeric

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.

This property must be prefixed with `property=`:

```
property=replicator.extractor.dbms.transaction_frag_size
```

--replicator-rest-api

Option	`--replicator-rest-api` [511]	
Config File Options	`replicator-rest-api` [511]	
Description	Enable (default) or Disable APIv2	
Default	true	
Valid Values	false	
	true	

--replicator-rest-api-address

Option	`--replicator-rest-api-address` [511]
Config File Options	`replicator-rest-api-address` [511]
Description	Address for the API to bind too.
Default	127.0.0.1

--replicator-rest-api-authentication

Option	`--replicator-rest-api-authentication` [511]
Config File Options	`replicator-rest-api-authentication` [511]

Description	Enforce authentication for the API.
Default	true

--replicator-rest-api-port

Option	`--replicator-rest-api-port` [512]
Config File Options	`replicator-rest-api-port` [512]
Description	Port for the Replicator API.
Default	8097

--replicator-rest-api-ssl

Option	`--replicator-rest-api-ssl` [512]
Config File Options	`replicator-rest-api-ssl` [512]
Description	Enable SSL for the API.
Default	true

--reset

Option	`--reset` [512]
Config File Options	`reset` [512]
Description	Clear the current configuration before processing any arguments
Value Type	string

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

--rest-api-admin-pass

Option	`--rest-api-admin-pass` [512]
Aliases	`--rest-api-admin-password` [512]
Config File Options	`rest-api-admin-pass` [512], `rest-api-admin-password` [512]
Description	Specify the initial Admin User Password for API access. rest-api-admin-password alias only available from version 7.1.2 onwards.
Value Type	string

Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

--rest-api-admin-user

Option	`--rest-api-admin-user` [512]
Config File Options	`rest-api-admin-user` [512]
Description	Specify the initial Admin Username for API access
Value Type	string

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

--rmi-port

Option	`--rmi-port` [512]
Aliases	`--repl-rmi-port` [512]
Config File Options	`repl-rmi-port` [512], `rmi-port` [512]
Description	Replication RMI listen port
Value Type	string
Default	10001

--rmi-user

Option	--rmi-user [512]
Config File Options	rmi-user [512]
Description	The username for RMI authentication
Value Type	string

--role

Option	--role [513]	
Aliases	--repl-role [513]	
Config File Options	repl-role [513], role [513]	
Description	What is the replication role for this service?	
Value Type	string	
Valid Values	master	
	relay	
	slave	

--router-gateway-port

Option	--router-gateway-port [513]
Config File Options	router-gateway-port [513]
Description	The router gateway port
Value Type	string

--router-jmx-port

Option	--router-jmx-port [513]
Config File Options	router-jmx-port [513]
Description	The router jmx port
Value Type	string

10.8.16. S tpm Options

--security-directory

Option	--security-directory [513]
Config File Options	security-directory [513]
Description	Storage directory for the Java security/encryption files
Value Type	string

--property=server.listen.address

Option	--property=server.listen.address [443]
Config File Options	property=server.listen.address [443]
Description	IP Address to force the connector to be bound too.
Value Type	string
Default	0.0.0.0

--service-alias

Option	--service-alias [513]
Aliases	--dataservice-service-alias [513]
Config File Options	dataservice-service-alias [513], service-alias [513]
Description	Replication alias of this dataservice

Value Type	string

--service-name

Option	`--service-name` [514]
Config File Options	`service-name` [514]
Description	Set the service name

Set the service name

--service-type

Option	`--service-type` [514]	
Aliases	`--repl-service-type` [514]	
Config File Options	`repl-service-type` [514], `service-type` [514]	
Description	What is the replication service type?	
Value Type	string	
Valid Values	local	
	remote	

--skip-statemap

Option	`--skip-statemap` [514]
Config File Options	`skip-statemap` [514]
Description	Do not copy the cluster-home/conf/statemap.properties from the previous install
Value Type	boolean

--slaves

Option	`--slaves` [514]
Aliases	`--dataservice-slaves` [514], `--members` [499]
Config File Options	`dataservice-slaves` [514], `members` [499], `slaves` [514]
Description	What are the Replicas for this dataservice?
Value Type	string

--start

Option	`--start` [514]
Config File Options	`start` [514]
Description	Start the services after configuration
Value Type	string

--start-and-report

Option	`--start-and-report` [514]
Config File Options	`start-and-report` [514]
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` [514]
Aliases	`--repl-svc-allow-any-remote-service` [514]
Config File Options	`repl-svc-allow-any-remote-service` [514], `svc-allow-any-remote-service` [514]
Description	Replicate from any service

Value Type	boolean	
Default	false	
Valid Values	true	

--svc-applier-block-commit-interval

Option	--svc-applier-block-commit-interval [515]	
Aliases	--repl-svc-applier-block-commit-interval [515]	
Config File Options	repl-svc-applier-block-commit-interval [515], svc-applier-block-commit-interval [515]	
Description	Minimum interval between commits	
Value Type	string	
Valid Values	0	When batch service is not enabled
	#d	Number of days
	#h	Number of hours
	#m	Number of minutes
	#s	Number of seconds

--svc-applier-block-commit-size

Option	--svc-applier-block-commit-size [515]
Aliases	--repl-svc-applier-block-commit-size [515]
Config File Options	repl-svc-applier-block-commit-size [515], svc-applier-block-commit-size [515]
Description	Applier block commit size (min 1)
Value Type	numeric

--svc-applier-filters

Option	--svc-applier-filters [515]
Aliases	--repl-svc-applier-filters [515]
Config File Options	repl-svc-applier-filters [515], svc-applier-filters [515]
Description	Replication service applier filters
Value Type	string

--svc-applier-last-applied-write-interval

Option	--svc-applier-last-applied-write-interval [515]
Config File Options	svc-applier-last-applied-write-interval [515]
Description	Interval (in seconds) to store the last known applied seqno and latency
Value Type	string

--svc-extractor-filters

Option	--svc-extractor-filters [515]
Aliases	--repl-svc-extractor-filters [515]
Config File Options	repl-svc-extractor-filters [515], svc-extractor-filters [515]
Description	Replication service extractor filters
Value Type	string

--svc-fail-on-zero-row-update

Option	--svc-fail-on-zero-row-update [515]
Aliases	--repl-svc-fail-on-zero-row-update [515]
Config File Options	repl-svc-fail-on-zero-row-update [515], svc-fail-on-zero-row-update [515]

Description	How should the replicator behave when a Row-Based Replication UPDATE or DELETE does not affect any rows.	
Value Type	string	
Default	stop	
Valid Values	ignore	No warnings in the log file, and replication continues
	warn	Log a Warning in the log file, but continue anyway

Warning

From release 7.0.1 the default for this property was changed to `stop`, previously, the default was `warn`.

The change in the default value may cause unexpected behavior in Active/Active topologies due to the Asynchronous nature of replication, however care should be taken if changing back to the original default of `warn`.

If you notice many entries in your replicator logs indicating zero row updates, and these warnings are being ignored, you may encounter data drift.

--svc-parallelization-type

Option	`--svc-parallelization-type` [516]	
Aliases	`--repl-svc-parallelization-type` [516]	
Config File Options	`repl-svc-parallelization-type` [516], `svc-parallelization-type` [516]	
Description	Method for implementing parallel apply	
Value Type	string	
Valid Values	disk	
	memory	
	none	

--svc-remote-filters

Option	`--svc-remote-filters` [516]
Aliases	`--repl-svc-remote-filters` [516]
Config File Options	`repl-svc-remote-filters` [516], `svc-remote-filters` [516]
Description	Replication service remote download filters
Value Type	string

--svc-reposition-on-source-id-change

Option	`--svc-reposition-on-source-id-change` [516]
Aliases	`--repl-svc-reposition-on-source-id-change` [516]
Config File Options	`repl-svc-reposition-on-source-id-change` [516], `svc-reposition-on-source-id-change` [516]
Description	The Primary will come ONLINE from the current position if the stored source_id does not match the value in the static properties
Value Type	string

--svc-shard-default-db

Option	`--svc-shard-default-db` [516]	
Aliases	`--repl-svc-shard-default-db` [516]	
Config File Options	`repl-svc-shard-default-db` [516], `svc-shard-default-db` [516]	
Description	Mode for setting the shard ID from the default db	
Value Type	string	
Valid Values	relaxed	
	stringent	

--svc-systemd-config-connector

Option	--svc-systemd-config-connector [516]
Config File Options	svc-systemd-config-connector [516]
Description	Used to provide custom systemd configuration that will be used in place of the default generated on.
Value Type	string

This property can be used to supply an alternative, custom, systemd configuration that will be used in place of the default generated one.

The value of the parameter should be specified as the name and location of the custom script to use instead, for example:

```
svc-systemd-config-connector=/home/tungsten/connector.service
```

Where connector.service file could look something like the following:

```
[Unit]
Description=Tungsten Connector
After=mysqld.service mysql.service mariadb.service tmanager.service treplictor.service

[Service]
User=tungsten
Type=forking
ExecStart=/opt/continuent/tungsten/tungsten-connector/bin/connector start sysd
ExecStop=/opt/continuent/tungsten/tungsten-connector/bin/connector stop sysd
ExecStartPre=/opt/my-pre-start-script.sh

[Install]
WantedBy=multi-user.target
```

--svc-systemd-config-manager

Option	--svc-systemd-config-manager [517]
Config File Options	svc-systemd-config-manager [517]
Description	Used to provide custom systemd configuration that will be used in place of the default generated on.
Value Type	string

This property can be used to supply an alternative, custom, systemd configuration that will be used in place of the default generated one.

The value of the parameter should be specified as the name and location of the custom script to use instead, for example:

```
svc-systemd-config-manager=/home/tungsten/manager.service
```

Where manager.service file could look something like the following:

```
[Unit]
Description=Tungsten Manager
After=mysqld.service mysql.service mariadb.service treplicator.service

[Service]
User=tungsten
Type=forking
ExecStart=/opt/continuent/tungsten/tungsten-manager/bin/manager start sysd
ExecStop=/opt/continuent/tungsten/tungsten-manager/bin/manager stop sysd
ExecStartPre=/opt/my-pre-start-script.sh

[Install]
WantedBy=multi-user.target
```

--svc-systemd-config-replicator

Option	--svc-systemd-config-replicator [517]
Config File Options	svc-systemd-config-replicator [517]
Description	Used to provide custom systemd configuration that will be used in place of the default generated on.
Value Type	string

This property can be used to supply an alternative, custom, systemd configuration that will be used in place of the default generated one.

The value of the parameter should be specified as the name and location of the custom script to use instead, for example:

```
svc-systemd-config-replicator=/home/tungsten/replicator.service
```

Where replicator.service file could look something like the following:

```
[Unit]
Description=Tungsten Replicator
After=mysqld.service mysql.service mariadb.service

[Service]
User=tungsten
Type=forking
ExecStart=/opt/continuent/tungsten/tungsten-replicator/bin/replicator start sysd
ExecStop=/opt/continuent/tungsten/tungsten-replicator/bin/replicator stop sysd
ExecStartPre=/opt/my-pre-start-script.sh

[Install]
WantedBy=multi-user.target
```

--svc-table-engine

Option	--svc-table-engine [518]
Aliases	--repl-svc-table-engine [518]
Config File Options	repl-svc-table-engine [518], svc-table-engine [518]
Description	Replication service table engine
Value Type	string
Default	innodb

--svc-thl-filters

Option	--svc-thl-filters [518]
Aliases	--repl-svc-thl-filters [518]
Config File Options	repl-svc-thl-filters [518], svc-thl-filters [518]
Description	Replication service THL filters
Value Type	string

10.8.17. T tpm Options

--target-dataservice

Option	--target-dataservice [518]
Aliases	--slave-dataservice [518]
Config File Options	slave-dataservice [518], target-dataservice [518]
Description	Dataservice to use to determine the value of host configuration
Value Type	string

--temp-directory

Option	--temp-directory [518]
Config File Options	temp-directory [518]
Description	Temporary Directory
Value Type	string

--template-file

Option	--template-file [518]
Config File Options	template-file [518]
Description	Display the keys that may be used in configuration template files
Value Type	string

--template-search-path

Option	--template-search-path [518]
Config File Options	template-search-path [518]

Description	Adds a new template search path for configuration file generation
Value Type	filename

--thl-directory

Option	`--thl-directory` [519]	
Aliases	`--repl-thl-directory` [519]	
Config File Options	`repl-thl-directory` [519], `thl-directory` [519]	
Description	Replicator log directory	
Value Type	string	
Default	{home directory}/thl	
Valid Values	{home directory}/thl	

The given value should be the base directory, to which tungsten will add the service name. For example, the following entry in the tungsten.ini:

```
[alpha]
...
...
thl-directory=/drv1/thl
...
```

Would result in the THL being placed in /drv1/thl/alpha

Note

Update of thl directory is only available when tpm is called from the staging installation directory, NOT from the running directory.

--thl-do-checksum

Option	`--thl-do-checksum` [519]
Aliases	`--repl-thl-do-checksum` [519]
Config File Options	`repl-thl-do-checksum` [519], `thl-do-checksum` [519]
Description	Execute checksum operations on THL log files
Value Type	string

--thl-interface

Option	`--thl-interface` [519]
Aliases	`--repl-thl-interface` [519]
Config File Options	`repl-thl-interface` [519], `thl-interface` [519]
Description	Listen interface to use for THL operations
Value Type	string

--thl-log-connection-timeout

Option	`--thl-log-connection-timeout` [519]
Aliases	`--repl-thl-log-connection-timeout` [519]
Config File Options	`repl-thl-log-connection-timeout` [519], `thl-log-connection-timeout` [519]
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` [519]
Aliases	`--repl-thl-log-file-size` [519]
Config File Options	`repl-thl-log-file-size` [519], `thl-log-file-size` [519]

Description	File size in bytes for THL disk logs
Value Type	numeric

--thl-log-fsync

Option	`--thl-log-fsync` [520]
Aliases	`--repl-thl-log-fsync` [520]
Config File Options	`repl-thl-log-fsync` [520], `thl-log-fsync` [520]
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` [520]	
Aliases	`--repl-thl-log-retention` [520]	
Config File Options	`repl-thl-log-retention` [520], `thl-log-retention` [520]	
Description	How long do you want to keep THL files.	
Value Type	string	
Default	7d	
Valid Values	#d	Number of days
	#h	Number of hours
	#m	Number of minutes
	#s	Number of seconds

--thl-port

Option	`--thl-port` [520]
Aliases	`--repl-thl-port` [520]
Config File Options	`repl-thl-port` [520], `thl-port` [520]
Description	Port to use for THL Operations
Value Type	numeric
Default	2112

--thl-protocol

Option	`--thl-protocol` [520]
Aliases	`--repl-thl-protocol` [520]
Config File Options	`repl-thl-protocol` [520], `thl-protocol` [520]
Description	Protocol to use for THL communication with this service
Value Type	string

--topology

Option	`--topology` [520]	
Aliases	`--dataservice-topology` [520]	
Config File Options	`dataservice-topology` [520], `topology` [520]	
Description	Replication topology for the dataservice.	
Value Type	string	
Valid Values	all-masters	
	cluster-alias	
	cluster-slave	

	clustered	
	direct	
	fan-in	
	master-slave	
	star	

--track-schema-changes

Option	`--track-schema-changes` [521]
Config File Options	`track-schema-changes` [521]
Description	This will enable filters that track DDL statements and write the resulting change to files on Replica hosts. The feature is intended for use in some batch deployments.
Value Type	string

10.8.18. U tpm Options

--user

Option	`--user` [521]
Config File Options	`user` [521]
Description	System User
Value Type	string

10.8.19. W tpm Options

--witnesses

Option	`--witnesses` [521]
Aliases	`--dataservice-witnesses` [521]
Config File Options	`dataservice-witnesses` [521], `witnesses` [521]
Description	Witness hosts for the dataservice
Value Type	string

Chapter 11. Tungsten REST APIv2

Version 7.0.0 of the Tungsten suite of products introduces the Public RESTful API (Referred to as APIv2)

The API will allow quick and easy access to monitoring, cluster manipulation and change of configuration for both humans and software.

> **Note**
>
> This documentation section refers to APIv2 for v7.x Tungsten Releases. To view documentation related to v8, please click here.

A number of key development decisions where made to allow the release (APIv2). The choice was made to release with a high level of security by default, and a minimal set of features so we can release quickly. Future releases will build on this initial foundation:

- Security First: the API is only enabled on localhost (127.0.0.1) by default. Additionally, SSL is now enabled by default (in previous releases, SSL was disabled by default).
- User/Password Protection: HTTP Basic Auth is required to access most API calls. RBAC will come in a future version.
- Per-layer API: each component has its own API.
- Per-host instances: Passwords are stored securely on each host. Of course, the Tungsten installation tool (tpm) allows deployment of identical user/password pairs across nodes, as does the Tungsten Dashboard GUI.
- The REST API provides read-only information through HTTP `GET` calls. Continuent has made the deliberate choice of using only `POST` actions when they affect the state or configuration of the software. The reason why we made that choice is a matter of simplification: there is an overlap between the definitions of `PUT` and `POST`, and the decision to chose between the two is often a balance with pros and cons, triggering sterile discussions. Rather that spending time debating and choosing, we decided to offer only one, so you will not find any use of `PUT` in our list of functions.

For the full developer documentation, listing all available calls, please refer to the following links:

- Proxy Developer Documentation
- Manager Developer Documentation
- Replicator Developer Documentation
- Common to all Componenents Developer Documentation

Each layer of the product (Manager, Replicator, Proxy) has its own API, allowing for granular access and control of all cluster operations.

> **Warning**
>
> API calls triggering configuration changes are protected by a flag, `i-am-sure=true`, in order to avoid unwanted, potentially dramatic, configuration changes. This applies to:
>
> - configuration/module/servicesmap
> - reset
> - offline
> - online
> - onhold
> - addDataService
> - addDataSource

11.1. Getting Started with Tungsten REST API

11.1.1. Configuring the API

11.1.1.1. Network Ports

Each cluster process listens for API calls using the following ports by default:

Port	Component
8090	Manager
8096	Connector (Proxy)
8097	Replicator

Should you wish to choose your own ports for the API, this can be handled by specifying the new ports in the tpm config using the following properties:

```
connector-rest-api-port=8096
manager-rest-api-port=8090
replicator-rest-api-port=8097
```

11.1.1.2. User Management

With our focus on security in the first release of the API, without any user created, only ping and createAdminUser calls will be available. tpm makes it easy to create the administration user at install through the following options:

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

Once an admin user is created, all other APIs call will be available using basic HTTP authentication with that user/password.

If you haven't created the user as part of the initial cluster installation, you can use the Dashboard GUI to do this for you during Dashboard configuration, or use the CLI calls explained below.

Note

If you plan to use the Dashboard GUI, you MUST ensure the API User has been created to alow the Dashboard to function correctly.

Using CURL commands

The createAdminUser call will allow creation and modification of REST API users.

```
shell> curl -k -H 'Content-type: application/json' --request POST 'https://127.0.0.1:8096/api/v2/createAdminUser?i-am-sure=true' \
> --data-raw '{
>   "payloadType": "credentials",
>   "user":"tungsten",
>   "pass":"security"
> }'

{ "payloadType":"StringPayload",
  "payloadVersion":"1",
  "payload":{
      "value":"https://127.0.0.1:8096/api/v2/user/tungsten"
           }
}
```

Using tapi

The tapi tool can make the creation of the admin user simple and straightforward, the following example shows the use of this approach.

```
shell> tapi --create --host hostnamehere --create-user usernamehere --create-password passwordhere -v
```

Important

Note that users created/modified through the curl and tapi calls only apply to the host on which the call was done. The same, identical call, will have to be re-run on each host of the cluster.

Important

Usernames must NOT contain any of the following reserved characters: `(space) ! # $ & ' ( ) * + , / : ; = ? @ [ ]`

11.1.1.3. SSL/Encryption

SSL is enabled by default and will use the same keystore, certificates and aliases as the cluster TLS ones.

You can specify them with `java-tls-keystore-path` and `java-truststore-path` tpm options, or let tpm generate the certificates for you during installation.

For more detailed information on SSL, see Chapter 5, *Deployment: Security*

11.1.1.4. Enabling and Disabling the API

As previously mentioned, the API is enabled by default, listening only on localhost.

To disable the API, you will need to specify the following tpm options:

```
replicator-rest-api=false
manager-rest-api=false
connector-rest-api=false
```

To change the address that the API is listening on, you need to specify the following tpm options:

```
replicator-rest-api-address=0.0.0.0
connector-rest-api-address=0.0.0.0
manager-rest-api-address=0.0.0.0
```

For more information on all of the tpm options for managing the API, see Section 11.1.4, “Summary of all API TPM options”

11.1.2. How to Access the API

11.1.2.1. CURL calls and Examples

curl is the simplest command line tool to access the API.

Note that in the examples below, we use self signed certificates, which is why the `--insecure` or `-k` flag is passed to curl

Ping command

```
shell> curl -k --request GET 'https://127.0.0.1:8096/api/v2/ping'

{ "payloadType":"PingPayload",
  "payloadVersion":"1",
  "payload":{
      "message":"Ping test",
      "date":"Mon Mar 10 10:12:18 UTC 2025",
      "hostName":"gilmbp-8.local",
      "pid":16743,
      "jvmUptime":152513
      }
}
```

In this example, we sent a simple ping. Result comes as an `HTTP OK` response with a payload containing a string message “ping test”, the date of execution, the hostname of the executor and its pid, plus the number of milliseconds elapsed since the java executable was started

Create admin user

```
shell> curl -k -H 'Content-type: application/json' --request POST 'https://127.0.0.1:8096/api/v2/createAdminUser?i-am-sure=true' \
> --data-raw '{
>   "payloadType": "credentials",
>   "user":"tungsten",
>   "pass":"security"
> }'

{ "payloadType":"StringPayload",
  "payloadVersion":"1",
  "payload":{
      "value":"https://127.0.0.1:8096/api/v2/user/tungsten"
           }
}
```

This `POST` request shows how input data can be passed to API calls as a json string.

In return, the createAdminUser function will give a link to the API call allowing you to display details for the user we just created. Let’s follow this link:

```
shell> curl -k --request GET 'https://127.0.0.1:8096/api/v2/user/tungsten' -utungsten:security

{ "payloadType":"CredentialsPayload",
  "payloadVersion":"1",
  "payload":{
      "user":"tungsten",
      "pass":"**obfuscated**",
      "access":"full"
           }
}
```

This call shows how the user/password tuple is passed.

The equivalent call using base64 encoded Authorization header, obtained with `echo -ne "tungsten:security" | base64` would look something like the following:

```
shell> curl -k --request GET 'https://127.0.0.1:8096/api/v2/user/tungsten' --header 'Authorization: Basic dHVuZ3N0ZW46c2VjdXJpdHk='

  "payload":{
      "user":"tungsten",
      "pass":"**obfuscated**",
      "access":"full"
           }
}
```

11.1.2.2. tapi

tapi is a convenient command line tool developed by Continuent that will ease access to Tungsten components APIs without having to remember full REST call URLs.

Full documentation on tapi can be found here: Section 9.19, “The tapi Command”

11.1.2.3. External Tools

PostMan is one of the most popular GUI tools for REST API testing and use

11.1.3. Data Structures

11.1.3.1. Generic Payloads

Tungsten API defines its own payloads for both inputs and output. The generic structure looks like the following:

```
{
   "payloadType": "TypeOfPayload",
   "payloadVersion": "1",
   "payload":
   {
     "key"="value"
   }
}
```

Where `payloadType` announces the type of data that will be contained in `payload` in the given `payloadVersion`

As an example, a very simple payload is found in the `StringPayload` data structure and only consists in a key/value pair:

```
{
   "payloadType": "StringPayload",
   "payloadVersion": "1",
   "payload":
   {
      "value"="stringvalue"
   }
}
```

11.1.3.2. INPUT and OUTPUT payloads

When starting up with a fresh installation of tungsten, if no admin user has been provided to tpm, credentials can be sent to the various functions via the following payload:

```
{
    "payloadType": "CredentialsPayload",
    "payload":
    {
     "user":"<user>",
     "pass":"<password>"
    }
}
```

The same payload structure, slightly enriched, will be found in response to listing the user via:

```
{
    "payloadType": "CredentialsPayload",
    "payloadVersion": "1",
    "payload": {
        "user": "tungsten",
        "pass": "<obfuscated>",
        "access": "full"
    }
```

```
}
```

Various other payloads used and produced by Tungsten REST API entry point will be found in the detailed technical documentation. Links below:

- Proxy Developer Documentation
- Manager Developer Documentation
- Replicator Developer Documentation
- Common to all Components Developer Documentation

11.1.4. Summary of all API TPM options

The following is a summary of all of the tpm options for controlling the various API settings:

User Management Options

Parameter	Description	Default Value
rest-api-admin-user [512]	Specify the initial Admin User Name for API access.	
rest-api-admin-pass [512], rest-api-admin-password [512]	Specify the initial Admin User Password for API access. rest-api-admin-password [512] alias only available from version 7.1.2 onwards.	

Connector Options

Parameter	Description	Default Value
connector-rest-api [482]	Enable or Disable the API.	true when disable-security-controls=true [490] or not defined.
connector-rest-api-ssl [483]	Enable or Disable SSL	true when disable-security-controls=true [490] or not defined.
connector-rest-api-address [482]	Address for the API to bind to	127.0.0.1
connector-rest-api-port [482]	Port for the Connector API	8096
connector-rest-api-authentication [482]	Enforce authentication for the API	true

Manager Options

Parameter	Description	Default Value
manager-rest-api [498]	Enable or Disable the API.	true when disable-security-controls=true [490] or not defined.
manager-rest-api-ssl [498]	Enable or Disable SSL	true when disable-security-controls=true [490] or not defined.
manager-rest-api-address [498]	Address for the API to bind to	127.0.0.1
manager-rest-api-port [498]	Port for the Manager API	8090
manager-rest-api-authentication [498]	Enforce authentication for the API	true

Replicator Options

Parameter	Description	Default Value
replicator-rest-api [511]	Enable or Disable the API.	true when disable-security-controls=true [490] or not defined.
replicator-rest-api-ssl [512]	Enable or Disable SSL	true when disable-security-controls=true [490] or not defined.

Parameter	Description	Default Value
replicator-rest-api-address [511]	Address for the API to bind to	127.0.0.1
replicator-rest-api-port [512]	Port for the Replicator API	8097
replicator-rest-api-authentication [511]	Enforce authentication for the API	true

11.2. Proxy (Connector) API Specifics

The Proxy API is enabled by default, but only listens to `localhost` connections on port 8096.

The proxy REST API can be disabled with the tpm flag:

```
connector-rest-api=false
```

If required, the listen port can be changed using the tpm option:

```
connector-rest-api-port=8096
```

Exposing the API to a different network address for remote access, like an internal network, consists in changing the listen address of the API server.

For example, granting access to local 192.168.1.* network would translate to the tpm option:

```
connector-rest-api-address=192.168.1.0
```

> **Warning**
>
> Note that exposing the API to a public network can introduce a security breach like brute-force or DDoS attack exposure

The Proxy Developer Documentation can be viewed here

11.2.1. Proxy States

Tungsten Connector Proxy has 3 different operational states detailed here Section 7.8, “Connector Operational States”

The REST API provides easy access to this status shown in the following example

```
GET 'https://localhost:8096/api/v2/connector/status'
```

The response will be wrapped inside a simple string payload with the current state:

```
{
    "payloadType": "StringPayload",
    "payloadVersion": "1",
    "payload": {
        "value": "OFFLINE"
    }
}
```

Changing this status is NOT recommend during regular and normal operations. If needed, it can still be done with the following calls:

```
https://127.0.0.1:8096/api/v2/connector/online

https://127.0.0.1:8096/api/v2/connector/onhold

https://127.0.0.1:8096/api/v2/connector/offline
```

Note that this last call can be flagged to go offline immediately with the `immediate` flag:

```
https://127.0.0.1:8096/api/v2/connector/offline?immediate=true
```

11.2.2. List and Set Proxy Configuration

Tungsten proxy configuration is divided into several categories. All categories can be listed using:

```
GET 'https://127.0.0.1:8096/api/v2/connector/configuration/'
```

For example:

```
GET 'https://127.0.0.1:8096/api/v2/connector/configuration/module/extra'
```

will return the product version and paths to the key and trust stores, wrapped inside a `TungstenPropertiesPayload`:

```
{
    "payloadType": "TungstenPropertiesPayload",
    "payloadVersion": "1",
    "payload": {
        "TungstenVersion": "Tungsten Clustering 7.0.3",
        "javax.net.ssl.trustStore": "../../tungsten-connector/conf/tungsten_truststore.ts",
        "javax.net.ssl.keyStore": "../../tungsten-connector/conf/tungsten_keystore.jks"
    }
}
```

Changing a configuration value is done through a POST call with the value passed either as a URL parameter:

```
POST 'https://c1:8096/api/v2/connector/configuration/module/connector/sslCapable?value=true'
```

or as a JSON `ConfigurationItemPayload`

```
POST 'https://127.0.0.1:8096/api/v2/connector/configuration/module/connector'
{
    "payloadType": "ConfigurationItemPayload",
    "payload": {
        "item": "sslCapable",
        "value": "true"
    }
}
```

11.2.3. User Map

Current user map entries can be listed with

```
GET 'https://127.0.0.1:8096/api/v2/connector/configuration/module/userMap'
{
    "payloadType": "TungstenPropertiesPayload",
    "payloadVersion": "1",
    "payload": {
        "app_user": "% europe null",
        "other_user": "% europe null"
    }
}
```

In order to create a user in memory, the password and data service name will have to be passed as a url parameter:

```
POST 'https://127.0.0.1:8096/api/v8/connector/configuration/module/userMap/app_user?value=secret europe'
```

Note that after this call, the user map only exists in memory. Persistent storage to disk will need to be done as follows:

```
POST 'https://127.0.0.1:8096/api/v2/connector/configuration/module/userMap/writeToDisk'
```

11.2.4. Cluster Configuration Manipulation

Within a Continuent Tungsten Cluster installation, the connector receives its cluster configuration directly from the managers. It is NOT desirable to change it, there are high risks of destabilizing the cluster, creating split brains or writing data to replicas.

For this reason, if the Proxy is connected to a cluster, a confirmation flag `?i-am-sure=true` will have to be passed to the following call examples.

For standalone proxies cluster configuration changes are accessible without the confirmation flag.

11.2.4.1. Creating a Data Service

To create a data service name "europe"

```
POST 'https://127.0.0.1:8096/api/v2/connector/addDataService/?dataServiceName=europe'
```

11.2.4.2. Adding a Data Source

Adding a data source requires the following call with a payload of type TungstenPropertiesPayload

```
POST 'https://127.0.0.1:8096/api/v2/connector/addDataSource'
{
    "payloadType": "TungstenPropertiesPayload",
    "payloadVersion": "1",
    "payload": {
        "name": "c4",
        "driver": "org.drizzle.jdbc.DrizzleDriver",
        "isAvailable": true,
```

```
        "role": "slave",
        "host": "c4",
        "state": "ONLINE",
        "url": "jdbc:mysql:thin://c4:13306/${DBNAME}?jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull&tinyInt1isBit=false&allowMultiQuerie
        "dataServiceName": "europe",
        "isComposite": false
    }
}
```


Important

No validation of the data source will be performed when creating it. Errors, if any, will appear when using/connecting-to the data source

11.2.4.3. Creating a Full Cluster Configuration

A cluster configuration, called "data services map", can be passed through a single call to

```
POST 'https://127.0.0.1:8096/api/v2/connector/configuration/module/servicesmap'
```

The cluster map may look like the following example:


```
{
    "payloadType": "DataServicesMapPayload",
    "payload": {
        "europe": {
            "c1": {
                "dataServiceName": "europe",
                "name": "c1",
                "isAvailable": true,
                "role": "master",
                "host": "c1",
                "state": "ONLINE",
                "url": "jdbc:mysql:thin://c1:13306/${DBNAME}?jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull&tinyInt1isBit=false&allowMul
                "isComposite": false,
                "driver": "org.drizzle.jdbc.DrizzleDriver"
            },
            "c2": {
                "dataServiceName": "europe",
                "name": "c2",
                "isAvailable": true,
                "role": "slave",
                "host": "c2",
                "state": "ONLINE",
                "url": "jdbc:mysql:thin://c2:13306/${DBNAME}?jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull&tinyInt1isBit=false&allowMul
                "isComposite": false,
                "driver": "org.drizzle.jdbc.DrizzleDriver"
            },
            "c3": {
                "dataServiceName": "europe",
                "name": "c3",
                "isAvailable": true,
                "role": "slave",
                "host": "c3",
                "state": "ONLINE",
                "url": "jdbc:mysql:thin://c3:13306/${DBNAME}?jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull&tinyInt1isBit=false&allowMul
                "isComposite": false,
                "driver": "org.drizzle.jdbc.DrizzleDriver"
            }
        }
    }
}
```


11.2.4.4. Data Source Changes, Switches and Failovers

The same REST API entry point can be used to modify the current cluster configuration. Bringing a node offline can be done by passing a data source configuration and setting the flag `"isAvailable": false`

Similarly, a full "switch" operation will be accomplished by exchanging the "roles", giving the new primary the `"master"` role, and the replica `"slave"`. Note that, just as Tungsten Clustering does, you might want to first set both nodes to `"isAvailable": false` before proceeding to the role change.

11.2.4.5. Resetting the Cluster Configuration

This call should obviously used with great care as it will completely remove all known data sources and data services from the current proxy instance; thus the confirmation flag `i-am-sure=true` has to be passed

```
https://127.0.0.1:8096/api/v2/connector/reset?i-am-sure=true
```

11.3. Manager API Specifics

The Manager API is enabled by default, but only listens to `localhost` connections on port 8090.

The Manager REST API can be disabled with the tpm flag:

```
manager-rest-api=false
```

If required, the listen port can be changed using the tpm option:

```
manager-rest-api-port=8090
```

Exposing the API to a different network address for remote access, like an internal network, consists in changing the listen address of the API server.

For example, granting access to local 192.168.1.* network would translate to the tpm option:

```
manager-rest-api-address=192.168.1.0
```

> **Warning**
>
> Note that exposing the API to a public network can introduce a security breach like brute-force or DDoS attack exposure

The Manager Developer Documentation can be viewed here

11.3.1. Manager Status

Using the following `GET` call, you can list the manager status. This will display a number of useful metrics, such as start time, datasource, cluster name, the current coordinator, and more.

```
GET https://localhost:8090/api/v2/manager/status
```

This will return a simple StatusPayload, similar to the following example:

```
{
  "payloadType" : "StatusPayload",
  "payloadVersion" : "1",
  "payload" : {
    "dataServiceName" : "west",
    "dataSourceName" : "db1",
    "startTime" : "2024-08-18T06:43:57.873 UTC",
    "uptimeSeconds" : 151,
    "state" : "ONLINE",
    "isCoordinator" : false,
    "isWitness" : false,
    "managerPID" : 4527,
    "parentPID" : 4508,
    "policyMode" : "AUTOMATIC",
    "coordinator" : "db3",
    "jmxUseEncryption" : true,
    "jmxUseAuthentication" : true,
    "jgroupsUseEncryption" : true,
    "dbUseEncryption" : true,
    "keystoreLocation" : "/opt/continuent/share/tungsten_keystore.jks",
    "trustoreLocation" : "/opt/continuent/share/tungsten_truststore.ts"
  }
}
```

11.3.2. Cluster Topology

The REST API provides easy access to view the cluster topology, as shown in the following example

```
GET 'https://localhost:8090/api/v2/manager/cluster/topology'
```

The response will be wrapped inside a ClusterTopologyPayload and look something like the following:

```
{
    "payloadType": "ClusterTopologyPayload",
    "payloadVersion": "1",
    "payload": {
        "topology": "CLUSTERED_MASTER_SLAVE",
        "name": "usa",
        "services": [
            {
```

```
                "name": "east",
                "role": "slave",
                "datasources": [
                    {
                        "name": "db4",
                        "role": "relay"
                    },
                    {
                        "name": "db5",
                        "role": "slave"
                    },
                    {
                        "name": "db6",
                        "role": "slave"
                    }
                ]
            },
            {
                "name": "west",
                "role": "master",
                "datasources": [
                    {
                        "name": "db1",
                        "role": "master"
                    },
                    {
                        "name": "db2",
                        "role": "slave"
                    },
                    {
                        "name": "db3",
                        "role": "slave"
                    }
                ]
            }
        ],
        "routers": [
            {
                "name": "db1",
                "connections": {
                    "active": 0,
                    "created": 0
                }
            },
            {
                "name": "db2",
                "connections": {
                    "active": 0,
                    "created": 0
                }
            },
            {
                "name": "db3",
                "connections": {
                    "active": 0,
                    "created": 0
                }
            },
            {
                "name": "db4",
                "connections": {
                    "active": 0,
                    "created": 0
                }
            },
            {
                "name": "db5",
                "connections": {
                    "active": 0,
                    "created": 0
                }
            },
            {
                "name": "db6",
                "connections": {
                    "active": 0,
                    "created": 0
                }
            }
        ]
    }
}
```


It's also possible to omit the routers from the topology output, using the following call:

```
GET https://localhost:8090/api/v2/manager/cluster/topology?includeRouters=no
```

11.3.3. Service Status

To view the current service status for the usa cluster, the following command can be called (Note we are excluding the routers in this example for readability)

```
GET https://localhost:8090/api/v2/manager/status/service/usa?includeRouters=false
```

The returned ServiceStatusPayload will look something like the following

```
{
    "payloadType": "ServiceStatusPayload",
    "payloadVersion": "1",
    "payload": {
        "name": "usa",
        "type": "CLUSTERED_MASTER_SLAVE",
        "composite": true,
        "multimaster": false,
        "policy": "AUTOMATIC",
        "coordinators": [
            "db2",
            "db4"
        ],
        "datasources": [
            "east",
            "west"
        ]
    }
}
```

11.3.4. Datasource Status

A datasource within a cluster could either refer to the entire cluster service itself, or a specific host.

To list the status of the datasource at the cluster level, the following call can be used

```
GET https://localhost:8090/api/v2/manager/status/service/usa/datasource/west
```

This will return the following DatasourceStatusPayload

```
{
    "payloadType": "DatasourceStatusPayload",
    "payloadVersion": "1",
    "payload": {
        "name": "west",
        "service": "usa",
        "role": "master",
        "state": "ONLINE",
        "connections": {
            "active": 0,
            "created": 0
        },
        "seqno": -1,
        "appliedLatency": 1.254,
        "relativeLatency": 0.0,
        "standby": false,
        "archive": false,
        "witness": false
    }
}
```

Similarly, to list the status of the datasource at the host level, the following call can be used

```
GET https://localhost:8090/api/v2/manager/status/service/west/datasource/db1
```

This will return the following DatasourceStatusPayload

```
{
    "payloadType": "DatasourceStatusPayload",
    "payloadVersion": "1",
    "payload": {
        "name": "db1",
        "service": "west",
        "role": "master",
        "state": "ONLINE",
        "connections": {
```

```
            "active": 42,
            "created": 128
        },
        "seqno": 12856,
        "appliedLatency": 0.745,
        "relativeLatency": 1.647,
        "standby": false,
        "archive": false,
        "witness": false
    }
}
```

11.3.5. Cluster Control Commands

Using the API, it is possible to do a backup, switch, shun or welcome the node, put replicator online/offline, ping a datasource and all other commands that can be achieved from the cctrl command line utility. Here we present the ping and the switch command.

A full list of all the commands are listed in the Manager API Developer Documentation

11.3.5.1. Ping a Datasource

To ping a datasource, the following command can be used, resulting in a BooleanPayload return as shown in the following example

```
shell> curl 'GET https://127.0.0.1:8090/api/v2/manager/service/west/datasource/db1/ping'
{
    "payloadType": "BooleanPayload",
    "payloadVersion": "1",
    "payload": {
        "value": true
    }
}
```

11.3.5.2. Issue a Switch

To initiate a switch within a cluster, you can issue the following commands, resulting in a TaskPayload

```
shell> curl 'GET https://localhost:8090/api/v2/manager/control/service/west/switch'
{
    "payloadType": "TaskPayload",
    "payloadVersion": "1",
    "payload": {
        "taskId": "84ad5757-0a6c-44d9-a631-686a740150a3",
        "state": "in_progress"
    }
}
```

A cluster switch is deemed a long lasting command, and therefore you can then use the taskId returned from above, to view the status of the switch, as follows

```
shell> curl 'GET https://127.0.0.1:8090/api/v2/manager/task/84ad5757-0a6c-44d9-a631-686a740150a3'

{
    "payloadType": "TaskPayload",
    "payloadVersion": "1",
    "payload": {
        "taskId": "84ad5757-0a6c-44d9-a631-686a740150a3",
        "state": "complete",
        "taskResult": {
            "newPrimary": "db3"
        }
    }
}
```

From the output, you can see the switch completed and the new Primary node is db3.

11.4. Replicator API Specifics

The Replicator API is enabled by default but only listens to `localhost` connections on port 8097.

The Replicator REST API can be disabled with the tpm flag:

```
replicator-rest-api=false
```

If required, the listen port can be changed using the tpm option:

```
replicator-rest-api-port=8097
```

Exposing the API to a different network address for remote access, like an internal network, consists in changing the "listen address" of the API server. For example, granting access to local 192.168.1.* network would translate to the tpm flag:

```
replicator-rest-api-listen-address=192.168.1.0
```

> **Warning**
>
> Note that exposing the API to a public network can introduce a security breach like brute-force or DDoS attacks exposure.

The replicator API has two sets of API calls, based either on the whole replicator or on a service of the replicator. A few examples follow and the full Replicator Developer Documentation can be viewed here

11.4.1. Replicator Endpoints

The following endpoints are available for the whole replicator :

- `services`
- `status`
- `version`
- `online`
- `offline`
- `purge`
- `reset`

11.4.1.1. services

The replicators existing services can be listed with `GET` request as follows

```
GET 'https://127.0.0.1:8097/api/v2/replicator/services'
```

The output will look like


```
{
  "payloadType": "ServicesPayload",
  "payloadVersion": "1",
  "payload": [
    {
      "serviceType": "unknown",
      "appliedLastSeqno": -1,
      "relativeLatency": -1,
      "role": "slave",
      "appliedLatency": -1,
      "masterConnectUri": "thl://centos1:2112/",
      "started": true,
      "state": "OFFLINE:ERROR",
      "serviceName": "east"
    },
    {
      "serviceType": "local",
      "appliedLastSeqno": 28,
      "relativeLatency": 434041.992,
      "role": "master",
      "appliedLatency": 1.138,
      "masterConnectUri": "thls://localhost:/",
      "started": true,
      "state": "ONLINE",
      "serviceName": "west"
    }
  ]
}
```


11.4.1.2. status

The status of the replicator services will be retrieved using a `GET` request as follows

```
GET 'https://127.0.0.1:8097/api/v2/replicator/status'
```

This will show the individual status of all the replicator services, as this example shows:


```
{
  "payloadType": "ReplicatorStatusPayload",
  "payloadVersion": "1",
  "payload": {
    "east": {
      "appliedLastEventId": "NONE",
      "appliedLastSeqno": -1,
      "appliedLatency": -1,
      "autoRecoveryEnabled": false,
      "autoRecoveryTotal": 0,
      "channels": -1,
      "clusterName": "east",
      "currentEventId": "NONE",
      "currentTimeMillis": 1646836106399,
      "dataServerHost": "continuent4",
      "extensions": "",
      "host": "continuent4",
      "latestEpochNumber": -1,
      "masterConnectUri": "thl://centos1:2112/",
      "masterListenUri": "thl://continuent4:2112/",
      "maximumStoredSeqNo": -1,
      "minimumStoredSeqNo": -1,
      "offlineRequests": "NONE",
      "pendingError": "Event application failed: seqno=10 fragno=0 message=Table »
          mats.ct1351 not found in database. Unable to generate a valid statement.",
      "pendingErrorCode": "NONE",
      "pendingErrorEventId": "mysql-bin.000015:0000000000368340;-1",
      "pendingErrorSeqno": 10,
      "pendingExceptionMessage": "Table mats.ct1351 not found in database. Unable »
          to generate a valid statement.",
      "pipelineSource": "UNKNOWN",
      "relativeLatency": -1,
      "resourceJdbcDriver": "org.drizzle.jdbc.DrizzleDriver",
      "resourceJdbcUrl": "jdbc:mysql:thin://continuent4:3306/${DBNAME}? »
          jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull& »
          tinyInt1isBit=false&allowMultiQueries=true&yearIsDateType=false",
      "resourcePrecedence": 99,
      "resourceVendor": "mysql",
      "rmiPort": 10000,
      "role": "slave",
      "seqnoType": "java.lang.Long",
      "serviceName": "east",
      "serviceType": "unknown",
      "simpleServiceName": "east",
      "siteName": "default",
      "sourceId": "continuent4",
      "state": "OFFLINE:ERROR",
      "timeInStateSeconds": 434526.394,
      "timezone": "GMT",
      "transitioningTo": "",
      "uptimeSeconds": 434526.967,
      "useSSLConnection": false,
      "version": "Tungsten Replicator 7.0.3"
    },
    "west": {
      "appliedLastEventId": "mysql-bin.000016:0000000000010112;-1",
      "appliedLastSeqno": 28,
      "appliedLatency": 1.138,
      "autoRecoveryEnabled": false,
      "autoRecoveryTotal": 0,
      "channels": 1,
      "clusterName": "west",
      "currentEventId": "mysql-bin.000016:0000000000010112",
      "currentTimeMillis": 1646836106401,
      "dataServerHost": "continuent4",
      "extensions": "",
      "host": "continuent4",
      "latestEpochNumber": 28,
      "masterConnectUri": "thls://localhost:/",
      "masterListenUri": "thls://continuent4:2112/",
      "maximumStoredSeqNo": 28,
      "minimumStoredSeqNo": 0,
      "offlineRequests": "NONE",
      "pendingError": "NONE",
      "pendingErrorCode": "NONE",
      "pendingErrorEventId": "NONE",
      "pendingErrorSeqno": -1,
      "pendingExceptionMessage": "NONE",
      "pipelineSource": "jdbc:mysql:thin://continuent4:3306/tungsten_west? »
          noPrepStmtCache=true&allowMultiQueries=true&stripQueryComments=false& »
```

```
          enabledProtocols=TLSv1.3,TLSv1.2,TLSv1.1&enabledCipherSuites= »
          TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, »
          TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, »
          TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,TLS_ECDH_RSA_WITH_AES_256_CBC_SHA, »
          TLS_RSA_WITH_AES_256_CBC_SHA,TLS_RSA_WITH_AES_128_CBC_SHA&connectTimeout=15",
      "relativeLatency": 434526.401,
      "resourceJdbcDriver": "org.drizzle.jdbc.DrizzleDriver",
      "resourceJdbcUrl": "jdbc:mysql:thin://continuent4:3306/${DBNAME}? »
          jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull& »
          tinyInt1isBit=false&allowMultiQueries=true&yearIsDateType=false",
      "resourcePrecedence": 99,
      "resourceVendor": "mysql",
      "rmiPort": 10000,
      "role": "master",
      "seqnoType": "java.lang.Long",
      "serviceName": "west",
      "serviceType": "local",
      "simpleServiceName": "west",
      "siteName": "default",
      "sourceId": "continuent4",
      "state": "ONLINE",
      "timeInStateSeconds": 434526.28,
      "timezone": "GMT",
      "transitioningTo": "",
      "uptimeSeconds": 434526.479,
      "useSSLConnection": true,
      "version": "Tungsten Replicator 7.0.3"
    }
  }
}
```


The optional statusType parameter can be added to specify the kind of status. The following options are:

- `channel_assignments`
- `services`
- `shards`
- `stages`
- `stores`
- `tasks`
- `watches`

For example, to view the status of tasks, issue:

```
GET 'https://127.0.0.1:8097/api/v2/replicator/status?statusType=tasks'
```

Which will produce output similar to the following example:


```
{
  "payloadType": "ReplicatorStatusByTypePayload",
  "payloadVersion": "1",
  "payload": {
    "type": "tasks",
    "status": {
      "west": [
        {
          "lastCommittedBlockTime": "1.037",
          "currentLastSeqno": "28",
          "extractTime": "1.034",
          "currentLastEventId": "mysql-bin.000016:0000000000010112;-1",
          "eventCount": "1",
          "filterTime": "0.0",
          "appliedLastSeqno": "28",
          "averageBlockSize": "0.500     ",
          "appliedLastEventId": "mysql-bin.000016:0000000000010112;-1",
          "stage": "binlog-to-q",
          "currentBlockSize": "0",
          "appliedLatency": "1.123",
          "cancelled": "false",
          "commits": "2",
          "otherTime": "0.0",
          "timeInCurrentEvent": "434695.351",
          "currentLastFragno": "0",
          "state": "extract",
          "applyTime": "0.0",
          "taskId": "0",
```

```
        "lastCommittedBlockSize": "1"
      },
      {
        "lastCommittedBlockTime": "1.068",
        "currentLastSeqno": "28",
        "extractTime": "1.034",
        "currentLastEventId": "mysql-bin.000016:0000000000010112;-1",
        "eventCount": "1",
        "filterTime": "0.0",
        "appliedLastSeqno": "28",
        "averageBlockSize": "0.500     ",
        "appliedLastEventId": "mysql-bin.000016:0000000000010112;-1",
        "stage": "q-to-thl",
        "currentBlockSize": "0",
        "appliedLatency": "1.123",
        "cancelled": "false",
        "commits": "2",
        "otherTime": "0.001",
        "timeInCurrentEvent": "434695.336",
        "currentLastFragno": "0",
        "state": "extract",
        "applyTime": "0.015",
        "taskId": "0",
        "lastCommittedBlockSize": "1"
      }
    ],
    "east": []
  }
 }
}
```


11.4.1.3. version

This `GET` request returns the running version of the replicator

```
GET 'https://127.0.0.1:8097/api/v2/replicator/version'
```


```
{
  "payloadType": "StringPayload",
  "payloadVersion": "1",
  "payload": {
    "value": "Tungsten Replicator 7.0.3"
  }
}
```


For more precise version of the replicator, another `GET` request can be used

```
GET 'https://127.0.0.1:8097/api/v2/tungstenVersion'
```


```
{
  "payloadType": "VersionPayload",
  "payloadVersion": "1",
  "payload": {
    "productName": "Tungsten Replicator",
    "versionMajor": 7,
    "versionMinor": 0,
    "versionRevision": 0,
    "versionBuild": null,
    "tungstenVersion": "Tungsten Replicator 7.0.3"
  }
}
```


11.4.1.4. offline/online

These endpoints allows putting all services online / offline.

This is a `POST` request sent, for example

```
POST 'https://127.0.0.1:8097/api/v2/replicator/online'
POST 'https://127.0.0.1:8097/api/v2/replicator/offline'
```

This will return a task via a `TaskPayload` as below


```
{
  "payloadType": "TaskPayload",
  "payloadVersion": "1",
  "payload": {
    "taskId": "ecaed8a7-5b57-4a76-a1a5-a2de14d9cc79",
    "state": "in_progress",
    "operation": "OnlineTask"
  }
```

```
}
```

Both operations accept an input payload (`OnlinePayload` / `OfflinePayload`). An example is shown below :

```
{
  "payloadType": "OnlinePayload",
  "payloadVersion": "1",
  "payload": {
    "options": [
      {
        "type":"toHeartbeat",
        "value":"stop"
      }
    ]
  }
}
```

11.4.1.5. purge

This endpoint purges all non-tungsten connections connected to the different services.

This is a `POST` request sent, for example

```
POST 'https://127.0.0.1:8097/api/v2/replicator/purge'
```

A `PurgePayload` can be provided to specify a timeout. This will return a `TaskPayload` as an online call, for example.

11.4.1.6. reset

This endpoint resets all replicator services.

This is a `POST` request sent, for example

```
POST 'https://127.0.0.1:8097/api/v2/replicator/reset'
```

A `ResetPayload` can be provided to specify what should be reseted, for exmaple:

```
{
  "payloadType": "ResetPayload",
  "payloadVersion": "1",
  "payload": {
    "type": "thl"
  }
}
```

Valid values for the type are : `all, db, thl, relay`

This will return a `TaskPayload` as an online call, for example.

11.4.2. Service Endpoints

Services can also be controlled individually through a number of endpoints. Some are the same as the main replicator service endpoints (`service, status, online, offline, purge, reset`), but other are specific to services, for a full list, review the Replicator API Developer Docs

A few examples of a selection of these endpoints follow

11.4.2.1. backupCapabilities

Returns the defined backup capabilities for the service.

This is a `GET` request to

```
GET 'https://127.0.0.1:8097/api/v2/replicator/service/west/backupCapabilities'
```

and returns a `BackupCapabilitiesPayload`, for exmaple:

```
{
  "payloadType": "BackupCapabilitiesPayload",
  "payloadVersion": "1",
  "payload": {
    "storageAgents": [
      "fs"
    ],
    "backupAgents": [
      "mariabackup-full",
      "mariabackup-incremental",
```

```
      "mysqldump",
      "xtrabackup-incremental",
      "xtrabackup-full",
      "mariabackup",
      "xtrabackup"
    ],
    "defaultBackupAgent": "mysqldump",
   "defaultStorageAgent": "fs"
  }
}
```

11.4.2.2. backup / restore

Backups or restores the service.

This is a `POST` request :

```
POST 'https://127.0.0.1:8097/api/v2/replicator/service/west/backup'
or
POST 'https://127.0.0.1:8097/api/v2/replicator/service/west/restore'
```

and a `BackupPayLoad` can be specified to provide different settings :

```
{
  "payloadType": "BackupPayload",
  "payloadVersion": "1",
  "payload": {
    "agentName": "mysqldump",
    "storageName": "fs"
  }
}
```

If no payload is provided, backup will use the default backup and storage agents, as shown by `backupCapabilities`, while restore will use the last available backup of the service.

11.4.2.3. setrole

Changes the role of the replicator service.

This is a `POST` request

```
POST 'https://127.0.0.1:8097/api/v2/replicator/service/west/setrole'
```

A payload is mandatory to specify which should be the new role, for example:

```
{
  "payloadType": "SetRolePayload",
  "payloadVersion": "1",
  "payload": {
    "role": "primary"
  }
}
```

Valid roles are : `primary, replica, relay, archive, thl_server, thl_client, direct`

For other calls, refer to the Replicator API Developer documentation.

11.4.3. Service THL Endpoints

A few endpoints are provided to get information about the service THL or to manipulate it, these are as follows:

- `index`
- `info`
- `encryption`
- `compression`
- `genkey`
- `inflight-protocol`

These `GET` requests provide information about THL, for example to view THL Metadata (Equivalent to isssuing thl info, the following GET call can be used. An example of the output follows:

`info` : THL metadata

```
GET 'https://127.0.0.1:8097/api/v2/replicator/service/west/thl/info'
{
    "payloadType": "THLInfoPayload",
    "payloadVersion": "1",
    "payload": {
        "logDirectory": "/opt/continuent/thl/europe/",
        "logFiles": 1,
        "logSizeinMB": 10.775296,
        "minSeqno": 0,
        "maxSeqno": 3600,
        "events": 3600,
        "oldestFile": {
            "name": "thl.data.0000000001",
            "lastModified": 1742995271000,
            "sizeInMB": 10.775296
        },
        "newestFile": {
            "name": "thl.data.0000000001",
            "lastModified": 1742995271000,
            "sizeInMB": 10.775296
        }
    }
}
```

Other valid GET calls are as follows:

`index` : THL files index

```
GET 'https://127.0.0.1:8097/api/v2/replicator/service/west/thl/index'
```

`encryption` : encryption state of THL (enabled or not)

```
GET 'https://127.0.0.1:8097/api/v2/replicator/service/west/thl/encryption'
```

`compression` : compression state of THL (enabled or not)

```
GET 'https://127.0.0.1:8097/api/v8/replicator/service/west/thl/compression'
```

For more detail please refer to the Replicator Developer Documentation.

11.4.3.1. compression / encryption

When used with a `POST` request, this will turn on or off either compression or encryption.

```
POST 'https://127.0.0.1:8097/api/v2/replicator/service/west/thl/encryption'
```

Here is an example to turn encryption on :

```
{
  "payloadType": "BooleanPayload",
  "payloadVersion": "1",
  "payload": {
    "value": true
  }
}
```

The service has to be offline to turn compression / encryption on or off.

11.4.3.2. genkey

This is a `POST` request that will ask the replicator to generate a new encryption key for the THL.

The service has to be online to generate a new THL key.

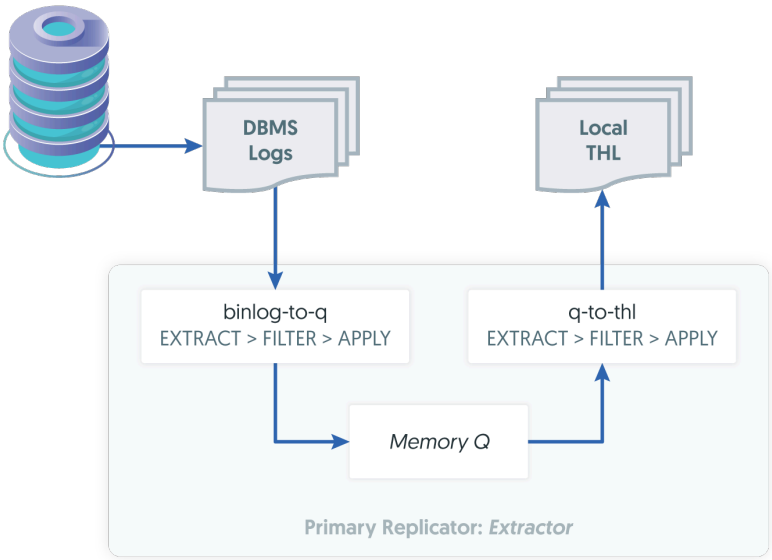
Chapter 12. Replication Filters

Filtering operates by applying the filter within one, or more, of the stages configured within the replicator. Stages are the individual steps that occur within a pipeline, that take information from a source (such as MySQL binary log) and write that information to an internal queue, the transaction history log, or apply it to a database. Where the filters are applied ultimately affect how the information is stored, used, or represented to the next stage or pipeline in the system.

For example, a filter that removed out all the tables from a specific database would have different effects depending on the stage it was applied. If the filter was applied on the Extractor before writing the information into the THL, then no Applier could ever access the table data, because the information would never be stored into the THL to be transferred to the Targets. However, if the filter was applied on the Applier, then some Appliers could replicate the table and database information, while other Appliers could choose to ignore them. The filtering process also has an impact on other elements of the system. For example, filtering on the Extractor may reduce network overhead, albeit at a reduction in the flexibility of the data transferred.

In a standard replicator configuration with MySQL, the following stages are configured in the Extractor, as shown in Figure 12.1, "Filters: Pipeline Stages on Extractors".

Figure 12.1. Filters: Pipeline Stages on Extractors



Where:

- `binlog-to-q` Stage

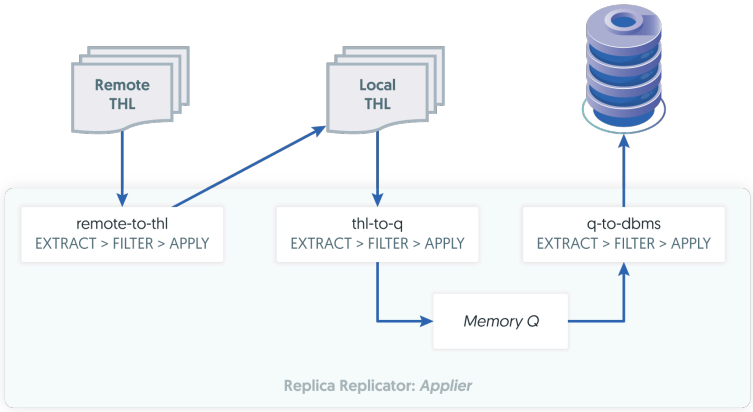
  The `binlog-to-q` stage reads information from the MySQL binary log and stores the information within an in-memory queue.

- `q-to-thl` Stage

  The in-memory queue is written out to the THL file on disk.

Within the Applier, the stages configured by default are shown in Figure 12.2, "Filters: Pipeline Stages on Appliers".

Figure 12.2. Filters: Pipeline Stages on Appliers



- `remote-to-thl` Stage

  Remote THL information is read from an trext; datasource and written to a local file on disk.

- `thl-to-q` Stage

  The THL information is read from the file on disk and stored in an in-memory queue.

- `q-to-dbms` Stage

  The data from the in-memory queue is written to the target database.

Filters can be applied during any configured stage, and where the filter is applied, alters the content and availability of the information. The staging and filtering mechanism can also be used to apply multiple filters to the data, altering content when it is read and when it is applied.

Where more than one filter is configured for a pipeline, each filter is executed in the order it appears in the configuration. For example, within the following fragment:

```
...
replicator.stage.binlog-to-q.filters=settostring,enumtostring,pkey,colnames
...
```

`settostring` is executed first, followed by `enumtostring`, `pkey` and finally `colnames`.

For certain filter combinations this order can be significant. Some filters rely on the information provided by earlier filters.

12.1. 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` [515]

  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` [518]

  Apply the filter between the internal queue and when the transactions are written to the THL on the Extractor. (`q-to-thl`).

- `repl-svc-remote-filters` [516]

  Apply the filter between reading from the remote THL server and writing to the local THL files on the Applier (`remote-to-thl`).

- `repl-svc-applier-filters` [515]

  Apply the filter between reading from the internal queue and applying to the destination database (`q-to-dbms`).

Properties and options for an individual filter can be specified by setting the corresponding property value on the tpm command-line.

For example, to ignore a database schema on a Applier, the `replicate` filter can be enabled, and the *`replicator.filter.replicate.ignore`* specifies the name of the schemas to be ignored. To ignore the schema `contacts`:

Update your `/etc/tungsten/tungsten.ini` as shown in the example below, followed by issuing tpm update:

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

[servicename]
...
repl-svc-applier-filters=replicate
property=replicator.filter.replicate.ignore=contacts
...

shell> tpm update
```

A bad filter configuration will not stop the replicator from starting, but the replicator will be placed into the `OFFLINE` state.

To disable a previously enabled filter, remove (or comment out) the values from you `/etc/tungsten/tungsten.ini` file, and issue tpm update

Multiple filters can be applied on any stage, and the filters will be processed and called within the order defined within the configuration. For example, with the following configuration:

```
[servicename]
...
repl-svc-applier-filters=enumtostring,settostring,pkey
...
```

The filters are called in order:

1. `enumtostring`
2. `settostring`
3. `pkey`

The order and sequence can be important if operations are being performed on the data and they are relied on later in the stage. For example, if data is being filtered by a value that exists in a `SET` column within the source data, the `settostring` filter must be defined before the data is filtered, otherwise the actual string value will not be identified.

Warning

In some cases, the filter order and sequence can also introduce errors. For example, when using the `pkey` filter and the `optimizeupdates` filters together, `pkey` may remove KEY information from the THL before `optimizeupdates` attempts to optimize the ROW event, causing the filter to raise a failure condition.

The currently active filters can be determined by using the trepctl status -name stages [389] command:

```
shell> trepctl status -name stages
Processing status command (stages)...
...
NAME                 VALUE
----                 -----
applier.class      : com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier
applier.name       : dbms
blockCommitRowCount: 10
committedMinSeqno  : 3600
extractor.class    : com.continuent.tungsten.replicator.thl.THLParallelQueueExtractor
extractor.name     : parallel-q-extractor
filter.0.class     : com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter
filter.0.name      : mysqlsessions
filter.1.class     : com.continuent.tungsten.replicator.filter.PrimaryKeyFilter
filter.1.name      : pkey
filter.2.class     : com.continuent.tungsten.replicator.filter.BidiRemoteSlaveFilter
filter.2.name      : bidiSlave
name               : q-to-dbms
processedMinSeqno  : -1
taskCount          : 5
Finished status command (stages)...
```

The above output is from a standard Applier replication installation showing the default filters enabled. The filter order can be determined by the number against each filter definition.

12.2. Enabling Additional Filters

The Tungsten Replicator configuration includes a number of filter configurations by default. However, not all filters are given a default configuration, and for some filters, multiple configurations may be needed to achieve more complex filtering requirements. Internally, filter configuration is defined through a property file that defines the filter name and corresponding parameters.

For example, the `rename` configuration is defined as follows:

```
replicator.filter.rename=com.continuent.tungsten.replicator.filter.RenameFilter
replicator.filter.rename.definitionsFile=${replicator.home.dir}/samples/extensions/java/rename.csv
```

The first line creates a new filter configuration using the corresponding Java class. In this case, the filter is named `rename`, as defined by the string *`replicator.filter.rename`*.

Configuration parameters for the filter are defined as values after the filter name. In this example, *`definitionsFile`* is the name of the property examined by the class to set the CSV file where the rename definitions are located.

To create an entirely new filter based on an existing filter class, a new property should created with the new filter definition in the configuration file.

Additional properties from this base should then be used. For example, to create a second rename filter definition called `custom`:

```
replicator.filter.rename.custom=com.continuent.tungsten.replicator.filter.RenameFilter
replicator.filter.rename.custom.definitionsFile=${replicator.home.dir}/samples/extensions/java/renamecustom.csv
```

The filter can be enabled against the desired stage using the filter name `custom`:

```
[servicename]
...
repl-svc-applier-filters=custom
...
```

12.3. Filter Reference

The different filter types configured and available within Tungsten Replicator are designed to provide a number of different functionality and operations. Since the information exchanged through the THL system contains a copy of the statement or the row data that is being updated, the filters allow schemas, table and column names, as well as actual data to be converted at the stage in which they are applied.

Filters are identified according to the underlying Java class that defines their operation. For different filters, further configuration and naming is applied according to the templates used when Tungsten Cluster is installed through tpm.

Tungsten Replicator also comes with a number of JavaScript filters that can either be used directly, or that can be modified and adapted to suit individual requirements. These filter scripts are located in `tungsten-replicator/support/filters-javascript`.

For the purposes of classification, the different filters have been categorised according to their main purpose:

- Auditing

  These filters provide methods for tracking database updates alongside the original table data. For example, in a financial database, the actual data has to be updated in the corresponding tables, but the individual changes that lead to that update must also be logged individually.

- Content

  Content filters modify or update the content of the transaction events. These may alter information, for the purposes of interoperability (such as updating enumerated or integer values to their string equivalents), or remove or filter columns, tables, and entire schemas.

- Logging

  Logging filters record information about the transactions into the standard replicator log, either for auditing or debugging purposes.

- Optimization

  The optimization filters are designed to simplify and optimize statements and row updates to improve the speed at which those updates can be applied to the destination dataserver.

- Transformation

  Transformation filters rename or reformat schemas and tables according to a set of rules. For example, multiple schemas can be merged to a single schema, or tables and column names can be updated

- Validation

  Provide validation or consistency checking of either the data or the replication process.

- Miscellaneous

Other filters that cannot be allocated to one of the existing filter classes.

In the following reference sections:

- Pre-configured filter name is the filter name that can be used against a stage without additional configuration.
- Property prefix is the prefix string for the filter to be used when assigning property values.
- Classname is the Java class name of the filter.
- Parameter is the name of the filter parameter can be set as a property within the configuration.
- Data compatibility indicates whether the filter is compatible with row-based events, statement-based events, or both.

12.3.1. `ansiquotes.js` Filter

The `ansiquotes` filter operates by inserting an SQL mode change to `ANSI_QUOTES` into the replication stream before a statement is executed, and returning to an empty SQL mode.

Pre-configured filter name		`ansiquotes`	
JavaScript Filter File		`tungsten-replicator/support/filters-javascript/ansiquotes.js`	
Property prefix		*`replicator.filter.ansiquotes`*	
Stage compatibility		`binlog-to-q`	
tpm Option compatibility		`--svc-extractor-filters` [515]	
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);
```

12.3.2. BidiRemoteSlave (BidiSlave) Filter

The BidiRemoteSlaveFilter is used by Tungsten Replicator to prevent statements that originated from this service (i.e. where data was extracted), being re-applied to the database. This is a requirement for replication to prevent data that may be transferred between hosts being re-applied, particularly in Active/Active and other bi-directional replication deployments.

Pre-configured filter name	`bidiSlave`

Classname		com.continuent.tungsten.replicator.filter.BidiRemoteSlaveFilter	
Property prefix		*replicator.filter.bidiSlave*	
Stage compatibility			
tpm Option compatibility			
Data compatibility		Any event	
Parameters			
Parameter	Type	Default	Description
localServiceName	string	*${local.service.name}*	Local service name of the service that reads the binary log
allowBidiUnsafe	boolean	false	If true, allows statements that may be unsafe for bi-directional replication
allowAnyRemoteService	boolean	false	If true, allows statements from any remote service, not just the current service

The filter works by comparing the server ID of the THL event that was created when the data was extracted against the server ID of the current server.

When deploying through the tpm service the filter is automatically enabled for remote Appliers. For complex deployments, particularly those with bi-directional replication (including active/active), the *allowBidiUnsafe* parameter may need to be enabled to allow certain statements to be re-executed.

12.3.3. breadcrumbs.js Filter

The breadcrumbs filter records regular 'breadcrumb' points into a MySQL table for systems that do not have global transaction IDs. This can be useful if recovery needs to be made to a specific point. The example also shows how metadata information for a given event can be updated based on the information from a table.

Pre-configured filter name		ansiquotes	
JavaScript Filter File		tungsten-replicator/support/filters-javascript/breadcrumbs.js	
Property prefix		*replicator.filter.breadcrumbs*	
Stage compatibility		binlog-to-q	
tpm Option compatibility		--svc-extractor-filters [515]	
Data compatibility		Any event	
Parameters			
Parameter	Type	Default	Description
server_id	numeric	(none)	MySQL server ID of the current host

To use the filter:

1. A table is created and populated with one more rows on the Target server. For example:

```
CREATE TABLE `tungsten_svc1`.`breadcrumbs` (
 `id` int(11) NOT NULL PRIMARY KEY,
 `counter` int(11) DEFAULT NULL,
 `last_update` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP) ENGINE=InnoDB;
INSERT INTO tungsten_svc1.breadcrumbs(id, counter) values(@@server_id, 1);
```

2. Now set an event to update the table regularly. For example, within MySQL an event can be created for this purpose:

```
CREATE EVENT breadcrumbs_refresh
  ON SCHEDULE EVERY 5 SECOND
  DO
    UPDATE tungsten_svc1.breadcrumbs SET counter=counter+1;
SET GLOBAL event_scheduler = ON;
```

The filter will extract the value of the counter each time it sees to the table, and then mark each transaction with a particular server ID with the counter value plus an offset. For convenience we assume row replication is enabled.

If you need to failover to another server that has different logs, you can figure out the restart point by looking in the THL for the breadcrumb metadata on the last transaction. Use this to search the binary logs on the new server for the correct restart point.

The filter itself work in two stages, and operates because the JavaScript instance is persistent as long as the Replicator is running. This means that data extracted during replication stays in memory and can be applied to later transactions. Hence the breadcrumb ID and offset information can be identified and used on each call to the filter function.

The first part of the filter event identifies the breadcrumb table and extracts the identified breadcrumb counter:

```
if (table.compareToIgnoreCase("breadcrumbs") == 0)
{
  columnValues = oneRowChange.getColumnValues();
  for (row = 0; row < columnValues.size(); row++)
  {
    values = columnValues.get(row);
    server_id_value = values.get(0);
    if (server_id == null || server_id == server_id_value.getValue())
    {
      counter_value = values.get(1);
      breadcrumb_counter = counter_value.getValue();
      breadcrumb_offset = 0;
    }
  }
}
```

The second part updates the event metadata using the extracted breadcrumb information:

```
topLevelEvent = event.getDBMSEvent();
if (topLevelEvent != null)
{
  xact_server_id = topLevelEvent.getMetadataOptionValue("mysql_server_id");
  if (server_id == xact_server_id)
  {
    topLevelEvent.setMetaDataOption("breadcrumb_counter", breadcrumb_counter);
    topLevelEvent.setMetaDataOption("breadcrumb_offset", breadcrumb_offset);
  }
}
```

To calculate the offset (i.e. the number of events since the last breadcrumb value was extracted), the filter determines if the event was the last fragment processed, and updates the offset counter:

```
if (event.getLastFrag())
{
  breadcrumb_offset = breadcrumb_offset + 1;
}
```

12.3.4. CaseTransform Filter

The `CaseTransform` filter can be used to force convert Schema, Table and Column names to either upper or lower case.

Pre-configured filter name		`casetransform`	
Classname		`com.continuent.tungsten.replicator.filter.CaseMappingFilter`	
Property prefix		*`replicator.filter.casetransform`*	
Stage compatibility			
tpm Option compatibility			
Data compatibility		Any Event	
Parameters			
Parameter	Type	Default	Description
`to_upper_case`	`boolean`	`true`	If true, converts object names to upper case; if false converts them to lower case

This filter can be useful when replicating between environments that have different case sensitivity settings in place.

Uage Example

To force Upper Case on extractor:

```
svc-extractor-filters=casetransform
property=replicator.filter.casetransform.to_upper_case=true
```

To force Lower Case on applier:

```
svc-applier-filters=casetransform
```

```
property=replicator.filter.casetransform.to_upper_case=false
```

12.3.5. ColumnName Filter

The `ColumnNameFilter` loads the table specification information for tables and adds this information to the THL data for information extracted using row-base replication.

Pre-configured filter name		`colnames`	
Classname		`com.continuent.tungsten.replicator.filter.ColumnNameFilter`	
Property prefix		*`replicator.filter.colnames`*	
Stage compatibility		`binlog-to-q`	
tpm Option compatibility		`--svc-extractor-filters` [515]	
Data compatibility		Row events	
Keeps Cached Data		Yes	
Cached Refreshed When?		Emptied when going `OFFLINE`; Updated when `ALTER` statement seen	
Metadata Updated		Yes; `tungsten_filter_columnname=true`	
Parameters			
Parameter	Type	Default	Description
`user`	string	*`${replicator.global.extract.db.user}`*	The username for the connection to the database for looking up column definitions
`password`	string	*`${replicator.global.extract.db.password}`*	The password for the connection to the database for looking up column definitions
`url`	string	`jdbc:mysql:thin://${replicator.global.extract.db.host}: » ${replicator.global.extract.db.port}/${replicator.schema}?createDB=true`	JDBC URL of the database connection to use for looking up column definitions
`addSignedFlag`	`boolean`	`true`	Determines whether the signed flag information for columns should be added to the metadata for each column.
`ignoreMissingTables`	`boolean`	`true`	When true, tables that do not exist will not trigger metadata and column names to be added to the THL data.

Note

This filter is designed to be used for testing and with heterogeneous replication where the field name information can be used to construct and build target data structures.

The filter is required for the correct operation of heterogeneous replication, for example when replicating to MongoDB. The filter works by using the replicator username and password to access the underlying database and obtain the table definitions. The table definition information is cached within the replication during operation to improve performance.

When extracting data from thew binary log using row-based replication, the column names for each row of changed data are added to the THL.

Enabling this filter changes the THL data from the following example, shown without the column names:

```
SEQ# = 27 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 18:29:38.0
- EPOCH# = 11
- EVENTID = mysql-bin.000012:0000000000004369;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = test
 - TABLE = sales
 - ROW# = 0
  - COL(1: ) = 1
  - COL(2: ) = 23
  - COL(3: ) = 45
  - COL(4: ) = 45000.00
```

To a version where the column names are included as part of the THL record:

```
SEQ# = 43 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 18:34:18.0
- EPOCH# = 28
- EVENTID = mysql-bin.000012:0000000000006814;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = test
 - TABLE = sales
 - ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 23
  - COL(3: city) = 45
  - COL(4: value) = 45000.00
```

When the row-based data is applied to a non-MySQL database the column name information is used by the applier to specify the column, or they key when the column and value is used as a key/value pair in a document-based store.

12.3.6. ConvertStringFromMySQL Filter

The `ConvertStringFromMySQLFilter` is designed to be used in replicators that are used in conjunction either with existing native MySQL to MySQL replication deployments, or clustering deployments where the replication has been configured to use native MySQL byte storage for strings. These are incompatible with heterogeneous deployments as the string is stored internally and in the THL in a format that is useful only within similarly configured replicators.

Conversion can be selected to happen for all valid columns (`VARCHAR` or `CHAR` column types only), or for selected columns within specific tables and schemas. All conversions are made with the relevant character set for the table and THL event.

Note

Conversion will not occur on incompatible columns. For example, conversion will not be applied to `INT` columns. This is the case even if the column has been explicitly set to convert the column.

Pre-configured filter name	*convertstringfrommysql*		
Classname	`com.continuent.tungsten.replicator.ConvertStringFromMySQLFilter`		
Property prefix	*Not defined*		
Stage compatibility	any		
tpm Option compatibility			
Data compatibility	Row events only		
Parameters			
definitionsFile	string	`support/filters-config/convertstringfrom-mysql.json`	JSON file containing the definition of which events and which tables to skip

Configuration of the filter is made using the generic JSON file, which supports both default options to happen for all tables not otherwise explicitly specified. The default JSON file converts all valid (`VARCHAR` or `CHAR`) column types only:

```
{
    "__default": {
        "*" : "true",
    },
    "SCHEMA" : {
        "TABLE" : {
            "COLUMN" : "true",
        },
    }
}
```

Warning

For column specific selection to work, the column names must be included within the THL. The `colnames` filter must have been enabled either before this filter, or on the extractor where the data was originally extracted.

The default section handles the default response when an explicit schema or table name does not appear. Further sections are then organised by schema, table and column name. Where the setting is `true`, conversion will take place. A `false` disables conversion.

To enable conversion on a single column `DESCRIPTION` within the SALES.INVOICE schema/table while disabling conversion on all other columns:

```
{
    "__default": {
        "*" : "false",
    },
    "SALES" : {
        "INVOICE" : {
            "DESCRIPTION" : "true",
        },
    }
}
```

To convert all compatible columns in all tables within a schema:

```
{
    "__default": {
        "*" : "false",
    },
    "SALES" : {
        "*" : {
            "*" : "true",
        },
    }
}
```

A primary use case for this filter is for Cluster-Extractor replication from a cluster to a datawarehouse. For more details, please see Section 3.9, “Replicating Data Out of a Cluster”.

Source Cluster Example

For Cluster-Extractor replication to a datawarehouse, the source cluster nodes must use ROW-based MySQL binary logging, and also must have two extractor filters enabled, `colnames` and `pkey`.

For example, on every cluster node the lines below would be added to the `/etc/tungsten/tungsten.ini`> file in the service stanza, then tpm update would be executed:

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

For staging deployments, prepend two hyphens to each line and include on the command line.

For more details about configuring the source cluster, please see Section 3.9, “Replicating Data Out of a Cluster”.

Target Cluster-Extractor Example

On the replication Applier node, copy the `convertstringfrommysql.json` filter configuration sample file into the `/opt/continuent/share` directory then edit it to suit:

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

Once the `convertstringfrommysql` JSON configuration file has been edited, update the `/etc/tungsten/tungsten.ini` file to add and configure the convertstringfrommysql filter.

For example, comfigure a service named `omega` on `host6` to read from the cluster nodes defined by cluster-alias `alpha`.

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

[omega]
topology=cluster-slave
relay=host6
relay-source=alpha
repl-svc-remote-filters=convertstringfrommysql
property=replicator.filter.convertstringfrommysql.definitionsFile=/opt/replicator/share/convertstringfrommysql.json
```

For more details about configuring the target Cluster-Extractor node, please see Section 3.9, “Replicating Data Out of a Cluster”.

12.3.7. DatabaseTransform (dbtransform) Filter

This filter can be used to rename databases (schemas) and/or tables between source and targets

Pre-configured filter name		dbtransform	
Classname		com.continuent.tungsten.replicator.filter.DatabaseTransformFilter	
Property prefix		*replicator.filter.dbtransform*	
Stage compatibility			
tpm Option compatibility			
Data compatibility		ROW Events only	
Parameters			
Parameter	Type	Default	Description
transformTables	boolean	false	If set to true, forces the rename transformations to operate on tables, not databases
from_regex1	string	foo	The search regular expression to use when renaming databases or tables (group 1); corresponds to *to_regex1*
to_regex1	string	bar	The replace regular expression to use when renaming databases or tables (group 1); corresponds to *from_regex1*
from_regex2	string		The search regular expression to use when renaming databases or tables (group 2); corresponds to *to_regex2*
to_regex2	string		The replace regular expression to use when renaming databases or tables (group 2); corresponds to *from_regex2*
from_regex3	string		The search regular expression to use when renaming databases or tables (group 3); corresponds to *to_regex3*
to_regex3	string		The replace regular expression to use when renaming databases or tables (group 3); corresponds to *from_regex3*
from_regex4	string		The search regular expression to use when renaming databases or tables (group 4); corresponds to *to_regex4*
to_regex4	string		The replace regular expression to use when renaming databases or tables (group 4); corresponds to *from_regex4*

The dbtransoform filter can be used to apply standard Java Regex expressions to rename databases and/or tables between source and target.

Up to 4 from/to regex patterns can be provided. By default the transofrmation will be applied to Database Schema names. To use the transofrm for Table Names, specify the transformTables=true option.

The filter will only transform database or tables, not a mix of the two. For more advanced transofrmation you may want to consider the rename filter instead

The filter only wors with ROW events. Statement based transofrmation are not supported with this filter.

12.3.8. dbrename.js Filter

The dbrename JavaScript filter renames database (schemas) using two parameters from the properties file, the *dbsource* and *dbtarget*. Each event is then processed, and the statement or row based schema information is updated to *dbtarget* when the *dbsource* schema is identified.

Pre-configured filter name		dbrename	
JavaScript Filter File		tungsten-replicator/support/filters-javascript/dbrename.js	
Property prefix		*replicator.filter.dbrename*	
Stage compatibility		binlog-to-q	
tpm Option compatibility		--svc-extractor-filters [515]	
Data compatibility		Any event	
Parameters			
Parameter	Type	Default	Description
dbsource	string	(none)	Source table name (database to be renamed)

dbtarget	string	(none)	New database 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);
                }
            }
        }
    }
}
```

12.3.9. `dbselector.js` Filter

Filtering only a single database schema can be useful when you want to extract a single schema for external processing, or for sharding information across multiple replication targets. The `dbselector` filter deletes all statement and row changes, except those for the selected table. To configure, the *db* parameter to the filter configuration specifies the schema to be replicated.

Pre-configured filter name		`dbselector`	
JavaScript Filter File		`tungsten-replicator/support/filters-javascript/dbselector.js`	
Property prefix		*replicator.filter.dbselector*	
Stage compatibility		`binlog-to-q`, `q-to-thl`, `q-to-dbms`	
tpm Option compatibility		`--svc-extractor-filters` [515], `--svc-applier-filters` [515]	
Data compatibility		Any event	
Parameters			
Parameter	Type	Default	Description
db	string	(none)	Database to be selected

Within the filter, statement changes look for the schema in the `StatementData` object and remove it from the array:

```
if (d instanceof com.continuent.tungsten.replicator.dbms.StatementData)
{
    if(d.getDefaultSchema().compareTo(db)!=0)
    {
```

```
        data.remove(i);
        i--;
    }
}
```

Because entries are being removed from the list of statements, the iterator used to process each item must be explicitly decremented by 1 to reset the counter back to the new position.

Similarly, when looking at row changes in the `RowChangeData`:

```
else if(d instanceof com.continuent.tungsten.replicator.dbms.RowChangeData)
{
    rowChanges = data.get(i).getRowChanges();

    for(j=0;j<rowChanges.size();j++)
    {
        oneRowChange = rowChanges.get(j);

        if(oneRowChange.getSchemaName().compareTo(db)!=0)
        {
            rowChanges.remove(j);
            j--;
        }
    }
}
```

12.3.10. `dbupper.js` Filter

The `dbupper` filter changes the case of the schema name for all schemas to uppercase. The schema information is easily identified in the statement and row based information, and therefore easy to update.

Pre-configured filter name		`dbupper`	
JavaScript Filter File		`tungsten-replicator/support/filters-javascript/dbupper.js`	
Property prefix		*`replicator.filter.dbupper`*	
Stage compatibility		`binlog-to-q`	
tpm Option compatibility		`--svc-extractor-filters` [515], `--svc-applier-filters` [515]	
Data compatibility		Any event	
Parameters			
Parameter	Type	Default	Description
`from`	`string`	(none)	Database name to be converted to uppercase

For example, within statement data:

```
from = d.getDefaultSchema();
if (from != null)
{
    to   = from.toUpperCase();
    d.setDefaultSchema(to);
}
```

12.3.11. `dropcolumn.js` Filter

The `dropcolumn` filter enables columns in the THL to be dropped. This can be useful when replicating Personal Identification Information, such as email addresses, phone number, personal identification numbers and others are within the THL but need to be filtered out on the Target.

Pre-configured filter name		`dropcolumn`	
JavaScript Filter File		`tungsten-replicator/support/filters-javascript/dropcolumn.js`	
Property prefix		*`replicator.filter.dropcolumn`*	
Stage compatibility		`binlog-to-q`, `q-to-dbms`	
tpm Option compatibility		`--svc-extractor-filters` [515], `--svc-applier-filters` [515]	
Data compatibility		Any event	
Parameters			
Parameter	Type	Default	Description
`definitionsFile`	Filename	`~/dropcolumn.json`	Location of the definitions file for dropping columns

fillGaps	Boolean	false	When true, re-order THL Column Index IDs to sequential numbers

The filter is available by default as dropcolumn, and the filter is configured through a JSON file that defines the list of columns to be dropped. The filter relies on the colnames filter being enabled.

To enable the filter:

```
shell> tpm update --svc-extractor-filters=colnames,dropcolumn \
    --property=replicator.filter.dropcolumn.definitionsFile=/opt/continuent/share/dropcolumn.json
```

A sample configuration file is provided in tungsten-replicator/support/filters-config/dropcolumn.json. The format of the file is a JSON array of schema/table/column specifications:

```
[
    {
        "schema": "vip",
        "table": "clients",
        "columns": [
            "personal_code",
            "birth_date",
            "email"
        ]
    },
    ...
]
```

Where:

- schema specifies the name of the schema on which to apply the filtering. If * is given, all schemas are matched.
- table specifies the name of the table on which to apply the filtering. If * is given, all tables are matched.
- columns is an array of column names to be matched.

For example:

```
[
    {
        "schema": "vip",
        "table": "clients",
        "columns": [
            "personal_code",
            "birth_date",
            "email"
        ]
    },
    ...
]
```

Filters the columns email, birth_date, and personal_code within the clients table in the vip schema.

To filter the telephone column in any table and any schema:

```
[
    {
        "schema": "*",
        "table": "*",
        "columns": [
            "telephone"
        ]
    }
]
```

Care should be taken when dropping columns on the Target and Source when the column order is different or when the names of the column differ:

- If the column order is same, even if dropcolumn.js is used, leave the default setting for the property *replicator.applier.dbms.getColumnMetaDataFromDB=true*.
- If the column order is different on the Source and Target, set *replicator.applier.dbms.getColumnMetadataFromDB=false*
- If slave's column names are different, regardless of differences in the order, use the default property setting *replicator.applier.dbms.getColumnMetadataFromDB=true*

If the filter is enabled on the extractor, the columns will be removed from the THL and the resulting THL Index will appear to have gaps in the THL column index ID, for example:

```
...
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = sample
 - TABLE = demotable
 - ROW# = 0
  - COL(1: id) = 11
  - COL(2: col_a) = UK
  - COL(4: col_c) = 5678
  - COL(5: col_d) = ABC
  - COL(7: col_g) = 2019-09-05 07:21:48.0
  - KEY(1: id) = NULL
...
```

For JDBC targets, this is expected and required to ensure accurate column mapping, however for Batch Apliers the gap in the Index ID's will cause the applier to fail with a CSV Column Mismatch error, therefore, if your target is a Batch target (eg Hadoop, Redshift, Vertica) you need to add the following property to your configuration:

```
property=replicator.filter.dropcolumn.fillGaps=true
```

With this property in place, the resulting THL from the above example would now look like the following:

```
...
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = sample
 - TABLE = demotable
 - ROW# = 0
  - COL(1: id) = 11
  - COL(2: col_a) = UK
  - COL(3: col_c) = 5678
  - COL(4: col_d) = ABC
  - COL(5: col_g) = 2019-09-05 07:21:48.0
  - KEY(1: id) = NULL
...
```

12.3.12. `dropcomments.js` Filter

The `dropcomments` filter removes comments from statements within the event data.

Pre-configured filter name		`dropcomments`	
JavaScript Filter File		`tungsten-replicator/support/filters-javascript/dropcomments.js`	
Property prefix		*`replicator.filter.dropcomments`*	
Stage compatibility		`binlog-to-q`, `q-to-dbms`	
tpm Option compatibility		`--svc-extractor-filters` [515], `--svc-applier-filters` [515]	
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--;
}
```

12.3.13. `dropddl.js` Filter

Note

Note that this filter is only intended for use with MySQL<>MySQL replication and will not have any benefit at this time for heterogeneous replication.

There may be occasions where you do not require specific DDL statements to be replicated. For example, you may NOT want to allow TRIGGERS or VIEWS to be replicated, but you do want TABLES, INDEXES, FUNCTIONS and PROCEDURES. The dropddl filter is intended for this purpose.

The dropddl filter allows you to configure which, if any, ddl statements to drop from THL and not be replicated.

Pre-configured filter name		dropddl	
JavaScript Filter File		tungsten-replicator/support/filters-javascript/dropddl.js	
Property prefix		*replicator.filter.dropddl*	
Stage compatibility		binlog-to-q, q-to-dbms	
tpm Option compatibility		--svc-extractor-filters [515], --svc-applier-filters [515]	
Data compatibility		Any event	
Parameters			
Parameter	Type	Default	Description
dropTableDDL	boolean	false	Drop CREATE TABLE and DROP TABLE statements
dropViewDDL	boolean	false	Drop CREATE VIEW and DROP VIEWS statements
dropIndexDDL	boolean	false	Drop CREATE INDEX and DROP INDEX statements
dropTriggerDDL	boolean	false	Drop CREATE TRIGGER and DROP TRIGGER statements
dropFunctionDDL	boolean	false	Drop CREATE FUNCTION and DROP FUNCTION statements
dropProcedureDDL	boolean	false	Drop CREATE PROCEDURE and DROP PROCEDURE statements
dropUserDDL	boolean	false	Drop CREATE USER and ALTER USER statements
dropGrantDDL	boolean	false	Drop GRANT and REVOKE statements

Example:

If you wish to drop VIEW and TRIGGER DDL on the applier, the following options need to be added to your configuration:

```
svc-applier-filters=dropddl
property=replicator.filter.dropddl.dropViewDDL=true
property=replicator.filter.dropddl.dropTriggerDDL=true
```

12.3.14. dropmetadata.js Filter

All events within the replication stream contain metadata about each event. This information can be individually processed and manipulated. The dropmetadata filter removes specific metadata from each event, configured through the *option* parameter to the filter.

Pre-configured filter name		dropmetadata	
JavaScript Filter File		tungsten-replicator/support/filters-javascript/dropmetadata.js	
Property prefix		*replicator.filter.ansiquotes*	
Stage compatibility		binlog-to-q, q-to-dbms	
tpm Option compatibility		--svc-extractor-filters [515], --svc-applier-filters [515]	
Data compatibility		Any event	
Parameters			
Parameter	Type	Default	Description
option	string	(none)	Name of the metadata field to be dropped

Metadata information can be processed at the event top-level:

```
metaData = event.getDBMSEvent().getMetadata();
for(m = 0; m < metaData.size(); m++)
{
    option = metaData.get(m);
    if(option.getOptionName().compareTo(optionName)==0)
    {
        metaData.remove(m);
        break;
```

```
    }
}
```

12.3.15. `droprow.js` Filter

The `droprow` filter can be used to selectively filter out transactions based on matching column values at the ROW level.

Pre-configured filter name		`droprow`	
JavaScript Filter File		`tungsten-replicator/support/filters-javascript/droprow.js`	
Property prefix		*`replicator.filter.droprow`*	
Stage compatibility		`binlog-to-q`, `q-to-dbms`	
tpm Option compatibility		`--svc-extractor-filters` [515], `--svc-applier-filters` [515]	
Data compatibility		Row Events	
Parameters			
Parameter	Type	Default	Description
`definitionsFile`	Filename	`~/droprow.json`	Location of the definitions file for row filtering.
`matchCase`	Boolean	`true`	Controls whether text based comparisons are Case Sensitive (`true`) or not (`false`)
`rule`	String	`matchany`	Valid Values : `matchany | matchall`. If more than one column/value pair are defined, this property will determine whether there must be a match for all columns (`matchall`) or only one (`matchany`).

The filter is available by default as `droprow`, and the filter is configured through a JSON file that defines the list of column/values to match in a row to be dropped.

This filter has the following requirements and caveats:

- The filter relies on the colnames filter being enabled.
- The filter will only work on ROW events, therefore the source database needs to be running in Row-Based binary logging mode.
- The filter will only compare values as part of an INSERT satement and NEW values as part of an UPDATE statement.
- Rows that are affected by UPDATE and DELETE statements where the values are part of the WHERE clause will not be removed from THL.
- If the filter is applied to tables with Foreign Keys, be sure to inclue all tables in the hierarchy as part of the filter to avoid referential integrity errors.

To enable the filter, for staging based deployments:

```
shell> tpm update --svc-extractor-filters=colnames,droprow \
    --property=replicator.filter.droprow.definitionsFile=/opt/continuent/share/droprow.json
```

Additional parameters that override the defaults can be also supplied, for example:

```
shell> tpm update --svc-extractor-filters=colnames,droprow \
    --property=replicator.filter.droprow.definitionsFile=/opt/continuent/share/droprow.json \
    --property=replicator.filter.droprow.rule=matchall
```

To enable the filter, for ini based deployments:

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

[servicename]
...
svc-extractor-filters=colnames,droprow
property=replicator.filter.droprow.definitionsFile=/opt/continuent/share/droprow.json
...

shell> tpm update
```

Additional parameters that override the defaults can be also supplied, for example:

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

```
[servicename]
...
svc-extractor-filters=colnames,droprow
property=replicator.filter.droprow.definitionsFile=/opt/continuent/share/droprow.json
property=replicator.filter.droprow.rule=matchall
...

shell> tpm update
```

A sample configuration file is provided in `tungsten-replicator/support/filters-config/droprow.json`. The format of the file is a JSON array of schema/table/column/value specifications. The match rule can also be supplied on a per table basis as per the examples below. If not supplied the global default for the filter, as described above, will be used:

```
[
  {
    "schema": "vip",
    "table": "clients",
    "rule": "matchall",
    "columns": [
      {
        "column": "city",
        "value": "London"
      },
      {
        "column": "country",
        "value": "UK"
      }
    ]
  }
]
```

Where:

- `schema` specifies the name of the schema on which to apply the filtering.
- `table` specifies the name of the table on which to apply the filtering.
- `rule` specifies the matching rule to apply the filtering. Setting the value in the JSON will override the global default specified in the main configuration (See above).

  `matchany`(default) will remove rows if only one condition matches. Equivalent to an OR condition.

  `matchall` will remove rows if only all conditions match. Equivalent to an AND condition.
- `columns` is an array of column/values to be matched.
- `column` is the name of the column in the table to match.
- `value` is the Value of the column to match. When Rule is `matchany` this can also be supplied as an array of Values (See second example below).

For example:

```
[
  {
    "schema": "vip",
    "table": "customers",
    "rule": "matchany",
    "columns": [
      {
        "column": "country",
        "value": [ "Australia", "UK" ]
      }
    ]
  }
]
```

The bove example filters rows from the customers table, in the vip schema where the country column is either "Australia" OR "UK"

12.3.16. `dropstatementdata.js` Filter

Within certain replication deployments, enforcing that only row-based information is replicated is important to ensure that the row data is replicated properly. For example, when replicating to databases that do not accept statements, these events must be filtered out.

Pre-configured filter name	`dropstatementdata`

JavaScript Filter File		tungsten-replicator/support/filters-javascript/dropstatementdata.js	
Property prefix		*replicator.filter.dropstatementdata*	
Stage compatibility		binlog-to-q, q-to-dbms	
tpm Option compatibility		--svc-extractor-filters [515], --svc-applier-filters [515]	
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--;
    }
}
```

12.3.17. dropsqlmode.js Filter

Different releases of MySQL occasionally add/remove various sql_modes, and in doing so if replicating between different versions, you may receive errors during replication as MySQL will reject unknown sql_modes.

This filter was specifically added to handle replication between MySQL 5.7 and MySQL 8 where a number of sql_modes where removed.

Pre-configured filter name		dropsqlmode	
JavaScript Filter File		tungsten-replicator/support/filters-javascript/dropsqlmode.js	
Property prefix		*replicator.filter.dropsqlmode*	
Stage compatibility		q-to-dbms	
tpm Option compatibility		--svc-applier-filters [515]	
Data compatibility		Any event	
Parameters			
Parameter	Type	Default	Description
modes	string	See below	| separated string of sql_modes to drop

By default, the filter is configured to drop the following sql_modes: NO_AUTO_CREATE_USER,NO_FIELD_OPTIONS,NO_KEY_OPTIONS,NO_TABLE_OPTIONS

If you wish to add/remove sql_modes from this list, you will need to override the property as follows:

```
property=replicator.filter.dropsqlmode.modes=NO_AUTO_CREATE_USER|NO_FIELD_OPTIONS|NO_KEY_OPTIONS|NO_TABLE_OPTIONS| TIME_TRUNCATE_FRACTIONAL
```

12.3.18. dropxa.js Filter

The dropxa filter removes XA Transaction Events within the event data.

Pre-configured filter name		dropxa	
JavaScript Filter File		tungsten-replicator/support/filters-javascript/dropxa.js	
Property prefix		*replicator.filter.dropxa*	
Stage compatibility		binlog-to-q	
tpm Option compatibility		--svc-extractor-filters [515]	
Data compatibility		Any event	
Parameters			
Parameter	Type	Default	Description

XA Transaction support was added into the replicator in version 6.1.14 however this filter can still be used if required to remove XA Transaction events if they are not required for replication

12.3.19. Dummy Filter

Pre-configured filter name	dummy
Classname	com.continuent.tungsten.replicator.filter.DummyFilter
Property prefix	*replicator.filter.dummy*
Stage compatibility	
tpm Option compatibility	
Data compatibility	Any event
Parameters	
(none)	

As the name suggests, the dummy filter does nothing, however it can be used as a simple mechanism to ensure that filters load correctly.

12.3.20. EnumToString Filter

The EnumToString filter translates ENUM datatypes within MySQL tables into their string equivalent within the THL.

Pre-configured filter name		enumtostring	
Classname		com.continuent.tungsten.replicator.filter.EnumToStringFilter	
Property prefix		*replicator.filter.enumtostring*	
Stage compatibility		binlog-to-q	
tpm Option compatibility		--repl-svc-extractor-filters [515]	
Data compatibility		Row events	
Metadata Updated		Yes; tungsten_filter_enumtostring=true	
Parameters			
Parameter	Type	Default	Description
user	string	*${replicator.global.extract.db.user}*	The username for the connection to the database for looking up column definitions
password	string	*${replicator.global.extract.db.password}*	The password for the connection to the database for looking up column definitions
url	string	jdbc:mysql:thin://${replicator.global.extract.db.host}: » ${replicator.global.extract.db.port}/${replicator.schema}?createDB=true	JDBC URL of the database connection to use for looking up column definitions
reconnectTimeout	integer	3600	Timeout for refreshing connection to database

The EnumToString filter should be used with heterogeneous replication to ensure that the data is represented as the string value, not the internal numerical representation.

In the THL output below, the table has a ENUM column, country:

```
mysql> describe salesadv;
+----------+-----------------------------------+------+-----+---------+----------------+
| Field    | Type                              | Null | Key | Default | Extra          |
+----------+-----------------------------------+------+-----+---------+----------------+
| id       | int(11)                           | NO   | PRI | NULL    | auto_increment |
| country  | enum('US','UK','France','Australia') | YES  |     | NULL    |                |
| city     | int(11)                           | YES  |     | NULL    |                |
| salesman | set('Alan','Zachary')             | YES  |     | NULL    |                |
| value    | decimal(10,2)                     | YES  |     | NULL    |                |
+----------+-----------------------------------+------+-----+---------+----------------+
```

When extracted in the THL, the representation uses the internal value (for example, 1 for the first enumerated value). This can be seen in the THL output below.

```
SEQ# = 138 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:09:35.0
```

```
- EPOCH# = 122
- EVENTID = mysql-bin.000012:0000000000021434;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = test
 - TABLE = salesadv
 - ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 1
  - COL(3: city) = 8374
  - COL(4: salesman) = 1
  - COL(5: value) = 35000.00
```

For the `country` column, the corresponding value in the THL is `1`. With the `EnumToString` filter enabled, the value is expanded to the corresponding string value:

```
SEQ# = 121 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:05:14.0
- EPOCH# = 102
- EVENTID = mysql-bin.000012:0000000000018866;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = test
 - TABLE = salesadv
 - ROW# = 0
  - COL(1: id) = 1
  - COL(2: country) = US
  - COL(3: city) = 8374
  - COL(4: salesman) = Alan
  - COL(5: value) = 35000.00
```

The information is critical when applying the data to a dataserver that is not aware of the table definition when replicating to non-MySQL target.

The examples here also show the Section 12.3.38, “SetToString Filter” and Section 12.3.5, “ColumnName Filter” filters.

12.3.21. EventMetadata Filter

Filters events based on metadata; used by default within sharding and Active/Active topologies

Pre-configured filter name	`eventmetadata`
Classname	`com.continuent.tungsten.replicator.filter.EventMetadataFilter`
Property prefix	*`replicator.filter.eventmetadata`*
Stage compatibility	
tpm Option compatibility	
Data compatibility	Row events
Metadata Updated	Yes; `tungsten_filter_settostring=true`
Parameters	
(none)	

12.3.22. `foreignkeychecks.js` Filter

The `foreignkeychecks` filter switches off foreign key checks for statements using the following statements:

```
CREATE TABLE
DROP TABLE
ALTER TABLE
RENAME TABLE
```

Pre-configured filter name	`foreignkeychecks`
JavaScript Filter File	`tungsten-replicator/support/filters-javascript/foreignkeychecks.js`

Property prefix		*replicator.filter.foreignkeychecks*	
Stage compatibility		binlog-to-q, q-to-dbms	
tpm Option compatibility		--svc-extractor-filters [515], --svc-applier-filters [515]	
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.

12.3.23. Heartbeat Filter

The heartbeat file detects heartbeat events on Sources or Targets

Pre-configured filter name		[none]	
Classname		com.continuent.tungsten.replicator.filter.HeartbeatFilter	
Property prefix		[none]	
Stage compatibility			
tpm Option compatibility			
Data compatibility		Any event	
Parameters			
Parameter	Type	Default	Description
heartbeatInterval	numeric	3000	Interval in milliseconds when a heartbeat event is inserted into the THL

12.3.24. `insertsonly.js` Filter

The `insertsonly` filter filters events to only include ROW-based events using `INSERT`.

Pre-configured filter name		insertsonly	
JavaScript Filter File		tungsten-replicator/support/filters-javascript/insertonly.js	
Property prefix		*replicator.filter.insertonly*	
Stage compatibility		q-to-dbms	
tpm Option compatibility		--svc-applier-filters [515]	
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--;
}
```

12.3.25. Logging Filter

Logs filtered events through the standard replicator logging mechanism

Pre-configured filter name	`logger`
Classname	`com.continuent.tungsten.replicator.filter.LoggingFilter`
Property prefix	*`replicator.filter.logger`*
Stage compatibility	
tpm Option compatibility	
Data compatibility	Any event
Parameters	
(none)	

12.3.26. MySQLSessionSupport (mysqlsessions) Filter

Filters transactions for session specific temporary tables and variables

Pre-configured filter name	`mysqlsessions`
Classname	`com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter`
Property prefix	*`replicator.filter.mysqlsession`*
Stage compatibility	
tpm Option compatibility	
Data compatibility	Any event
Parameters	
(none)	

12.3.27. mapcharset Filter

Note

This filter was introduced in version 6.1.12

Maps charsets in newer MySQL 8 environments allowing THL to be applied to older versions of MySQL.

Pre-configured filter name	`mapcharset`
Classname	`com.continuent.tungsten.replicator.filter.mapcharset`
Property prefix	*`replicator.filter.mapcharset`*
Stage compatibility	q-to-dbms
tpm Option compatibility	--svc-applier-filters
Data compatibility	Any event
Parameters	
(none)	

Warning

The filter must not be used if applying into MySQL 8 or greater.

- Once a replica has been upgraded to MySQL 8 or greater, the filter MUST be disabled.

The filter utilisies the `tungsten-replicator/support/filters-config/mapcharset.json` to determine the source/target charset mappings.

The json file is a very simple array structure in the format of <SourceCharset>: <TargetCharset>

The default installation of this filter should cover most, if not all, scenarios.

This, however, can be changed by updating `mapcharset.json` file which maps MySQL collation id, as seen by the following mysql query example:

```
mysql> select * from INFORMATION_SCHEMA.COLLATIONS where character_set_name like 'utf8mb4' order by id;

+----------------------------+--------------------+-----+------------+-------------+---------+---------------+
| COLLATION_NAME             | CHARACTER_SET_NAME | ID  | IS_DEFAULT | IS_COMPILED | SORTLEN | PAD_ATTRIBUTE |
+----------------------------+--------------------+-----+------------+-------------+---------+---------------+
| utf8mb4_general_ci         | utf8mb4            |  45 |            | Yes         |       1 | PAD SPACE     |
| utf8mb4_bin                | utf8mb4            |  46 |            | Yes         |       1 | PAD SPACE     |
| utf8mb4_unicode_ci         | utf8mb4            | 224 |            | Yes         |       8 | PAD SPACE     |
| utf8mb4_icelandic_ci       | utf8mb4            | 225 |            | Yes         |       8 | PAD SPACE     |
...
| utf8mb4_0900_ai_ci         | utf8mb4            | 255 | Yes        | Yes         |       0 | NO PAD        |
| utf8mb4_de_pb_0900_ai_ci   | utf8mb4            | 256 |            | Yes         |       0 | NO PAD        |
| utf8mb4_is_0900_ai_ci      | utf8mb4            | 257 |            | Yes         |       0 | NO PAD        |
| utf8mb4_lv_0900_ai_ci      | utf8mb4            | 258 |            | Yes         |       0 | NO PAD        |
| utf8mb4_ro_0900_ai_ci      | utf8mb4            | 259 |            | Yes         |       0 | NO PAD        |
...
| utf8mb4_ru_0900_as_cs      | utf8mb4            | 307 |            | Yes         |       0 | NO PAD        |
| utf8mb4_zh_0900_as_cs      | utf8mb4            | 308 |            | Yes         |       0 | NO PAD        |
| utf8mb4_0900_bin           | utf8mb4            | 309 |            | Yes         |       1 | NO PAD        |
+----------------------------+--------------------+-----+------------+-------------+---------+---------------+
75 rows in set (0.00 sec)
```

For example, the default mapping contains :

```
{
...
  "255": "45"
...
}
```

This means that collation `utf8mb4_0900_ai_ci` is mapped to `utf8mb4_general_ci`.

`tungsten-replicator/support/filters-config/mapcharset.readme` contains a full list of the default mappings.

12.3.28. NetworkClient Filter

The `NetworkClientFilter` processes data in selected columns

Pre-configured filter name	`networkclient`
Classname	`com.continuent.tungsten.replicator.filter.NetworkClientFilter`
Property prefix	*`replicator.filter.networkclient`*
Stage compatibility	Any
tpm Option compatibility	`--svc-extractor-filters` [515], `--svc-thl-filters` [518], `--svc-applier-filters` [515]
Data compatibility	Row events

Parameters

Parameter	Type	Default	Description
`definitionsFile`	pathname	${replicator.home.dir}/samples/extensions/java/network-client.json	The name of a file containing the definitions for how columns should be processed by filters
`serverPort`	number	3112	The network port to use when communicating with the network client
`timeout`	number	10	Timeout in seconds before treating the network client as failed when waiting to send or receive content.

The network filter operates by sending field data, as defined in the corresponding filter configuration file, out to a network server that processes the information and sends it back to be re-introduced in place of the original field data. This can be used to translate and reformat information during the replication scheme.

The filter operation works as follows:

- All filtered data will be sent to a single network server, at the configured port.
- A single network server can be used to provide multiple transformations.
- The JSON configuration file for the filter supports multiple types and multiple column definitions.
- The protocol used by the network filter must be followed to effectively process the information. A failure in the network server or communication will cause the replicator to raise an error and replication to go `OFFLINE`.
- The network server must be running before the replicator is started. If the network server cannot be found, replication will go `OFFLINE`.

Correct operation requires building a suitable network filter using the defined protocol, and creating the JSON configuration file. A sample filter is provided for reference.

12.3.28.1. Network Client Configuration

The format of the configuration file defines the translation operation to be requested from the network client, in addition to the schema, table and column name. The format for the file is JSON, with the top-level hash defining the operation, and an array of field selections for each field that should be processed accordingly. For example:

```
{
   "String_to_HEX_v1" : [
      {
         "table" : "hextable",
         "schema" : "hexdb",
         "columns" : [
            "hexcol"
         ]
      }
   ]
}
```

The operation in this case is `String_to_HEX_v1`; this will be sent to the network server as part of the request. The column definition follows.

To send multiple columns from different tables to the same translation:

```
{
   "String_to_HEX_v1" : [
      {
         "table" : "hextable",
         "schema" : "hexdb",
         "columns" : [
            "hexcol"
         ]
      },
      {
         "table" : "hexagon",
         "schema" : "sourcetext",
         "columns" : [
            "itemtext"
         ]
      }
   ]
}
```

Alternatively, to configure different operations for the same two tables:

```
{
   "String_to_HEX_v1" : [
      {
         "table" : "hextable",
         "schema" : "hexdb",
         "columns" : [
            "hexcol"
         ]
      }
   ],
   "HEX_to_String_v1" : [
      {
         "table" : "hexagon",
         "schema" : "sourcetext",
         "columns" : [
            "itemtext"
         ]
      }
   ]
```

```
}
```

12.3.28.2. Network Filter Protocol

The network filter protocol has been designed to be both lightweight and binary data compatible, as it is designed to work with data that may be heavily encoded, binary, or compressed in nature.

The protocol operates through a combined JSON and optional binary payload structure that communicates the information. The JSON defines the communication type and metadata, while the binary payload contains the raw or translated information.

The filter communicates with the network server using the following packet types:

- prepare

  The `prepare` message is called when the filter goes online, and is designed to initialize the connection to the network server and confirm the supported filter types and operation. The format of the connection message is:

  ```
  {
     "payload" : -1,
     "type" : "prepare",
     "service" : "firstrep",
     "protocol" : "v0_9"
  }
  ```

  Where:

  - `protocol`

    The protocol version.

  - `service`

    The name of the replicator service that called the filter.

  - `type`

    The message type.

  - `payload`

    The size of the payload; a value of -1 indicates that there is no payload.

  The format of the response should be a JSON object and payload with the list of supported filter types in the payload section. The payload immediately follows the JSON, with the size of the list defined within the `payload` field of the returned JSON object:

  ```
  {
     "payload" : 22,
     "type" : "acknowledged",
     "protocol" : "v0_9",
     "service" : "firstrep",
     "return" : 0
  }Perl_BLOB_to_String_v1
  ```

  Where:

  - `protocol`

    The protocol version.

  - `service`

    The name of the replicator service that called the filter.

  - `type`

    The message type; when acknowledging the original prepare request it should be `acknowledge`.

  - `return`

    The return value. A value of 0 (zero) indicates no faults. Any true value indicates there was an issue.

  - `payload`

    The length of the appended payload information in bytes. This is used by the filter to identify how much additional data to read after the JSON object has been read.

The payload should be a comma-separated list of the supported transformation types within the network server.

- filter

  The `filter` message type is sent by Tungsten Replicator for each value from the replication stream that needs to be filtered and translated in some way. The format of the request is a JSON object with a trailing block of data, the payload, that contains the information to be filtered. For example:

```
{
   "schema" : "hexdb",
   "transformation" : "String_to_HEX_v1",
   "service" : "firstrep",
   "type" : "filter",
   "payload" : 22,
   "row" : 0,
   "column" : "hexcol",
   "table" : "hextable",
   "seqno" : 145196,
   "fragments" : 1,
   "protocol" : "v0_9",
   "fragment" : 1
}48656c6c6f20576f726c64
```

  Where:

  - `protocol`

    The protocol version.

  - `service`

    The service name the requested the filter.

  - `type`

    The message type, in this case, `filter`.

  - `row`

    The row of the source information from the THL that is being filtered.

  - `schema`

    The schema of the source information from the THL that is being filtered.

  - `table`

    The table of the source information from the THL that is being filtered.

  - `column`

    The column of the source information from the THL that is being filtered.

  - `seqno`

    The sequence number of the event from the THL that is being filtered.

  - `fragments`

    The number of fragments in the THL that is being filtered.

  - `fragment`

    The fragment number within the THL that is being filtered. The fragments may be sent individually and sequentially to the network server, so they may need to be retrieved, merged, and reconstituted depending on the nature of the source data and the filter being applied.

  - `transformation`

    The transformation to be performed on the supplied payload data. A single network server can support multiple transformations, so this information is provided to perform the corrupt operation. The actual transformation to be performed is taken from the JSON configuration file for the filter.

  - `payload`

The length, in bytes, of the payload data that will immediately follow the JSON filter request..

The payload that immediately follows the JSON block is the data from the column that should be processed by the network filter.

The response package should contain a copy of the supplied information from the requested filter, with the `payload` size updated to the size of the returned information, the message type changed to `filtered`, and the payload containing the translated data. For example:

```
{
   "transformation" : "String_to_HEX_v1",
   "fragments" : 1,
   "type" : "filtered",
   "fragment" : 1,
   "return" : 0,
   "seqno" : 145198,
   "table" : "hextable",
   "service" : "firstrep",
   "protocol" : "v0_9",
   "schema" : "hexdb",
   "payload" : 8,
   "column" : "hexcol",
   "row" : 0
}FILTERED
```

12.3.28.3. Sample Network Client

The following sample network server script is written in Perl, and is designed to translated packed hex strings (two-hex characters per byte) from their hex representation into their character representation.

```
#!/usr/bin/perl

use Switch;
use IO::Socket::INET;
use JSON qw( decode_json encode_json);
use Data::Dumper;

# auto-flush on socket
$| = 1;

my $serverName = "Perl_BLOB_to_String_v1";

while(1)
{
# creating a listening socket
my $socket = new IO::Socket::INET (
    LocalHost => '0.0.0.0',
    LocalPort => '3112',
    Proto => 'tcp',
    Listen => 5,
    Reuse => 1
);
die "Cannot create socket $!\n" unless $socket;
print "********\nServer waiting for client connection on port 3112\n******\n\n\n";

# Waiting for a new client connection
my $client_socket = $socket->accept();

# Fet information about a newly connected client
my $client_address = $client_socket->peerhost();
my $client_port = $client_socket->peerport();
print "Connection from $client_address:$client_port\n";

my $data = "";

while(    $data = $client_socket->getline())
{
    # Eead up to 1024 characters from the connected client
    chomp($data);

    print "\n\nReceived: <$data>\n";

    # Decode the JSON part
    my $msg = decode_json($data);

    # Extract payload
    my $payload = undef;

    if ($msg->{payload} > 0)
        {
            print STDERR "Reading $msg->{payload} bytes\n";
            $client_socket->read($payload,$msg->{payload});
            print "Payload: <$payload>\n";
```

```
         }

    switch( $msg->{'type'} )
    {
        case "prepare"
        {
              print STDERR "Received prepare request\n";

            # Send acknowledged message
            my $out = '{ "protocol": "v0_9", "type": "acknowledged", ' .
                      '"return": 0, "service": "' . $msg->{'service'} . '", "payload": ' .
                      length($serverName) . '}' . "\n" . $serverName;

            print $client_socket "$out";
            print "Sent: <$out>\n";
                print STDERR "Sent acknowledge request\n";
        }
        case "release"
        {
            # Send acknowledged message
            my $out = '{ "protocol": "v0_9", "type": "acknowledged", ' .
                      '"return": 0, "service": "' . $msg->{'service'} . '", "payload": 0}';

            print $client_socket "$out\n";
            print "Sent: <$out>\n";
        }
        case "filter"
        {
            # Send filtered message

                  print STDERR "Sending filtered payload\n";

            my $filtered = "FILTERED";
            my $out = <<END;
{
"protocol": "v0_9",
"type": "filtered",
"transformation": "$msg->{'transformation'}",
"return": 0,
"service": "$msg->{'service'}",
"seqno": $msg->{'seqno'},
"row": $msg->{'row'},
"schema": "$msg->{'schema'}",
"table": "$msg->{'table'}",
"column": "$msg->{'column'}",
"fragment": 1,
"fragments": 1,
"payload": @{[length($filtered)]}
}
END

$out =~ s/\n//g;
            print "About to send: <$out>\n";
            $client_socket->send("$out\n" . $filtered);
print("Response sent\n");
        }
    }

print("End of loop, hoping for next packet\n");
}

    # Notify client that we're done writing
    shutdown($client_socket, 1);

$socket->close();
}
```

12.3.29. `nocreatedbifnotexists.js` Filter

The `nocreatedbifnotexists` filter removes statements that start with:

```
CREATE DATABASE IF NOT EXISTS
```

Pre-configured filter name	`nocreatedbifnotexists`
JavaScript Filter File	`tungsten-replicator/support/filters-javascript/nocreatedbifnotexists.js`
Property prefix	*`replicator.filter.nocreatedbifnotexists`*
Stage compatibility	`q-to-dbms`
tpm Option compatibility	`--svc-applier-filters` [515]

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()` [589] function that defines the string to be examined:

```
function prepare()
{
  beginning = "CREATE DATABASE IF NOT EXISTS";
}
```

Row based changes can be ignored, but for statement based events, the SQL is examine and the statement removed if the SQL starts with the text in the `beginning` variable:

```
sql = d.getQuery();
if(sql.startsWith(beginning))
{
    data.remove(i);
    i--;
}
```

12.3.30. OptimizeUpdates Filter

The `optimizeupdates` filter works with row-based events to simplify the update statement and remove columns/values that have not changed. This reduces the workload and row data exchanged between replicators.

Pre-configured filter name	`optimizeupdates`
Classname	`com.continuent.tungsten.replicator.filter.OptimizeUpdatesFilter`
Property prefix	*`replicator.filter.optimizeupdates`*
Stage compatibility	
tpm Option compatibility	
Data compatibility	Row events
Parameters	
(none)	

The filter operates by removing column values for keys in the update statement that do not change. For example, when replicating the row event from the statement:

```
mysql>  update testopt set msg = 'String1', string = 'String3' where id = 1;
```

Generates the following THL event data:

```
- SQL(0) =
 - ACTION = UPDATE
 - SCHEMA = test
 - TABLE = testopt
 - ROW# = 0
  - COL(1: id) = 1
  - COL(2: msg) = String1
  - COL(3: string) = String3
  - KEY(1: id) = 1
```

Column 1 [`id`] in this case is automatically implied by the KEY entry required for the update.

With the `optimizeupdates` filter enabled, the data in the THL is simplified to:

```
- SQL(0) =
 - ACTION = UPDATE
 - SCHEMA = test
 - TABLE = testopt
 - ROW# = 0
  - COL(2: msg) = String1
  - COL(3: string) = String4
  - KEY(1: id) = 1
```

In tables where there are multiple keys the stored THL information can be reduced further.

Warning

The filter works by comparing the value of each KEY and COL entry in the THL and determining whether the value has changed or not. If the number of keys and columns do not match then the filter will fail with the following error message:

```
Caused by: java.lang.Exception: Column and key count is different in this event! Cannot filter
```

This may be due to a filter earlier within the filter configuration that has optimized or simplified the data. For example, the `pkey` filter removes KEY entries from the THL that are not primary keys, or `dropcolumn` which drops column data.

The following error message may appear in the logs and in the output from trepctl status to indicate that this ordering issue may be the problem:

```
OptimizeUpdatesFilter cannot filter, because column and key count is different.
Make sure that it is defined before filters which remove keys (eg. PrimaryKeyFilter).
```

12.3.31. PrimaryKey Filter

The `PrimaryKey` filter adds primary key information to row-based replication data. This is required by heterogeneous environments to ensure that the primary key is identified when updating or deleting tables. Without this information, the primary key to use, for example as the document ID in a document store such as MongoDB, is generated dynamically. In addition, without this filter in place, when performing update or delete operations a full table scan is performed on the target dataserver to determine the record that must be updated.

Pre-configured filter name	`pkey`
Classname	`com.continuent.tungsten.replicator.filter.PrimaryKeyFilter`
Property prefix	*`replicator.filter.pkey`*
Stage compatibility	`binlog-to-q`
tpm Option compatibility	`--repl-svc-extractor-filters` [515]
Data compatibility	Row events
Keeps Cached Data	Yes
Cached Refreshed When?	Emptied when going `OFFLINE`; Updated when `ALTER` statement seen
Metadata Updated	Yes; `tungsten_filter_primarykey=true`

Parameters

Parameter	Type	Default	Description
`user`	string	*`${replicator.global.extract.db.user}`*	The username for the connection to the database for looking up column definitions
`password`	string	*`${replicator.global.extract.db.password}`*	The password for the connection to the database for looking up column definitions
`url`	string	*`jdbc:mysql:thin://${replicator.global.extract.db.host}:» ${replicator.global.extract.db.port}/${replicator.schema}?createDB=true`*	JDBC URL of the database connection to use for looking up column definitions
`addPkeyToInsert`	`boolean`	`false`	If set to true, primary keys are added to `INSERT` operations. This setting is required for batch loading
`addColumnsToDeletes`	`boolean`	`false`	If set to true, full column metadata is added to `DELETE` operations. This setting is required for batch loading
`custompkey`	`filename`	`(empty)`	If set to the value of a JSON file, the file is used to determine custom primary key specifications in place of database derived versions. See Section 12.3.31.1, "Setting Custom Primary Key Definitions".
`reconnectTimeout`	integer	3600	Timeout for refreshing connection to database

Note

This filter is designed to be used for testing and with heterogeneous replication where the field name information can be used to construct and build target data structures.

For example, in the following THL fragment, the key information includes data for all columns, which the is the default behavior for `UPDATE` and `DELETE` operations.

```
SEQ# = 142 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:31:04.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:0000000000022187;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = UPDATE
 - SCHEMA = test
 - TABLE = salesadv
 - ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 1
  - COL(3: city) = 8374
  - COL(4: salesman) = 1
  - COL(5: value) = 89000.00
  - KEY(1: id) = 2
  - KEY(2: country) = 1
  - KEY(3: city) = 8374
  - KEY(4: salesman) = 1
  - KEY(5: value) = 89000.00
```

When the `PrimaryKey` filter is enabled, the key information has been optimized to only contain the actual primary keys in the row-based THL record:

```
SEQ# = 142 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:31:04.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:0000000000022187;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = UPDATE
 - SCHEMA = test
 - TABLE = salesadv
 - ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 1
  - COL(3: city) = 8374
  - COL(4: salesman) = 1
  - COL(5: value) = 89000.00
  - KEY(1: id) = 2
```

The final line shows the addition of the primary key `id` added to THL event.

Important

The filter determines primary key information by examining the DDL for the table, and keeping that information in an internal cache. If the DDL for a table is not known, or an `ALTER TABLE` statement is identified, the cache information is updated before the THL is then modified with the primary key information.

In the situation where you enable the filter, but have not create primary key information on the tables, it is possible that creating or adding other index types (such as `UNIQUE`) on a table, could lead to the incorrect primary key information being updated in the THL, particularly if there are active transactions taking place during and/or immediately after the `ALTER` statement.

The safest way to perform an index update in case remains the same as for any safe DDL update:

- Put the replicator offline
- Change the DDL for the table or tables
- Put the replicator online

The two options, *`addPkeyToInsert`* and *`addColumnsToDeletes`* add the primary key information to `INSERT` and `DELETE` operations respectively. In a heterogeneous environment, these options should be enabled to prevent full-table scans during update and deletes.

12.3.31.1. Setting Custom Primary Key Definitions

6.0.0 or later. Custom primary key configuration support available in 6.0.0 or later.

Not all tables and databases set or provide explicit primary key information, and in some cases, it is not possible to change the index definition on the source table to include primary key information. Without primary key information in the THL, replicating data into heterogeneous

targets can fail, because there is no way for the target environment to correctly identify the primary key information, and therefore the specific record or records to be updated.

The `pkey` filter supports defining custom columns to make up a primary key in this situation, avoiding the need to explicitly set index information within the database.

The custom primary key setting works as follows:

- When processing a THL entry, if the custom primary key configuration file has been set and exists, and the incoming schema/table name has been defined in the configuraiton file, the custom pkey configuration is used.
- Otherwise, the primary key information is taken from the database if it exists.

If neither of these conditions is met, then no primary key data is added to the THL during process.

To configure a custom primary key for one or more tables:

1. Copy the sample primary key configuration file, located within the distribution as `tungsten/tungsten-replicator/support/filters-config/custompkey.json` into the `share` directory within the installation. For example, `/opt/continuent/share`.
2. Update the configuration to include the specific primary key settings for the incoming table. The format of the file is JSON, using a structured layout based on the common JSON filter configuration format (see Section 12.4, “Standard JSON Filter Configuration”). The sample file contains an example for the schema `test` and the table `msg`:

   ```
   {
       "__default": {
           "IGNORE" : "pkey"
       },
       "test" : {
           "msg" : {
               "msg" : "pkey"
           }
       }
   }
   ```

   In the above example, `msg` is the name of the column to be specified as the primary key. The `pkey` indicates that this column must be a primary key field. You can also specify multiple columns:

   ```
   {
       "__default": {
           "IGNORE" : "pkey"
       },
       "test" : {
           "msg" : {
               "id"  : "pkey",
               "msg" : "pkey"
           }
       }
   }
   ```

   To include another table within the same schema:

   ```
   {
       "__default": {
           "IGNORE" : "pkey"
       },
       "test" : {
           "msg" : {
               "msg" : "pkey"
           },
           "orders" : {
               "orderid" : "pkey"
           }
       }
   }
   ```

   > **Note**
   >
   > The `__default` section must remain in place, although it has no impact on processing, it is used to ensure the file is valid for the filter configuration.

3. Update the configuration of your installtion through tpm to specify the custom primary key file in the properties:

   ```
   shell> tpm update alpha --property=replicator.filter.pkey.custompkey=/opt/continuent/share/custompkey.json
   ```

Once this process has been completed, the replicator will add the custom primary key fields to the THL during processing. For example:

```
- SQL(0) =
```

```
- ACTION = INSERT
- SCHEMA = test
- TABLE = msg
- ROW# = 0
 - COL(1: id) = 6
 - COL(2: msg) = new test
 - KEY(1: id) = NULL
```

12.3.32. PrintEvent Filter

Outputs transaction event information to the replication logging system

Pre-configured filter name	`printevent`
Classname	`com.continuent.tungsten.replicator.filter.PrintEventFilter`
Property prefix	*`replicator.filter.printevent`*
Stage compatibility	
tpm Option compatibility	
Data compatibility	Any event
Parameters	
(none)	

12.3.33. Rename Filter

The `rename` filter enables schemas to be renamed at the database, table and column levels, and for complex combinations of these renaming operations. Configuration is through a CSV file that defines the rename parameters. A single CSV file can contain multiple rename definitions. The rename operations occur only on ROW based events.

The `rename` filter also performs schema renames on statement data.

Pre-configured filter name		`rename`	
Classname		`com.continuent.tungsten.replicator.filter.RenameFilter`	
Property prefix		*`replicator.filter.rename`*	
Stage compatibility			
tpm Option compatibility			
Data compatibility		Row events.	
Parameters			
Parameter	Type	Default	Description
`definitionsFile`	string	`{replicator.home.dir}/samples/extensions/java/rename.csv`	Location of the CSV file that contains the rename definitions.

The CSV file is only read when an explicit reconfigure operation is triggered. If the file is changed, a configure operation (using tpm update) must be initiated to force reconfiguration.

To enable using the default CSV file:

```
shell> ./tools/tpm update alpha --svc-applier-filters=rename
```

The CSV consists of multiple lines, one line for each rename specification. Comments are supposed using the `#` character.

The format of each line of the CSV is:

```
originalSchema,originalTable,originalColumn,newSchema,newTable,newColumn
```

Where:

- `originalSchema`, `originalTable`, `originalColumn` define the original schema, table and column.

  Definition can either be:

  - Explicit schema, table or column name
  - `*` character, which indicates that all entries should match.

- `newSchema`, `newTable`, `newColumn` define the new schema, table and column for the corresponding original specification.

  Definition can either be:

  - Explicit schema, table or column name
  - `-` character, which indicates that the corresponding object should not be updated.

For example, the specification:

```
*,chicago,*,-,newyork,-
```

Would rename the table `chicago` in every database schema to `newyork`. The schema and column names are not modified.

The specification:

```
*,chicago,destination,-,-,source
```

Would match all schemas, but update the column `destination` in the table `chicago` to the column name `source`, without changing the schema or table name.

Processing of the individual rules is executed in a specific order to allow for complex matching and application of the rename changes.

- Rules are case sensitive.
- Schema names are looked up in the following order:
  1. `schema.table` (explicit schema/table)
  2. `schema.*` (explicit schema, wildcard table)
- Table names are looked up in the following order:
  1. `schema.table` (explicit schema/table)
  2. `*.table` (wildcard schema, explicit table)
- Column names are looked up in the following order:
  1. `schema.table` (explicit schema/table)
  2. `schema.*` (explicit schema, wildcard table)
  3. `*.table` (wildcard schema, explicit table)
  4. `*.*` (wildcard schema, wildcard table)
- Rename operations match the first specification according to the above rules, and only one matching rule is executed.

12.3.33.1. Rename Filter Examples

When processing multiple entries that would match the same definition, the above ordering rules are applied. For example, the definition:

```
asia,*,*,america,-,-
asia,shanghai,*,europe,-,-
```

Would rename `asia.shanghai` to `europe.shanghai`, while renaming all other tables in the schema `asia` to the schema `america`. This is because the explicit `schema.table` rule is matched first and then executed.

Complex renames involving multiple schemas, tables and columns can be achieved by writing multiple rules into the same CSV file. For example given a schema where all the tables currently reside in a single schema, but must be renamed to specific continents, or to a 'miscellaneous' schema, while also updating the column names to be more neutral would require a detailed rename definition.

Existing tables are in the schema `sales`:

```
chicago
newyork
london
paris
munich
moscow
tokyo
shanghai
sydney
```

Need to be renamed to:

```
northamerica.chicago
northamerica.newyork
europe.london
europe.paris
europe.munich
misc.moscow
asiapac.tokyo
asiapac.shanghai
misc.sydney
```

Meanwhile, the table definition needs to be updated to support more complex structure:

```
id
area
country
city
value
type
```

The area is being updated to contain the region within the country, while the value should be renamed to the three-letter currency code, for example, the `london` table would rename the `value` column to `gbp`.

The definition can be divided up into simple definitions at each object level, relying on the processing order to handle the individual exceptions. Starting with the table renames for the continents:

```
sales,chicago,*,northamerica,-,-
sales,newyork,*,northamerica,-,-
sales,london,*,europe,-,-
sales,paris,*,europe,-,-
sales,munich,*,europe,-,-
sales,tokyo,*,asiapac,-,-
sales,shanghai,*,asiapac,-,-
```

A single rule to handle the renaming of any table not explicitly mentioned in the list above into the `misc` schema:

```
*,*,*,misc,-,-
```

Now a rule to change the `area` column for all tables to `region`. This requires a wildcard match against the schema and table names:

```
*,*,area,-,-,region
```

And finally the explicit changes for the value column to the corresponding currency:

```
*,chicago,value,-,-,usd
*,newyork,value,-,-,usd
*,london,value,-,-,gbp
*,paris,value,-,-,eur
*,munich,value,-,-,eur
*,moscow,value,-,-,rub
*,tokyo,value,-,-,jpy
*,shanghai,value,-,-,cny
*,sydney,value,-,-,aud
```

12.3.34. Replicate Filter

The `replicate` filter enables explicit inclusion or exclusion of tables and schemas. Each specification supports wildcards and multiple entries.

Pre-configured filter name	`replicate`
Classname	`com.continuent.tungsten.replicator.filter.ReplicateFilter`
Property prefix	*`replicator.filter.replicate`*
Stage compatibility	Any
tpm Option compatibility	
Data compatibility	Any event

Parameters

Parameter	Type	Default	Description
`ignore`	string	empty	Comma separated list of database/tables to ignore during replication
`do`	string	empty	Comma separated list of database/tables to replicate

Rules using the supplied parameters are evaluated as follows:

- When both `do` and `ignore` are empty, updates are allowed to any table.
- When only `do` is specified, only the schemas (or schemas and tables) mentioned in the list are replicated.
- When only `ignore` is specified, all schemas/tables are replicated except those defined.

For each parameter, a comma-separated list of schema or schema and table definitions are supported, and wildcards using `*` (any number of characters) and `?` (single character) are also honored. For example:

- `do=sales`

  Replicates only the schema `sales`.

- `ignore=sales`

  Replicates everything, ignoring the schema `sales`.

- `ignore=sales.*`

  Replicates everything, ignoring the schema `sales`.

- `ignore=sales.quarter?`

  Replicates everything, ignoring all tables within the `sales` schema starting with `sales.quarter` and a single character. This would ignore `sales.quarter1` but replicate `sales.quarterlytotals`.

- `ignore=sales.quarter*`

  Replicates everything, ignoring all tables in the schema `sales` starting with `quarter`.

- `do=*.quarter`

  Replicates only the table named `quarter` within any schema.

- `do=sales.*totals,invoices`

  Replicates only tables in the `sales` schema that end with `totals`, and the entire `invoices` schema.

12.3.35. ReplicateColumns Filter

Removes selected columns from row-based transaction data.

Pre-configured filter name		`replicatecolumns`	
Classname		`com.continuent.tungsten.replicator.filter.ReplicateColumnsFilter`	
Property prefix		*`replicator.filter.replicatecolumns`*	
Stage compatibility			
tpm Option compatibility			
Data compatibility		Row events	
Parameters			
Parameter	Type	Default	Description
`ignore`	string	empty	Comma separated list of tables and optional columns names to ignore during replication
`do`	string	empty	Comma separated list of tables and optional column names to replicate

12.3.36. Row Add Database Name Filter

The `rowadddbname` filter adds a new column to every incoming row of data containing the schema name of the table. This can be used in combination with analytics replication targets where the information is being written into a single schema, concentrating data from multiple source databases into a single database. Optionally, the filter can also add the information as a primary key (required for some heterogeneous targets), and the hash of the source database name.

Pre-configured filter name	`rowadddbname`

Classname		com.continuent.tungsten.replicator.filter.JavaScriptFilter	
Property prefix		*replicator.filter.rowadddbname*	
Stage compatibility		Any	
tpm Option compatibility			
Data compatibility		Row-events only	
Parameters			
Parameter	Type	Default	Description
adddbhash	boolean	true	Add a hash column, computed using the Java hash value of the string for the incoming source database name
addkey	boolean	true	Add the information to the primary key data in the THL, in addition to the column data
fieldname	string	dbname	The name of the database name column that will be added
fieldhashname	string	dbname	The name of the database has name column that will be added

The `rowadddbname` filter adds a column to every row of THL processed by the filter. This can be used when data is being written into a single target schema within an analytics environment, and where the source database can be used to identify a customer, project or dataset, and therefore queried within the analytics platform either specifically or with other datasets.

The filter is able to perform the following modifications to every row of incoming data:

- Add the source database or schema name.
- Add a numerical hash value of the string of the source database of schema name.
- Add the database name (and hash name) to the primary key data.

For example, the source THL:

```
- ROW# = 0
  - COL(1: id) = 12
  - COL(2: msg) = Hello
  - COL(3: msg2) = World
  - KEY(1: id) = NULL
```

And after the filter has been applied:

```
- ROW# = 0
  - COL(1: id) = 12
  - COL(2: msg) = Hello
  - COL(3: msg2) = World
  - COL(4: dbname) = msg
  - COL(5: dbname_hash) = 108417
  - KEY(1: id) = NULL
  - KEY(4: dbname) = NULL
  - KEY(5: dbname_hash) = NULL
```

This filter is a required component of deployments when replicating into a single schema within Vertica (see Deploying the Vertica Applier) and Amazon Redshift (see Deploying the Amazon Redshift Applier).

12.3.37. Row Add Transaction Info Filter

The `rowaddtxninfo` filter examines an entire row-based event within the THL and then builds a list of transaction information which is embedded into the metadata for the event. This can be combined with functionality in corresponding appliers, such as Deploying the Kafka Applier which can then be used and embedded into messages or document-based databases.

Pre-configured filter name	rowaddtxninfo
Classname	com.continuent.tungsten.replicator.filter.JavaScriptFilter
Property prefix	*replicator.filter.rowaddtxninfo*
Stage compatibility	Any
tpm Option compatibility	
Data compatibility	Row-events only

Parameters			
Parameter	Type	Default	Description

The `rowaddtxninfo` filter processes an entire transaction to determine the following information:

- List of schemas and tables affected by the transaction.
- A count of the rows inserted, updated, or deleted per schema/table combination.
- A total count of all the rows inserted, updated, or deleted across the entire transaction.

Once the information about the transaction has been collected, the data is then formulated into key/value pairs that are incorporated into the metadata for the entire THL event.

For example, when inserting information into the same table across three separate statements the but the same overall transaction, the THL with the filter enabled looks like the example below:

```
shell> thl list -last
SEQ# = 162 / FRAG# = 0 (last frag)
- TIME = 2018-03-02 08:47:14.0
- EPOCH# = 162
- EVENTID = mysql-bin.000065:0000000000000534;-1
- SOURCEID = trfiltera
- METADATA = [mysql_server_id=366;dbms_type=mysql;tz_aware=true;strings=utf8;service=alpha;shard=msg;
» tungsten_filter_columnname=true; tungsten_filter_primarykey=true;tungsten_filter_enumtostring=true;
» txinfo_rowcount_msg.msg=1;txinfo_rowcount_msg.msgsub=2;txinfo_rowcount=3]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1, time_zone = '+00:00']
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = msg
 - TABLE = msg
 - ROW# = 0
  - COL(1: id) = 108
  - COL(2: msg) = txinfo
  - COL(3: msg2) = txinfo
  - KEY(1: id) = NULL
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1, time_zone = '+00:00']
- SQL(1) =
 - ACTION = INSERT
 - SCHEMA = msg
 - TABLE = msgsub
 - ROW# = 0
  - COL(1: id) = 108
  - COL(2: msg) = subtx
 - ROW# = 1
  - COL(1: id) = 108
  - COL(2: msg) = subxt
```

Examining the metadata more closely you can see the transaction information:

```
...   txinfo_rowcount_msg.msg=1;txinfo_rowcount_msg.msgsub=2;txinfo_rowcount=3 ...
```

This can be further extrapolated as:

- `txinfo_rowcount_msg.msg=1`

  One row has been updated in the `msg.msg` schema/table.

- `txinfo_rowcount_msg.msgsub=2`

  Two rows has been updated in the `msg.submsg` schema/table.

- `txinfo_rowcount=3`

  A total of three rows have been updated within the transaction.

> **Note**
>
> If you transactional information is needed within a Kafka deployment, this filter must have been enabled for the transaction information to be included. See Optional Configuration Parameters for Kafka, for more information.

12.3.38. SetToString Filter

The `SetToString` converts the `SET` column type from the internal representation to a string-based representation in the THL. This achieved by accessing the extractor database, obtaining the table definitions, and modifying the THL data before it is written into the THL file.

Pre-configured filter name		settostring	
Classname		com.continuent.tungsten.replicator.filter.SetToStringFilter	
Property prefix		*replicator.filter.settostring*	
Stage compatibility		binlog-to-q	
tpm Option compatibility		--repl-svc-extractor-filters [515]	
Data compatibility		Row events	
Parameters			
Parameter	Type	Default	Description
user	string	*${replicator.global.extract.db.user}*	The username for the connection to the database for looking up column definitions
password	string	*${replicator.global.extract.db.password}*	The password for the connection to the database for looking up column definitions
url	string	jdbc:mysql:thin://${replicator.global.extract.db.host}: » ${replicator.global.extract.db.port}/${replicator.schema}?createDB=true	JDBC URL of the database connection to use for looking up column definitions

The SetToString filter should be used with heterogeneous replication to ensure that the data is represented as the string value, not the internal numerical representation.

In the THL output below, the table has a SET column, salesman:

```
mysql> describe salesadv;
+----------+-----------------------------------+------+-----+---------+----------------+
| Field    | Type                              | Null | Key | Default | Extra          |
+----------+-----------------------------------+------+-----+---------+----------------+
| id       | int(11)                           | NO   | PRI | NULL    | auto_increment |
| country  | enum('US','UK','France','Australia') | YES  |     | NULL    |                |
| city     | int(11)                           | YES  |     | NULL    |                |
| salesman | set('Alan','Zachary')             | YES  |     | NULL    |                |
| value    | decimal(10,2)                     | YES  |     | NULL    |                |
+----------+-----------------------------------+------+-----+---------+----------------+
```

When extracted in the THL, the representation uses the internal value (for example, 1 for the first element of the set description). This can be seen in the THL output below.

```
SEQ# = 138 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:09:35.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:0000000000021434;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = test
 - TABLE = salesadv
 - ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 1
  - COL(3: city) = 8374
  - COL(4: salesman) = 1
  - COL(5: value) = 35000.00
```

For the salesman column, the corresponding value in the THL is 1. With the SetToString filter enabled, the value is expanded to the corresponding string value:

```
SEQ# = 121 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:05:14.0
- EPOCH# = 102
- EVENTID = mysql-bin.000012:0000000000018866;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = test
 - TABLE = salesadv
 - ROW# = 0
  - COL(1: id) = 1
```

```
- COL(2: country) = US
- COL(3: city) = 8374
- COL(4: salesman) = Alan
- COL(5: value) = 35000.00
```

The examples here also show the Section 12.3.20, “EnumToString Filter” and Section 12.3.5, “ColumnName Filter” filters.

12.3.39. Shard Filter

Used to enforce database schema sharding between specific Primaries

Pre-configured filter name		shardfilter	
Classname		com.continuent.tungsten.replicator.filter.ShardFilter	
Property prefix		*replicator.filter.shardfilter*	
Stage compatibility			
tpm Option compatibility			
Data compatibility		Any event	
Parameters			
Parameter	Type	Default	Description
enabled	boolean	false	If set to true, enables the shard filter
unknownShardPolicy	string	error	Select the filter policy when the shard unknown; valid values are accept, drop, warn, and error
unwantedShardPolicy	string	error	Select the filter policy when the shard is unwanted; valid values are accept, drop, warn, and error
enforcedHome	boolean	false	If true, enforce the home for the shard
allowWhitelisted	boolean	false	If true, allow explicitly whitelisted shards
autoCreate	boolean	false	If true, allow shard rules to be created automatically

12.3.40. shardbyseqno.js Filter

Shards within the replicator enable data to be parallelized when they are applied on the Target.

Pre-configured filter name		shardbyseqno	
JavaScript Filter File		tungsten-replicator/support/filters-javascript/shardbyseqno.js	
Property prefix		*replicator.filter.shardbyseqno*	
Stage compatibility		q-to-dbms	
tpm Option compatibility		--svc-applier-filters [515]	
Data compatibility		Any event	
Parameters			
Parameter	Type	Default	Description
shards	numeric	(none)	Number of shards to be used by the applier

The shardbyseqno filter updates the shard ID, which is embedded into the event metadata, by a configurable number of shards, set by the *shards* parameter in the configuration:

```
replicator.filter.shardbyseqno=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.shardbyseqno.script=${replicator.home}/samples/extensions/javascript/shardbyseqno.js
replicator.filter.shardbyseqno.shards=10
```

The filter works by setting the shard ID in the event using the setShardId() method on the event object:

```
event.setShardId(event.getSeqno() % shards);
```

Note

Care should be taken with this filter, as it assumes that the events can be applied in a completely random order by blindly updating the shard ID to a computed valued. Sharding in this way is best used when provisioning new Targets.

12.3.41. shardbytable.js Filter

An alternative to sharding by sequence number is to create a shard ID based on the individual database and table. The shardbytable filter achieves this at a row level by combining the schema and table information to form the shard ID. For all other events, including statement based events, the shard ID #UNKNOWN is used.

Pre-configured filter name		shardbytable	
JavaScript Filter File		tungsten-replicator/support/filters-javascript/shardbytable.js	
Property prefix		*replicator.filter.shardbytable*	
Stage compatibility		remote-to-thl	
tpm Option compatibility		--svc-remote-filters [516]	
Data compatibility		ROW	
Parameters			
Parameter	Type	Default	Description

> **Note**
>
> For this filter to function, you will need to ensure your database is configured for ROW based binary logging

The key part of the filter is the extraction and construction of the ID, which occurs during row processing:

```
oneRowChange = rowChanges.get(j);
schemaName = oneRowChange.getSchemaName();
tableName  = oneRowChange.getTableName();

id = schemaName + "_" + tableName;
if (proposedShardId == null)
{
  proposedShardId = id;
}
```

12.3.42. SkipEventByType Filter

The SkipEventByType filter enables you to skip individual events based on the event type, schema and table. For example, if you want to skip all DELETE events on the schema/table SALES.INVOICES (to prevent deletion of invoice data), this filter will skip the event entirely and it will not be applied to the target.

Pre-configured filter name		skipeventbytype	
Classname		com.continuent.tungsten.replicator.filter.SkipEventByTypeFilter	
Property prefix		*replicator.filter.skipeventbytype*	
Stage compatibility		any	
tpm Option compatibility		--repl-svc-extractor-filters [515], --repl-svc-applier-filters [515]	
Data compatibility		Row events	
Parameters			
Parameter	Type	Default	Description
definitionsFile	string	support/filters-config/skipeventbytype.json	JSON file containing the definition of which events and which tables to skip

Configuration of the filter is made using the generic JSON file, which supports both default options to happen for all tables not otherwise explicitly specified. The default JSON file allows all operations:

```
{
    "__default": {
        "INSERT" : "allow",
        "DELETE" : "allow",
        "UPDATE" : "allow"
    },
    "SCHEMA" : {
        "TABLE" : {
            "INSERT" : "allow",
            "DELETE" : "deny",
            "UPDATE" : "deny"
        }
    }
```

```
}
```

The default section handles the default response when an explicit schema or table name does not appear. Further sections are then organised by schema and then table name. Where the setting is `allow`, the operation will be processed. A `deny` skips the entire event.

To disable all `DELETE` operations, regardless of which table they occur in:

```
{
    "__default": {
        "INSERT" : "allow",
        "DELETE" : "deny",
        "UPDATE" : "allow"
    },
    "SCHEMA" : {
        "TABLE" : {
            "INSERT" : "allow",
            "DELETE" : "deny",
            "UPDATE" : "deny"
        }
    }
}
```

To normally allow all operations, except on the SALES.INVOICE schema/table:

```
{
    "__default": {
        "INSERT" : "allow",
        "DELETE" : "allow",
        "UPDATE" : "allow"
    },
    "SALES" : {
        "INVOICE" : {
            "INSERT" : "allow",
            "DELETE" : "deny",
            "UPDATE" : "deny"
        }
    }
}
```

12.3.43. TimeDelay (delay) Filter

The `TimeDelayFilter` delays writing events to the THL and should be used only on Appliers in the `remote-to-thl` stage. This delays writing the transactions into the THL files, but allows the application of the data to the database to continue without further intervention.

From version 6.1.4 the delayInMs Filter was also added which allows delay prescion in milliseconds. See Section 12.3.44, “TimeDelayMsFilter (delayInMS) Filter” for more detail.

Pre-configured filter name		`delay`	
Classname		`com.continuent.tungsten.replicator.filter.TimeDelayFilter`	
Property prefix		*`replicator.filter.delay`*	
Stage compatibility		remote-to-thl	
tpm Option compatibility		`--repl-svc-thl-filters` [518]	
Data compatibility		Any event	
Parameters			
Parameter	Type	Default	Description
`delay`	numeric	300	Number of seconds to delay transaction processing row

The `TimeDelayFilter` delays the application of transactions recorded in the THL. The delay can be used to allow point-in-time recovery of DML operations before the transaction has been applied to the Target, or where data may need to be audited or checked before transactions are committed.

> **Note**
>
> For effective operation, Source and Targets should be synchronized using NTP or a similar protocol.

To enable the `TimeDelayFilter`, update the tungsten.ini configuration file to enable the filter. For example, to enable the delay for 600 seconds:

```
shell> vi /etc/tungsten/tungsten.ini
[serviceName]
...
```

```
svc-applier-filters=delay
property=replicator.filter.delay.delay=600
...

shel> tpm update
```

Time delay of transaction events should be performed with care, since the delay will prevent a Target from being up to date compared to the Source. In the event of a node failure, an up to date Target is required to ensure that data is safe.

12.3.44. TimeDelayMsFilter (delayInMS) Filter

The `TimeDelayMsFilter` delays writing events to the THL and should be used only on Appliers in the `remote-to-thl` stage. This delays writing the transactions into the THL files, but allows the application of the data to the database to continue without further intervention.

This filter allows delay precision in milliseconds. If you wish to delay to second precision, then the `TimeDelay` filter would also be appropriate. See Section 12.3.43, "TimeDelay (delay) Filter" for more detail.

Pre-configured filter name		`delayInMs`	
Classname		`com.continuent.tungsten.replicator.filter.TimeDelayMsFilter`	
Property prefix		*`replicator.filter.delayInMs`*	
Stage compatibility		remote-to-thl	
tpm Option compatibility		`--repl-svc-thl-filters` [518]	
Data compatibility		Any event	
Parameters			
Parameter	Type	Default	Description
`delay`	numeric	60000	Number of milliseconds to delay transaction processing row

The `TimeDelayMsFilter` delays the application of transactions recorded in the THL. The delay can be used to allow point-in-time recovery of DML operations before the transaction has been applied to the Target, or where data may need to be audited or checked before transactions are committed.

> **Note**
>
> For effective operation, Source and Targets should be synchronized using NTP or a similar protocol.

To enable the `TimeDelayMsFilter`, update the tungsten.ini configuration file to enable the filter. For example, to enable the delay for 120000 milliseconds:

```
shell> vi /etc/tungsten/tungsten.ini
[serviceName]
...
svc-applier-filters=delayInMs
property=replicator.filter.delayInMs.delay=120000
...

shel> tpm update
```

Time delay of transaction events should be performed with care, since the delay will prevent an Target from being up to date compared to the Source. In the event of a node failure, an up to date Target is required to ensure that data is safe.

12.3.45. `tosingledb.js` Filter

This filter updates the replicated information so that it goes to an explicit schema, as defined by the user. The filter can be used to combine multiple tables to a single schema.

Pre-configured filter name	`tosingledb`
JavaScript Filter File	`tungsten-replicator/support/filters-javascript/tosingledb.js`
Property prefix	*`replicator.filter.ansiquotes`*
Stage compatibility	`q-to-dbms`
tpm Option compatibility	`--svc-applier-filters` [515]
Data compatibility	Any event
Parameters	

Parameter	Type	Default	Description
db	string	(none)	Database name into which to replicate all tables
skip	string	(none)	Comma-separated list of databases to be ignored

A database can be optionally ignored through the skip parameter within the configuration:

```
--property=replicator.filter.tosingledb.db=centraldb \
--property=replicator.filter.tosingledb.skip=tungsten
```

The above configures all data to be written into centraldb, but skips the database tungsten.

Similar to other filters, the filter operates by explicitly changing the schema name to the configured schema, unless the skipped schema is in the event data. For example, at a statement level:

```
if(oldDb!=null && oldDb.compareTo(skip)!=0)
{
  d.setDefaultSchema(db);
}
```

12.3.46. truncatetext.js Filter

The truncatetext filter truncates a MySQL BLOB field.

Pre-configured filter name	truncatetext
JavaScript Filter File	tungsten-replicator/support/filters-javascript/truncatetext.js
Property prefix	*replicator.filter.truncatetext*
Stage compatibility	binlog-to-q, q-to-dbms
tpm Option compatibility	--svc-extractor-filters [515], --svc-extractor-filters [515]
Data compatibility	Row events

Parameters			
Parameter	Type	Default	Description
length	numeric	(none)	Maximum size of truncated field (bytes)

The length is determined by the *length* parameter in the properties:

```
replicator.filter.truncatetext=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.truncatetext.script=${replicator.home.dir}/samples/extensions/javascript/truncatetext.js
replicator.filter.truncatetext.length=4000
```

Statement-based events are ignored, but row-based events are processed for each volume value, checking the column type, isBlob() method and then truncating the contents when they are identified as larger than the configured length. To confirm the type, it is compared against the Java class com.continuent.tungsten.replicator.extractor.mysql.SerialBlob, the class for a serialized BLOB value. These need to be processed differently as they are not exposed as a single variable.

```
if (value.getValue() instanceof com.continuent.tungsten.replicator.extractor.mysql.SerialBlob)
{
  blob = value.getValue();
  if (blob != null)
  {
    valueBytes = blob.getBytes(1, blob.length());
    if (blob.length() > truncateTo)
    {
      blob.truncate(truncateTo);
    }
  }
}
```

12.3.47. zerodate2null.js Filter

The zerodate2null filter looks complicated, but is very simple. It processes row data looking for date columns. If the corresponding value is zero within the column, the value is updated to NULL. This is required for MySQL to Oracle replication scenarios.

Pre-configured filter name	zerodate2null
JavaScript Filter File	tungsten-replicator/support/filters-javascript/zerodate2null.js
Property prefix	*replicator.filter.zerodate2null*

Stage compatibility		q-to-dbms	
tpm Option compatibility		--svc-applier-filters [515]	
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()
        }
      }
    }
  }
}
```

12.4. Standard JSON Filter Configuration

A number of the filters that are included as part of Tungsten Clustering use a standardised form of configuration file that is designed to be easy to use and familiar, while being flexible enough to support the needs of each filter. For the majority of filter configurations, the core focus of the configuration is based on a 'default' setting, and settings that are specific to a schema or table.

The JSON configuration follows this basic model. The following filters support the use of this JSON configuration file format:

- convertstringfrommysql
- pkey
- skipeventbytype

The basic format of the configuration is a JSON file that is split into two sections:

- A default section, which determines what will happen in the absence of a schema/table specific rule.
- A collection of schema and table specific entries that determine what happens for a specific schema/table combination.

Depending on the filter and use case, the information within both sections can then either be further divided into column-specific information, or the information may be configured as key/value pairs, or objects, to configure individual parts of the filter configuration.

For example, the following configuration file is from the pkey filter:

```
{
    "__default": {
        "IGNORE" : "pkey"
    },
    "test" : {
        "msg" : {
            "msg" : "pkey"
        }
    }
}
```

The above shows the the defaults section, and the schema/table specific section.

> **Note**
>
> Depending on the filter, the default section may merely be a placeholder to indicate the format of the file. The `__default` should never be removed.

The sample shows a full schema name, table name, and then column name configuration.

By comparison, the sample below has only schema and table name information, with the configuration within that section being used to define the key/value pairs for specific operations as part of the `skipeventbytype` filter:

```
{
    "__default": {
        "INSERT" : "allow",
        "DELETE" : "allow",
        "UPDATE" : "allow"
    },
    "SCHEMA" : {
        "TABLE" : {
            "INSERT" : "allow",
            "DELETE" : "deny",
            "UPDATE" : "deny"
        }
    }
}
```

The selection and execution of the rules is determined by some specific rules, as detailed in Section 12.4.1, “Rule Handling and Processing” and Section 12.4.2, “Schema, Table, and Column Selection”.

12.4.1. Rule Handling and Processing

The processing of the rules and the selection of the tables and appropriate response and operation is configured through the combination of the default and schema/table settings according to explicit rules:

- If the incoming data matches the schema and table (and optionally column) according to the rules, use the configuration information in that section.
- If the schema/table is not specified or does not have explicit configuration, use the configuration within the `__default` section instead.

The default rule is always processed and followed if there is no match for an explicit schema, table, or column definition.

12.4.2. Schema, Table, and Column Selection

The format of the JSON configuraiton and the selection of the schema, table, and column information is in the form of nested structure of JSON objects. The schema first, then the table, then optionally the column within a nested JSON structure. For example:

```
"test" : {
    "msg" : {
        "id" : "pkey"
    }
}
```

In the above example:

- `test` is the schema name
- `msg` is the table name within the `test` schema
- `id` is the column name within the `test.msg` table

For different tables within the same schema, place another entry at the same level:

```
"test" : {
    "msg" : {
        "id" : "pkey"
    },
    "orders" : {
        "id" : "pkey"
    },
}
```

The above now handles the tables `msg` and `orders` within the `test` schema.

Wildcards are also supported, using the `*` operator. For example:

```
"orders" : {
    "*" : {
```

```
        "INSERT" : "allow",
        "DELETE" : "deny",
        "UPDATE" : "deny"
    }
}
```

Would match all tables within the `orders` schema. If multiple definitions exist, then the matching operates on the closest match first. For example:

```
"orders" : {
    "sales" : {
        "INSERT" : "deny",
        "DELETE" : "deny",
        "UPDATE" : "deny"
    },
    "*" : {
        "INSERT" : "allow",
        "DELETE" : "deny",
        "UPDATE" : "deny"
    }
}
```

In the above, if the schema/table combination `orders.sales` is seen, the rule for that is always used first as it is explicitly stated. Only tables that do not match the wildcards will use the wildcard entry. If neither an explicit schema/table or wildcard exist, the default is used.

12.5. JavaScript Filters

In addition to the supplied Java filters, Tungsten Clustering also includes support for custom script-based filters written in JavaScript and supported through the JavaScript filter. This filter provides a JavaScript environment that exposes the transaction information as it is processed internally through an object-based JavaScript API.

The JavaScript implementation is provided through the Rhino open-source implementation. Rhino provides a direct interface between the underlying Java classes used to implement the replicator code and a full JavaScript environment. This enables scripts to be developed that have access to the replicator constructs and data structures, and allows information to be updated, reformatted, combined, extracted and reconstructed.

At the simplest level, this allows for operations such as database renames and filtering. More complex solutions allow for modification of the individual data, such as removing nulls, bad dates, and duplication of information.

Warning

If you previously implemented custom filters with older releases of Tungsten Clustering or with the now deprecated Open Source (OSS) Tungsten Replicaror release, you would have edited the `static-SERVICE.properties` file.

This is no longer a supported method of implementing custom filters, and doing so will break automated upgrades through tpm.

To enable custom filters, follow the process here: Section 12.5.2, “Installing Custom JavaScript Filters”

12.5.1. 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.

12.5.1.1. Implementable Functions

Each JavaScript filter must defined one or more functions that are used to operate the filter process. The `filter()` [589] 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()` [589]*

  The `prepare()` [589] 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)` [589]*

  The `filter()` [589] 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()` [589]*

  The `release()` [589] 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.

12.5.1.2. 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 configure and specify configuration information unique to that instance of the filter configuration. Since multiple filter configurations using the same filter definition can be created, using the filter-specific content is the simplest method for obtaining this information.

- Getting Filter Properties

  To obtain the properties configured for the filter within the static configuration file according to the context of the filter configuration, use the `filterProperties` class with the `getString()` method. For example, the `dbrename` filter uses two properties, dbsource and dbtarget to identify the database to be renamed and the new name. The definition for the filter within the configuration file might be:

```
replicator.filter.jsdbrename=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.jsdbrename.script=${replicator.home.dir}/support/filters-javascript/dbrename.js
```

```
replicator.filter.jsdbrename.dbsource=contacts
replicator.filter.jsdbrename.dbtarget=nyc_contacts
```

Within the JavaScript filter, they are retrieved using:

```
sourceName = filterProperties.getString("dbsource");
targetName = filterProperties.getString("dbtarget");
```

- Generic Replicator Properties

  General properties can be retrieved using the `properties` class and the `getString()` method:

```
master = properties.getString("replicator.thl.remote_uri");
```

12.5.1.3. 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.

12.5.1.4. Exposed Data Structures

Within the `filter()` [589] function that must be defined within the JavaScript filter, a single *`event`* object is supplied as the only argument. That event object contains all of the information about a single event as recorded within the THL as part of the replication process. Each event contains metadata information that can be used to identify or control the content, and individual statement and row data that contain the database changes.

The content of the information is a compound set of data that contains one or more further blocks of data changes, which in turn contains one or more blocks of SQL statements or row data. These blocks are defined using the Java objects that describe their internal format, and are exposed within the JavaScript wrapper as JavaScript objects, that can be parsed and manipulated.

At the top level, the Java object provided to the to the `filter()` [589] 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` [593] objects for each individual row that was changed.

When processing the event information, the data that is processed is live and should be updated in place. For example, when examining statement data, the statement needs only be updated in place, not re-submitted. Statements and rows can also be explicitly removed or added by deleting or extending the arrays that make up the objects.

A basic diagram of the structure is shown in the diagram below:

`ReplDBMSEvent`	`DBMSData`	`StatementData`	
	`DBMSData`	`StatementData`	
	`DBMSData`	`RowChangeData`	`OneRowChange` [593]
			`OneRowChange` [593]

			...
		StatementData	
ReplDBMSEvent	DBMSData	RowChangeData	OneRowChange [593]
			OneRowChange [593]
			...

A single event can contain both statement and row change information within the list of individual DBMSData events. An event or

12.5.1.4.1. 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
getAppliedLatency()	Returns the latency of the embedded event. See Section E.2.10, "Terminology: Fields *appliedLatency*"
getData()	Returns an array of the DBMSData objects within the event
getDBMSEvent()	Returns the original DBMSEvent object
getEpochNumber()	Get the Epoch number of the stored event. See THL EPOCH# [679]
getEventId()	Returns the native event ID. See THL EVENTID [680]
getExtractedTstamp()	Returns the timestamp of the event.
getFragno()	Returns the fragment ID. See THL SEQNO [679]
getLastFrag()	Returns true if the fragment is the last fragment in the event.
getSeqno()	Returns the native sequence number. See THL SEQNO [679]
getShardId()	Returns the shard ID for the event.
getSourceId()	Returns the source ID of the event. See THL SOURCEID [680]
setShardId()	Sets the shard ID for the event, which can be used by the filter to set the shard.

The primary method used is getData(), which returns an array of the individual DBMSData objects contain in the event:

```
function filter(event)
{
  data = event.getData();

if(data != null)
  {

    for (i = 0; i < data.size(); i++)
    {
        change = data.get(i);
...
```

Access to the underlying array structure uses the get() method to request individual objects from the array. The size() method returns the length of the array.

Removing or Adding Data Changes

Individual DBMSData objects can be removed from the replication stream by using the remove() method, supplying the index of the object to remove:

```
data.remove(1);
```

The add() method can be used to add new data changes into the stream. For example, data can be duplicated across tables by creating and adding a new version of the event, for example:

```
if(d.getDefaultSchema() != null &&
   d.getDefaultSchema().compareTo(sourceName)==0)
{
  newStatement = new
     com.continuent.tungsten.replicator.dbms.StatementData(d.getQuery(),
                                                           null,
                                                           targetName);
  data.add(data.size(),newStatement);
```

```
}
```

The above code looks for statements within the `sourceName` schema and creates a copy of each statement into the `targetName` schema.

The first argument to `add()` is the index position to add the statement. Zero (0) indicates before any existing changes, while using `size()` on the array effectively adds the new statement change at the end of the array.

Updating the Shard ID

The `setShardId()` method can also be used to set the shard ID within an event. This can be used in filters where the shard ID is updated by examining the schema or table being updated within the embedded SQL or row data. An example of this is provided in Section 12.3.41, "`shardbytable.js` Filter".

12.5.1.4.2. `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 12.5.1.4.3, "`StatementData` Objects". For row data, see Section 12.5.1.4.4, "`RowChangeData` Objects".

12.5.1.4.3. `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
`getQuery()`	Returns the SQL statement
`setQuery()`	Updates the SQL statement
`appendToQuery()`	Appends a string to an existing query
`getDefaultSchema()`	Returns the default schema in which the statement was executed. The schema may be null for explicit or multi-schema queries.
`setDefaultSchema()`	Set the default schema for the SQL statement
`getTimestamp()`	Gets the timestamp of the query. This is required if data must be applied with a relative value by combining the timestamp with the relative value

Updating the SQL

The primary method of processing statement based data is to load and identify the original SQL statement (using `getQuery()`, update or modify the SQL statement string, and then update the statement within the THL again using `setQuery()`. For example:

```
sqlOriginal = d.getQuery();
sqlNew = sqlOriginal.replaceAll('NOTEPAD','notepad');
d.setQuery(sqlNew);
```

The above replaces the uppercase 'NOTEPAD' with a lowercase version in the query before updating the stored query in the object.

Changing the Schema Name

Some schema and other information is also provided in this structure. For example, the schema name is provided within the statement data and can be explicitly updated. In the example below, the schema "products" is updated to "nyc_products":

```
if (change.getDefaultSchema().compareTo("products") == 0)
{
  change.setDefaultSchema("nyc_products");
}
```

A similar operation should be performed for any row-based changes. A more complete example can be found in Section 12.3.8, "dbrename.js Filter".

12.5.1.4.4. 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 [593] objects provided.

The following methods are supported for the RowChangeData object:

Method	Description
appendOneRowChange(rowChange)	Appends a single row change to the event, using the supplied OneRowChange [593] object.
getRowChanges()	Returns an array list of all the changes as OneRowChange [593] objects.
setRowChanges(rowChanges)	Sets the row changes within the event using the supplied list of OneRowChange objects.

For example, a typical row-based process will operate as follows:

```
if (d != null && d instanceof com.continuent.tungsten.replicator.dbms.RowChangeData)
 {
   rowChanges = d.getRowChanges();

   for(j = 0; j < rowChanges.size(); j++)
   {
     oneRowChange = rowChanges.get(j);
     // Do row filter
```

The OneRowChange [593] object contains the changes for just one row within the event. The class contains the information about the tables, field names and field values. The following methods are supported:

Method	Description
getAction()	Returns the row action type, i.e. whether the row change is an INSERT, UPDATE or DELETE
getColumnSpec()	Returns the specification of each column within the row change
getColumnValues()	Returns the value of each column within the row change
getSchemaName()	Gets the schema name of the row change
getTableName()	Gets the table name of the row change
setColumnSpec()	Sets the column specification using an array of column specifications
setColumnValues()	Sets the column values
setSchemaName()	Sets the schema name
setTableName()	Sets the table name

Changing Schema or Table Names

The schema, table and column names are exposed at different levels within the OneRowChange [593] object. Updating the schema name can be achieved by getting and setting the name through the getSchemaName() and setSchemaName() methods. For example, to add a prefix to a schema name:

```
rowchange.setSchemaName('prefix_' + rowchange.getSchemaName());
```

To update a table name, the getTableName() and setTableName() can be used in the same manner:

```
oneRowChange.setTableName('prefix_' + oneRowChange.getTableName());
```

Getting Action Types

Row operations are categorised according to the action of the row change, i.e. whether the change was an insert, update or delete operation. This information can be extracted from each row change by using the getAction() method:

```
action = oneRowChange.getAction();
```

The action information is returned as a string, i.e. `INSERT`, `UPDATE`, or `DELETE`. This enables information to be filtered according to the changes; for example by selectively modifying or altering events.

For example, `DELETE` events could be removed from the list of row changes:

```
for(j=0;j<rowChanges.size();j++)
{
  oneRowChange = rowChanges.get(j);
  if (oneRowChange.actionType == 'DELETE')
  {
    rowChanges.remove(j);
    j--;
  }
}
```

The `j--` is required because as each row change is removed, the size of the array changes and our current index within the array needs to be explicitly modified.

Extracting Column Definitions

To extract the row data, the `getColumnValues()` method returns the an array containing the value of each column in the row change. Obtaining the column specification information using `getColumnSpec()` returns a corresponding specification of each corresponding column. The column data can be used to obtain the column type information

To change column names or values, first the column information should be identified. The column information in each row change should be retrieved and/or updated. The `getColumnSpec()` returns the column specification of the row change. The information is returned as an array of the individual columns and their specification:

```
columns = oneRowChange.getColumnSpec();
```

For each column specification a `ColumnSpec` object is returned, which supports the following methods:

Method	Description
`getIndex()`	Gets the index of the column within the row change
`getLength()`	Gets the length of the column
`getName()`	Returns the column name if available
`getType()`	Gets the type number of the column
`getTypeDescription()`	
`isBlob()`	Returns true if the column is a blob
`isNotNull()`	Returns true if the column is configured as `NOT NULL`
`isUnsigned()`	Returns true if the column is unsigned.
`setBlob()`	Set the column blob specification
`setIndex()`	Set the column index order
`setLength()`	Returns the column length
`setName()`	Set the column name
`setNotNull()`	Set whether the column is configured as `NOT NULL`
`setSigned()`	Set whether the column data is signed
`setType()`	Set the column type
`setTypeDescription()`	Set the column type description

To identify the column type, use the `getType()` method which returns an integer matching the underlying data type. There are no predefined types, but common values include:

Type	Value	Notes
`INT`	4	
`CHAR` or `VARCHAR`	12	
`TEXT` or `BLOB`	2004	Use `isBlob()` to identify if the column is a blob or not
`TIME`	92	
`DATE`	91	
`DATETIME` or `TIMESTAMP`	92	

Type	Value	Notes
DOUBLE	8	

Other information about the column, such as the length, and value types (unsigned, null, etc.) can be determined using the other functions against the column specification.

Extracting Row Data

The `getColumnValues()` method returns an array that corresponds to the information returned by the `getColumnSpec()` method. That is, the method returns a complementary array of the row change values, one element for each row, where each row is itself a further array of each column:

```
values = oneRowChange.getColumnValues();
```

This means that index 0 of the array from `getColumnSpec()` refers to the same column as index 0 of the array for a single row from `getColumnValues()`.

getColumnSpec()	msgid	message	msgdate
getColumnValues()			
[0]	1	Hello New York!	Thursday, June 13, 2013
[1]	2	Hello San Francisco!	Thursday, June 13, 2013
[2]	3	Hello Chicago!	Thursday, June 13, 2013

This enables the script to identify the column type by the index, and then the corresponding value update using the same index. In the above example, the `message` field will always be index 1 within the corresponding values.

Each value object supports the following methods:

Method	Description
getValue()	Get the current column value
setValue()	Set the column value to the supplied value
setValueNull()	Set the column value to NULL

For example, within the `zerodate2null` sample, dates with a zero value are set to NULL using the following code:

```
columns = oneRowChange.getColumnSpec();
columnValues = oneRowChange.getColumnValues();
for (c = 0; c < columns.size(); c++)
{
  columnSpec = columns.get(c);
  type = columnSpec.getType();

  if (type == TypesDATE || type == TypesTIMESTAMP)
  {
    for (row = 0; row < columnValues.size(); row++)
    {
      values = columnValues.get(row);
      value = values.get(c);

      if (value.getValue() == 0)
      {
        value.setValueNull()
      }
    }
  }
}
```

In the above example, the column specification is retrieved to determine which columns are date types. Then the list of embedded row values is extracted, and iterates over each row, setting the value for a date that is zero (0) to be NULL using the `setValueNull()` method.

An alternative would be to update to an explicit value using the `setValue()` method.

12.5.2. Installing Custom JavaScript Filters

Once you have written your JavaScript filter, and ready to install it you need to follow the steps below. This will allow you to configure and apply the filter to your installation using the standard tpm procedure.

For this example, we will assume your new JavaScript file is called `number2binary.js`, and the filter has two additional boolean configuration properties 'roundup' and 'debug'

12.5.2.1. Step 1: Copy JavaScript files

By default, the software package will be contained in `/opt/continuent/software/tungsten-clustering-7.0.3-141`. Adjust the path in the examples accordingly if your environment differs.

The JavaScript file for your new filter(s) need copying to the following location:

```
/opt/continuent/software/tungsten-clustering-7.0.3-141/tungsten-replicator/samples/extensions/javascript
```

12.5.2.2. Step 2: Create Template Files

You need to create a template file which contains the location of the JavaScript file and the additional configuration properties with the appropriate default values.

Create a file called `number2binary.tpl` that contains the following:

```
replicator.filter.number2binary=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.number2binary.script=${replicator.home.dir}/samples/extensions/javascript/number2binary.js
replicator.filter.number2binary.roundup=true
replicator.filter.number2binary.debug=false
```

This tpl file needs to be copied into the following directory:

```
/opt/continuent/software/tungsten-clustering-7.0.3-141/tungsten-replicator/samples/conf/filters/default
```

12.5.2.3. Step 3: (Optional) Copy json files

If your filter uses json files to load configuration data, this needs to be copied into the `/opt/continuent/share` directory and also included in the tpl file created in Step 2. An example is as follows:

```
replicator.filter.{FILTERNAME}.definitionsFile=/opt/continuent/share/{FILTERNAME}.json
```

12.5.2.4. Step 4: Update Configuration

Now that all the files are in place you can include the custom filter in your configuration.

Any properties set with a default value in the tpl file, only need including if you wish to overwrite the default value

The following examples show how you can now include this in your tpm configuration:

Add the following to your `/etc/tungsten/tungsten.ini`:

```
svc-extractor-filters={existing filter definitions},number2binary
property=replicator.filter.number2binary.roundup=false
property=replicator.filter.number2binary.debug=true
```

Next, if you are installing the software for a new install:

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> tools/tpm install
```

If you are adding the filter to an existing installation, then you must issue the update from the software staging tree, as follows:

```
shell> cd /opt/continuent/software/tungsten-clustering-7.0.3-141
shell> tools/tpm update --replace-release
```

In the above examples we used the `svc-extractor-filters` [515] property for the extractor replicator. If you are applying your custom filters to your applier, then use `svc-applier-filters` [515] instead

Your custom filters are now installed in a clean and easy to manage process allowing you to use tpm for all future update processes

If there is a problem with the JavaScript filter during restart, the replicator will be placed into the `OFFLINE` state and the reason for the error will be provided within the replicator `trepsvc.log` log.

Chapter 13. Performance, Tuning and Testing

To help improve the performance of Tungsten Cluster, a number of guides and tuning techniques are provided in this chapter. This may involve parameters entirely within Tungsten Cluster, or changes and modifications to the parameters within the OS.

Tuning related to the Tungsten Replicator functionality

- Section 13.1, “Block Commit” — Increasing performance of replication solutions making use of block commit.

Tuning related to the network performance

- Section 13.2, “Improving Network Performance” — Increasing performance of networking between components by tuning OS parameters.

13.1. Block Commit

The replicator commits changes read from the THL and commits these changes in Replicas during the applier stage according to the block commit size or interval. These replace the single *replicator.global.buffer.size* parameter that controls the size of the buffers used within each stage of the replicator.

When applying transactions to the database, the decision to commit a block of transactions is controlled by two parameters:

- When the event count reaches the specified event limit (set by svc-applier-block-commit-size [515])
- When the commit timer reaches the specified commit interval (set by svc-applier-block-commit-interval [515])

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 [515].

The block commit interval can be controlled using the repl-svc-applier-block-commit-interval [515]. 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> cat /etc/tungsten/tungsten.ini

[defaults]
...
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> cat /etc/tungsten/tungsten.ini

[defaults]
...
repl-svc-applier-block-commit-interval=500
...
```

This would apply two commits every second, regardless of the block commit size.

When replicating to a data warehouse engine, particularly when using batch loading, such as Redshift, Vertica and Hadoop, larger block commit sizes and intervals may improve performance during the batch loading process:

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

[defaults]
...
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.

13.1.1. Monitoring Block Commit Status

The block commit status can be monitored using the trepctl status -name tasks [390] command. This outputs the *lastCommittedBlockSize* and *lastCommittedBlockTime* values which indicate the size and interval (in seconds) of the last block commit.

```
shell> trepctl status -name tasks
Processing status command (tasks)...
...
NAME                    VALUE
----                    -----
appliedLastEventId    : mysql-bin.000015:0000000000001117;0
appliedLastSeqno      : 5271
appliedLatency        : 4656.231
applyTime             : 0.066
averageBlockSize      : 0.500
cancelled             : false
commits               : 10
currentBlockSize      : 0
currentLastEventId    : mysql-bin.000015:0000000000001117;0
currentLastFragno     : 0
currentLastSeqno      : 5271
eventCount            : 5
extractTime           : 0.394
filterTime            : 0.017
lastCommittedBlockSize: 1
lastCommittedBlockTime: 0.033
otherTime             : 0.001
stage                 : q-to-dbms
state                 : extract
taskId                : 0
Finished status command (tasks)...
```

13.2. Improving Network Performance

The performance of the network can be critical when replicating data. The information transferred over the network contains the full content of the THL in addition to a small protocol overhead. Improving your network performance can have a significant impact on the overall performance of the replication process.

When using the Connector and client applications, improving the network performance will aid the overall performance of your application during both the client to connector, and connector to MySQL server connectivity.

The following network parameters should be configured within your `/etc/sysctl.conf` and can safely applied to all the hosts within your 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
```

13.3. 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 Java `OutOfMemory` exceptions, 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. An Applier pipeline might have a stage to read transactions from the Extractor and put them in the THL, a stage to read them back out of the THL into an in-memory queue, and a stage to apply those transactions to the Target. This model ensures high performance as the stages work independently. This streaming model is quite efficient and normally permits Tungsten to transfer even exceedingly large transactions, as the replicator breaks them up into smaller pieces called transaction fragments.

The pipeline model has consequences for memory management. First of all, replicators are doing many things at one, hence need enough memory to hold all current objects. Second, the replicator works fastest if the in-memory queues between stages are large enough that they do not ever become empty. This keeps delays in upstream processing from delaying things at the end of the pipeline. Also, it allows replicators to make use of block commit. Block commit is an important performance optimization in which stages try to commit many transactions at once on Targets to amortize the cost of commit. In block commit the end stage continues to commit transactions until it either runs out of work (i.e., the upstream queue becomes empty) or it hits the block commit limit. Larger upstream queues help keep the end stage from running out of work, hence increase efficiency.

Bearing this in mind, we can alter replicator behavior in a number of ways to make it use less memory or to handle larger amounts of traffic without getting a Java `OutOfMemory` error. You should look at each of these when tuning memory:

- Property `repl-java-mem-size` [496]. 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` [496] parameter.

- Property `repl-buffer-size` [477] 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.

- Property `repl-svc-applier-block-commit-size` [515] This parameter sets the block commit count in the final stage in the Applier pipeline. If you reduce the global buffer size, it is a good idea to set this to a fixed size, such as 10, to avoid reducing the block commit effect too much. Very low block commit values in this stage can cut update rates on Targets by 50% or more in some cases.

- Property `replicator.extractor.dbms.transaction_frag_size` [511]. This parameter controls the size of fragments for long transactions. Tungsten automatically breaks up long transactions into fragments. This parameter controls the number of bytes of binlog per transaction fragment. You can try making this value smaller to reduce overall memory usage if many transactions are simultaneously present. Normally however this value has minimal impact.

Finally, it is worth mentioning that the main cause of out-of-memory conditions in replicators is large transactions. In particular, Tungsten cannot fragment individual statements or row changes, so changes to very large column values can also result in OutOfMemory conditions. For now the best approach is to raise memory, as described above, and change your application to avoid such transactions.

The replicator commits changes read from the THL and commits these changes in Targets during the applier stage according to the block commit size or interval. These replace the single `repl-buffer-size` [477] 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 `repl-svc-applier-block-commit-size` [515])

- When the commit timer reaches the specified commit interval (set by `repl-svc-applier-block-commit-interval` [515])

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` [511] option. 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 interval (`repl-svc-applier-block-commit-interval` [515]) is interpreted as the interval in milliseconds if no suffix supplied. Suffix of s, m, h, and d for seconds, minutes, hours and days are supported.

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

```
[default]
...
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> cat /etc/tungsten/tungsten.ini
[default]
...
repl-svc-applier-block-commit-interval=500
...
```

This would apply two commits every second, regardless of the block commit size.

When replicating to a data warehouse engine, particularly when using batch loading, such as Redshift, Vertica and Hadoop, larger block commit sizes and intervals may improve performance during the batch loading process:

```
shell> cat /etc/tungsten/tungsten.ini
[default]
...
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.

13.4. Connector Memory Management

The memory model within the Tungsten Connector works as follows:

- Memory consumption consists of the core memory, plus the buffered memory used for each connection.
- Each connection uses the maximum size of an `INSERT`, `UPDATE` or `SELECT`, up to the configured size of the MySQL `max_allowed_packet` parameter.

For example, with 1000 concurrent connections, and a result or insert size of 1 MB, the memory usage will be 1 GB.

The default setting for the Tungsten Connector memory size is 256 MB. The memory allocation can be increased using tpm update and the `conn-java-mem-size` [478] option:

For example, add the following to your `/etc/tungsten/tungsten.ini`:

```
conn-java-mem-size=1024
```

13.5. Functional Testing

The following chapter includes a number of suggested functional tests that can be performed following installation in a Testing/POC environment

For a test to be successful, normal operations should be restored after every test. The default Primary server should regain Primary role with all replicators and data sources ONLINE

The tests are split into the following categories:

- Section 13.5.1, “Manual and Automatic Failover”
- Section 13.5.2, “Backup and Restore”
- Section 13.5.3, “Connectivity”
- Section 13.5.4, “Performance and Other Tests”

A PDF version of these tests can be downloaded by clicking here

13.5.1. Manual and Automatic Failover

Failover Test 1 - Administrator uses Tungsten to promote new Primary

Scenario	`shell> cctrl`
	`cctrl> use usa`
	`[LOGICAL] /usa> switch`

Expectation	• The Primary role will move a different server. Remaining servers will be reconfigured accordingly. • Examine how the application performs during the process • Data should be consistent

Failover Test 2 – Manually kill the Primary database process

Scenario	`shell> cctrl`
	`cctrl> use usa`
	`[LOGICAL] /usa> service Primary/mysql stop`

Expectation	• The database server stops and the Primary role is moved to another server. • Examine how the application performs during the process • Data should be consistent

Failover Test 3 – Remove power from the Primary database server

Scenario	Pull the power plug on the Primary server or run a restart command if that is not an option.
Expectation	• The database server stops and the Primary role is moved to another server. • Examine how the application performs during the process • Data should be consistent

Failover Test 4 – Manually kill a Replica database process

Scenario	`shell> cctrl`
	`cctrl> use usa`
	`[LOGICAL] /usa> service Replica1/mysql stop`

Expectation	• The Replica data source is marked as FAILED. • Examine how the application performs during the process

13.5.2. Backup and Restore

Backup Test 1 – Take a backup of a Replica and restore it to the same server

Scenario	`shell> cctrl`
	`cctrl> use usa`
	`[LOGICAL] /usa> datasource Replica1 backup`
	`[LOGICAL] /usa> datasource Replica1 restore`

Expectation	• The commands should complete successfully.

Backup Test 2 – Restore the backup to another server

Scenario	`shell> rsync -avze ssh /opt/continuent/backups/ Replica2:/opt/continuent/backups/`
	`shell> cctrl`
	`cctrl> use usa`
	`[LOGICAL] /usa> datasource Replica2 restore`

Expectation	• The commands should complete successfully.

Backup Test 3 – Take a backup of the Primary and restore to a Replica

Scenario	`shell> cctrl`
	`cctrl> use usa`
	`[LOGICAL] /usa> datasource Primary backup`
	`Primary> rsync -avze ssh /opt/continuent/backups/ Replica2:/opt/continuent/backups/`

	`[LOGICAL] /usa> datasource Replica2 restore`
Expectation	• The restore command should complete successfully, or. • Run `trepctl online -from-event ######:########` to bring the Replica ONLINE

13.5.3. Connectivity

Connectivity Test 1 – Connect to the connector and verify Primary host

Scenario	`shell> mysql -h`hostname` -P9999 -uapp -p -e"select @@hostname for update"`
Expectation	• The Primary hostname is returned.

Connectivity Test 2 – Verify access to Replicas

This test is only required if read/write splitting has been enabled. It should be run from a connector running on a server other than the Primary.

Scenario	`shell> mysql -h`hostname` -P9999 -uapp -p -e"select @@hostname"`
Expectation	• A hostname is returned. • Repeat the process on other hosts until a Replica hostname is returned

Connectivity Test 3 – Verify access to the Primary before and after a switch

Scenario	`shell> cctrl`
	`cctrl> use usa`
	`[LOGICAL] /usa> switch`

Expectation	• Run Test 1 again and verify the new Primary hostname is returned.

13.5.4. Performance and Other Tests

Performance Test 1 – Run a load test against the cluster

Scenario	Run load tests of some variety against the cluster to ensure the Connector and Replicator properly handle the load. Recommended load test tools are: • HammerDB. • sysbench
Expectation	Solution can handle 1TB of data with a minimum of 24k reads per minute and 1k writes per minute, over three tests. • Insert only from multiple locations • Read/write test • Heavy read test with few writes

Replicator Testing

Scenario	Evaluate system responsiveness in conjunction with performance tests (above). This is not a test of network speed
Expectation	Quick replication latency betweens regions (<100ms greater than the current ping latency)

Network Partition Testing

Scenario	Simulate a partitioned network (for example, by modifying security group rules in AWS), and continue to do reads and writes on multiple clusters for 30 minutes.
Expectation	After resolving the partition, clusters should resync.

Scenario	Create a similar network partition, and write the same record on both sides of the partition
Expectation	After resolving the partition, the replicators on both sides should report errors. Demonstrate possible resolutions:

	• Remove record on side, skip record on the other side, and bring replication back online. The record not removed will be replicated • Remove both records and skip both transactions • Modify records and bring replicators online

Appendix A. Release Notes

A.1. Tungsten Clustering 7.0.3 GA (4 Apr 2023)

Version End of Life. 28 Apr 2026

Release 7.0.3 contains a number of key bug fixes and improvements.

Behavior Changes

The following changes have been made to Tungsten Cluster and may affect existing scripts and integration tools. Any scripts or environment which make use of these tools should check and update for the new configuration:

- Installation and Deployment
  - Added `LimitNOFILE=infinity` in systemd scripts to workaround an issue with open files limit for clusters installed with systemd

    Issues: CT-2023
- Command-line Tools
  - Deprecated tungsten_get_position and tungsten_set_position scripts have been removed.

    Issues: CT-638
  - The tpm ssh-copy-cert command has been deprecated and removed in this release.

    Issues: CT-828
  - The `tps.pl` backup tool now shows the available command-line arguments when the TAB key is pressed.

    Issues: CT-1246
  - The tungsten_skip_all command (along with alias tungsten_skip_seqno) now shows the full pendingExceptionMessage instead of just pendingError, and the `More` choice shows the pendingErrorEventId and the pendingError.

    Issues: CT-2030
  - Enabling a custom backup script does not require it to be made the default option anymore, i.e. Setting `--repl-backup-method` to the custom backup script is not required anymore.

    Issues: CT-2039
  - The tungsten_find_seqno command is now a wrapper for tpm find-seqno.

    Issues: CT-2059
  - The tpm command now prints a warning when running as the root OS user during operations that make changes.

    Issues: CT-2064
  - The tpm report command with with `--extra` now shows the associated Connector listener ports and mode per port for when the connector mode is "listeners".

    Issues: CT-2087
- Tungsten Manager
  - Increased the default manager liveness heartbeat interval from 1s to 3s.

    Issues: CT-2095

Improvements, new features and functionality

- Command-line Tools
  - A new thl tail command has been added, allowing you to view the live THL changes as they are generated.

    Issues: CT-405

- A new command, tpm copy-keys, has been added to assist with the distribution of security-related files after an INI-based installation. This tool requires ssh access between the cluster nodes for the copy to work.

  Issues: CT-1910

- The tungsten_get_ports command now supports the `/opt/continuent/tungsten/tungsten-connector/conf/interfaces.json` file, if found.

  Issues: CT-2003

- The tungsten_get_ports command now use values based on values obtained from tpm query values when possible for the greatest output accuracy.

  Issues: CT-2008

- The tpm ask summary command has two new keys, `isSmartScale` and `isDirect`, which are also available individually on the command line.

  Issues: CT-2035

- When in Connector Proxy mode, the tpm report command now displays the Direct and SmartScale status via Channel 2: Connector to Database

  Issues: CT-2037

- Shell command tab completion for Tungsten-specific commands has been updated to support recent additions and changes.

  Issues: CT-2057

- Core Replicator

  - THL transfer protocol can now be changed dynamically. This can be done using the trepctl command tool

    ```
    shell> trepctl setdynamic -property replicator.thl.protocol.client.serialization -value DEFLATE
    Property replicator.thl.protocol.client.serialization successfully set to DEFLATE
    ```

    This can also be done through API v2

    ```
    curl --user tungsten:secret --insecure --request POST 'https://127.0.0.1:8097/api/v2/replicator/service/alpha/thl/inflight-protocol' -d
    '{
     "payloadType":"StringPayload",
     "payloadVersion":"1",
     "payload":
     {
     "value":"legacy"
     }
    }'

    {
     "payloadType": "StringPayload",
     "payloadVersion": "1",
     "payload":
     {
     "value": "THL inflight protocol is now set to legacy"
     }
    }
    ```

    Issues: CT-1862

- Tungsten Connector

  - Improved smart scale handling of session and global changes so they're all played against connected replica or in case of reconnection. As an example, `SET SESSION TRANSACTION READ ONLY` will be sent to both primary and replica connections so that a subsequent `SELECT @@SESSION.transaction_isolation` will return the expected value, even if executed on a replica.

    Issues: CT-1600

  - Added user-based authentication to existing host-based authentication to restrict connections per host and per user.

    For more details, see Section 7.6.10, “Host-based Authentication”

    Issues: CT-1978

  - A new advanced listeners feature has been added to the connector for even great control of client connectivity.

    For more details, see Section 7.12.3, “Advanced Listeners”

Issues: CT-2000

Bug Fixes

- Command-line Tools
  - Fixed rsync availability check in TPM to error out early when the executable is not found in the path

    Issues: CT-1028
  - The tungsten_get_status command now handles shunned nodes properly.

    Issues: CT-2026
  - The tpm ask keys command now lists all available keys properly.

    Issues: CT-2041
  - The tpm update command no longer aborts with a `Dataservice was NOT removed` error when service names with a hyphen are defined in the configuration. We strongly discourage the use of hyphens [-] in service names and recommend underscores [_] instead.

    Issues: CT-2052
  - The tpm validate-update command now performs the delete-service checks properly.

    Issues: CT-2054
  - The tungsten_find_events command no longer fails with error `Undefined subroutine &main::info`

    Issues: CT-2058
  - tpm now builds affinity the same way in bridge and proxy modes.

    Issues: CT-2060
  - Fixes an issue when tprovision would delete the MySQL PEM files in the `datadir`.

    Issues: CT-2067
  - The tpm diag command will now locate and kill any remaining tpm diag child processes on the calling host.

    Issues: CT-2068
  - Suppressed wrong tpm errors with autodetect hosts.

    Issues: CT-2072
  - Fixed a tpm issue trying to restore an empty policy.

    Issues: CT-2074
- Core Replicator
  - Fixed an issue that was occurring when an `ALTER USER` is applied on a mysql instance that uses lower case identifiers.

    Issues: CT-2024
  - Fixed an issue where an extracted JSON field could be invalid, with doubled commas in JSON arrays.

    Issues: CT-2049
  - Fixed an issue where status would fail if issued while a service was reconfiguring (in the middle of a setrole command for example).

    Issues: CT-2081
- Tungsten Connector
  - Fixed an issue with `CLIENT_QUERY_ATTRIBUTES` and/or `CLIENT_DEPRECATE_EOF` flags that would prevent fallback bridge mode to handle proxy connections properly.

    Issues: CT-2020

- Fixed an issue in Dynamic Active-Active environments where smartScale would always read from the site with write affinity, even if a different site is specified for read-affinity.

  Issues: CT-2040

A.2. Tungsten Clustering 7.0.2 GA (9 Dec 2022)

Version End of Life. 28 Apr 2026

Release 7.0.2 contains a number of key bug fixes and improvements.

Behavior Changes

The following changes have been made to Tungsten Cluster and may affect existing scripts and integration tools. Any scripts or environment which make use of these tools should check and update for the new configuration:

- Command-line Tools
  - The various user-xxxx.log files are no longer generated

    Issues: CT-1914
  - The check_tungsten.sh was deprecated in release 6.1.18 and has now been removed from this release.

    Issues: CT-1939
  - Both the tungsten_get_ports and tpm report commands have been updated to use the ss OS command when the netstat OS command is unavailable or deprecated.

    Issues: CT-2007
- Core Replicator
  - `repl_svc_extractor_multi_frag_service_detection` is now turned ON by default. Event shards are determined at extraction time. With fragmented events, the shard cannot be determined by only reading the first fragment, but needs to check the last fragment as well. With this setting turned OFF, there is no issue with pipelines that don't need it, i.e. no parallel apply downstream replicas. However, as this is done at extract time, THL contains this information, and adding or changing a replica using parallel apply could introduce issues.

    > **Note**
    >
    > It can be disabled if you see a performance overhead but this should be done with caution. For Aurora<>Aurora Active/Active deployments it is essential that this property be left ON.

    Issues: CT-1959
- Tungsten Connector
  - Improved tungsten show processlist by running underlying commands in parallel.

    Issues: CT-1569
- Core Clustering
  - Changed the behavior of the connector and manager upon local site failover:

    No longer sending write traffic to the remote site unless the local site is fully offline. In case of local failover, the connector will now pause connections until a new primary is elected. This will avoid risks of out-of-order apply after local failover

    Issues: CT-2009
- Tungsten Manager
  - A failsafe shunned cluster (Caused by a network split) will be auto recovered after the network connection is re-established.

    Issues: CT-241

Improvements, new features and functionality

- Command-line Tools

- rsync is now an option in tprovision, in addition to using xtrabackup and mysqldump. To use rsync, specify `-m rsync`.

  Using rsync by default will provision a replica in 2 passes:

  - The first pass will live copy (seed) the replica from the source.
  - The second pass will quiesce the source and run the rsync again, resulting in shorter down time than a single pass rsync

  Issues: CT-338

- A new `-c` option is now available with some trepctl commands that can be used in conjunction with the `-r` option to indicate the number of times to refresh before automatically terminating. For example, the following command:

  ```
  shell> trepctl perf -r 3 -c 10
  ```

  Will refresh the output every 3 seconds, 10 times.

  Issues: CT-679

- The trepctl status will now show the last known applied seqno and latency.

  This information is stored on disk at regular intervals (10s minimum) so as not to overload the replicator, therefore the value can be shown as slightly old dependant on when the status command was issued.

  By default, this feature is disabled. It can be enabled by setting the following parameter in the configuration :

  ```
  svc-applier-last-applied-write-interval=20
  ```

  This will write current position to disk every 20 seconds. This information is also exported by the Prometheus exporter.

  If the service is online, it will display the current value (the same as appliedLastSeqno and appliedLatency)

  ```
  shell> trepctl status
  Processing status command...
  NAME VALUE
  ---- -----
  appliedLastEventId : mysql-bin.000017:0000000151329854;70
  appliedLastSeqno : 999
  appliedLatency : 347707.0
  ...
  lastKnownAppliedLatency: 347707.0
  lastKnownAppliedSeqno : 999
  ...
  ```

  Issues: CT-1823

- The tungsten_merge_logs command now supports the `--before TIMESTAMP` and `--after TIMESTAMP` filters

  Issues: CT-1869

- Added a new log file (`tungsten-replicator/log/data-drift.log`) for data drift messages, i.e. :

  - an update statement was logged on primary, but did not update any row on replica
  - a delete statement was logged on primary, but did not delete any row on replica

  Issues: CT-1873

- The tpm ask summary command now provides the coordinator host and the isCoordinator boolean if the Manager is running on that node.

  Also, tpm ask now supports direct calls to coordinator, `\{isCoordinator|iscoordinator}` and `\{isBridgeMode|isBridge|bridge|isbridge|isbridge-mode}`.

  Issues: CT-1874

- The tungsten_generate_haproxy_for_api and tpm generate-haproxy-for-api commands now support using connector hosts in the backend definitions via `-c`, and extra backend flags to the backend hosts lines using `-f`.

  Issues: CT-1909

- The tungsten_generate_haproxy_for_api and tpm generate-haproxy-for-api commands no longer call the `Perl Data::Dumper` module.

  Issues: CT-1915

- The tungsten_reset_manager command now supports the ability to simply print out the path or paths to be cleared, one per line via the `-l` or `--list` arguments.

  Issues: CT-1917

- The tmonitor command now accepts cli args to specify the ports and will auto-configure the ports if they have been changed via the Tungsten configuration.

  Issues: CT-1919

- The tpm command calls to `glob` have been improved to be more strict and compliant.

  Issues: CT-1940

- The tpm ask stages and tpm ask allstages commands have been added to display the Replicator stages for the current node (stages) and the stages for each role (allstages).

  Issues: CT-1943

- The tpm ask command has five new variables available: `dsrole` & `dsstate` for the current datasource, and `trrole` & `trstate` for the current replicator, and `nodeinfo` which displays all 4 of the new variables.

  Issues: CT-1944

- A new standalone status script has been added called tungsten_get_status that shows the datasources and replicators for all nodes in all services along with seqno and latency.

  Issues: CT-1962

- The tpm report command now prints the hostname and listener ports where available when using the `--extra|-x` option or the new `--ports` option.

  Issues: CT-1969

- A new `dsctl` option has been added to the thl command, and a new `-event` option added to thl list

  Full details and examples of these new options can be found here

  Issues: CT-2012

- Core Replicator

  - Added a new feature that enables pausing a replicator stage for some amount of time.

    This will pause the given stage for 100 seconds.:

    ```
    trepctl pause -stage thl-to-q -time 100
    ```

    This will pause the stage indefinitely (or until restart, etc) Add -y to avoid the prompt message whether you are sure.

    ```
    trepctl pause -stage thl-to-q
    ```

    For the previous 2 commands, running a pause command again will override the previous command.

    This will resume the suspended stage (Note that if the stage is not paused, this will have no effect):

    ```
    trepctl resume -stage thl-to-q
    ```

    > **Note**
    >
    > Please note this pause does not survive a replicator restart or a service offline/online.

    Issues: CT-1912

  - Per-service tuning of the replicator thl directory is now possible for multi-service replicator-only installs as well as for clustering. The given value should be the base directory, to which tungsten will add the service name. For example, the following entry in the tungsten.ini:

    ```
    [alpha]
    ...
    ...
    thl-directory=/drv1/thl
    ...
    ```

Would result in the THL being placed in /drv1/thl/alpha

> **Note**
>
> Update of thl directory is only available when tpm is called from the staging installation directory, NOT from the running directory.

Issues: CT-1927

- A new replicator role (thl-applier) has been added to allow a replicator service to apply its locally available THL, without pulling from a remote host

  Issues: CT-1936

- Added a way to configure the maximum number of rows that can be grouped together when applying row based events for multiple insert or delete statements.

  For these properties to be in effect, you must ensure that `optimize-row-events=true` is either explicitly set in your configuration, or not present (since it will be enabled by default)

  For example, the following settings will limit the number of inserted or deleted rows applied at once to 10:

  ```
  optimize-row-events-limit-insert-rows=10
  optimize-row-events-limit-delete-rows=10
  ```

  The default values if not specified will be 50 for inserts and 100 for deletes. Note that for deletes to be optimized, the affected table MUST have a single column PK.

  Issues: CT-1980

- Tungsten Connector

  - The connector graceful-stop command now supports systemd service manager properly. The connector stop command now takes an optional argument that will make it a graceful stop. If connector stop is run without the parameter, it will stop the connector immediately. If a positive number of seconds is passed, it will wait, at most, this timeout for connections to disconnect (refusing new connections), after which it will force close all connections and shutdown the connector. connector graceful-stop behavior is unchanged: without the parameter, the connector will wait "forever" for connections to disconnect. A positive timeout in seconds can be passed to sever connections after the given delay

    Issues: CT-1921

  - Two new commands have ben introduced to cctrl to provide better insight and control of the connectors. These are:

    - datasource <dsname> connections [-l]

      This command displays the current number of connections running to the given node through connectors.

    - datasource <dsname> drain [optional timeout]

      This command will prevent new connection to be made to the given data source, while ongoing connection remain untouched.

    For more detils on the usage and options, see Section 9.1.3.5, “datasource Command”

    Issues: CT-1949

  - Added logging configuration example to print load balancers activity

    Issues: CT-1966

- Tungsten Manager

  - Added a new option to TPM `manager-replicator-offline-timeout=<timeout_in_sec>` that configures the timeout for the manager to wait until the replicator goes offline. When parallel applier is in use the default timeout was too low, so it's now user configurable so that it can be adjusted to suit different topologies. If not supplied, the default is 180 (3 minutes). This value should be sufficent in most use cases.

    Issues: CT-1892

- API

  - New logs files have been added for the REST API, these are as follows:

    - service_logs/connector-api.log

- service_logs/manager-api.log
- service_logs/replicator-api.log

Issues: CT-1983

Bug Fixes

- Installation and Deployment
  - ddlscan, dsctl and tungsten_send_diag are now added to the `aliases.sh` script.

    Issues: CT-813
  - Fixes issues where fixed properties and filters passed to tpm in service stanzas were not being configured correctly

    Issues: CT-1463
  - No longer using Tanuki wrapper functionality to print jvm version, which was creating defunct java processes at startup, now using internal code.

    Issues: CT-1876
  - The tpm install and tpm update commands now properly support the --thl-ports option for cross-site subservices.

    Issues: CT-1953
- Command-line Tools
  - The tungsten_skip_seqno command no longer fails when `-i` is specified, and now properly filters using `--filter` when there is a long error message.

    Issues: CT-1877
  - The tpm command now allows any case for section entries (i.e. [alpha_FROM_beta]) in the INI files.

    Issues: CT-1879
  - The tpm diag command now passes when the nodename defined in the `tungsten.ini` is the shortname, and DNS returns the FQDN.

    Issues: CT-1908
  - The tpm diag command now gathers the `mysql.log` file when SSL is enabled in the server.

    Issues: CT-1920
  - > **Note**
    >
    > This fix was released in Tungsten Clustering and Tungsten Replicator 7.0.2 Build 161.

    The tpm command checks for the existence of the mysql command-line client when installing/upgrading. The process will no longer abort with an error on non-MySQL targets such as heterogeneous replicator appliers, or Active-Witness hosts.

    Issues: CT-1924, CT-2018
  - Fixes an issue that prevented dsctl from connecting to MySQL if SSL was enabled.

    Issues: CT-1928
  - The tpm mysql command will now gracefully handle being run on a non-database node.

    Issues: CT-1946
  - The tpm mysql command no longer aborts with an access denied error on CentOS 6.

    Issues: CT-1977
  - Both TungstenAPI and tpasswd now properly update .passwords.store.orig backup file so that proper manipulation of passwords won't trigger tpm update failure

    Issues: CT-1981

- Backup and Restore
  - The cluster_backup script will no longer backup a replica if the replicator is in an ERROR state.

    Issues: CT-1036
- Core Replicator
  - Fixes an issue that would prevent a service from going offline at a specified time (trepctl online -until-time) when parallel apply is enabled. This is a rework of CT-1243.

    Issues: CT-1684
  - Fixed a possible issue when recovering an old primary as a replica after failover when parallel apply is enabled, that could lead the replica to be unable to come online and require a reprovisioning of this replica.

    Issues: CT-1890
  - Fixes an issue that prevented geometry datatypes with SRID from being replicated.

    Issues: CT-1904
  - Fixed an issue where filtered events would trigger a useless update to the service trep_commit_seqno table while it is overwritten anyway once the last statement of the applied event is done, just prior to committing the whole block.

    Issues: CT-1931
  - Fixed an issue where the replicator would hang after applying a `DROP TABLE` event, that originally failed on the primary, but got logged into the binlog.

    Issues: CT-1973
  - Fixed a parsing issue that would prevent the replicator from correctly detecting a `CREATE TABLE` statement with `START TRANSACTION`

    Issues: CT-1987
- Filters
  - Fixed an issue where the `dropsqlmodes` filter would fail to remove invalid sql modes from a multi-statement event

    Issues: CT-1993
- Tungsten Connector
  - Connector now auto detects default authentication plugin by retrieving MySQL data source variable `default_authentication_plugin` rather than just using MySQL server version

    Issues: CT-1926
  - No longer printing warning `"SequenceException: Parents differ"` when canceling connections of a composite data service

    Issues: CT-1964
  - Fixed connector logging configuration to show hostname and class printing logs

    Issues: CT-1965
- Tungsten Manager
  - The cctrl command datasource <ds> slave now sets the replicator role correctly. Previously, only the datasource role would change.

    Issues: CT-1882
  - A bug has been fixed that, in a few very rare cases, would allow replicas to continue to pull and apply THL from a failed primary whilst a failover was in the process of electing a new primary. This resulted in failovers being unable to complete fully. Whilst the new primary would be online and functioning, existing replicas in the cluster could experience errors due to THL discrepencies between the old and new primary nodes.

    Issues: CT-1986
- API
  - Fixed REST API `/api/v2/manager/control/service/{service}/switch` call. It will not switch anymore to a shunned node.

Issues: CT-796

- Calls to `/api/v2/manager/cluster/status` now return properly when a peer cluster is fully offline or unreachable.

  Issues: CT-1945

- The REST API call `/api/v2/manager/control/service/{service}/datasource/{datasource}/slave` now sets the role of the replicator correctly.

  Issues: CT-1975

A.3. Tungsten Clustering 7.0.1 GA (13 June 2022)

Version End of Life. 28 Apr 2026

Release 7.0.1 is the first minor release for the v7 release. This release contains a number of new features, minor bug fixes and a few key behavior changes, but importantly contains a critical bug fix for environments that use mixed-case table and database names, that could cause replication failures.

Customers already running v7.0.0 in production enviornments should upgrade to this release as soon as possible

Behavior Changes

The following changes have been made to Tungsten Cluster and may affect existing scripts and integration tools. Any scripts or environment which make use of these tools should check and update for the new configuration:

- Installation and Deployment
  - A better error message is now provided when keystore password and enclosed private key password don't match.

    Issues: CT-1853

- Command-line Tools
  - The cctrl output now displays the SSL status of the group communication and the connection to the database.

    ```
    $ cctrl
    Tungsten Clustering 7.0.0
    nyc: session established, encryption=true, authentication=true
    jgroups: [un]encrypted, database: [un]encrypted
    ```

    Issues: CT-1785, CT-1832

  - The tpm ask summary command is now sorted by key when the Perl `Tie::IxHash` module is installed.

    Issues: CT-1820

  - The following commands which call cctrl now support the pass-through `-t seconds` timeout argument which specifies how long cctrl will wait for a connection to the Manager process before aborting:
    - check_tungsten_latency
    - check_tungsten_online
    - check_tungsten_policy
    - check_tungsten_services
    - tungsten_show_processlist
    - zabbix_tungsten_latency
    - zabbix_tungsten_online
    - zabbix_tungsten_progress
    - zabbix_tungsten_services

    Issues: CT-1822

  - The tpm diag command now gathers the output of the tpm ask summary command

    Issues: CT-1827

- The tmonitor command now calls sudo internally as needed, removing the need for the tmonitor command to be run with sudo for the install and remove sub-commands.

  Issues: CT-1846

- The tpm generate-haproxy-for-api default starting port is now 8091.

  Issues: CT-1865

- The tungsten_find_orphaned command now places the log file in `$CONTINUENT_ROOT/service_logs/`, not `/tmp/`

  Issues: CT-1866

- Core Replicator

  - The default value for the tpm property `repl-svc-fail-on-zero-row-update` has been changed from `warn` to `stop`

    > **Warning**
    >
    > This is a significant change in behavior and could cause replicators, specifically cross-site replicators, in an Active/Active environment to stop and error more frequently.
    >
    > To maintain original behavior you will need to add this property with the value of `warn` to your configuration, providing you fully understand the inherent risk of ignoring such warnings.

    Issues: CT-1856

- API

  - The manager REST API `/api/v2/manager/status` call now returns information regarding to security.

    Issues: CT-1830

Known Issue

The following issues are known within this release but not considered critical, nor impact the operation of Tungsten Cluster. They will be addressed in a subsequent patch release.

- Installation and Deployment

  - After starting up Tungsten components, a `defunct` process for each running component can be found in the process listing.

    Whilst this does not cause any issues, it could generate unnecessary alerts for customers monitoring.

    The cause has been identified and affects version 7.0.0 and 7.0.1. This will be fixed in the next 7.0.2 release.

    Issues: CT-1876

Improvements, new features and functionality

- Command-line Tools

  - Added the ability to turn auto recovery on or off dynamically, removing the need to run tpm update.

    This is done by running the following command:

    ```
    shell> trepctl -service servicename setdynamic -property replicator.autoRecoveryMaxAttempts -value <number>
    ```

    > **Note**
    >
    > The service must be offline before changing the property

    Issues: CT-1088

  - The new command tungsten_upgrade_manager will help clear specific cached SSL values, to be used only when specified by Continuent Support.

    Issues: CT-1788

- A new tpm report sub-command has been added. By default, tpm report will generate a security report for all available communication channels on a per-node basis. The purpose of tpm report is to provide easy access to all of the settings that pertain to a specific topic. The default (and only) topic at this time is the security stance. More topics will be added over time.

  Issues: CT-1793, CT-1857

- The tapi command now supports the creation and modification of multiple APIv2 admin users. Two new cli args have been added: `--create-user` and `--create-pass`.

  Issues: CT-1801

- The tapi command now auto-detects the protocol, either http or https

  Issues: CT-1803

- Added connector<->manager link encryption status in the output of router * status command in cctrl

  Issues: CT-1811

- Added JMX encryption and authentication status to trepctl -verbose and connector command outputs

  Issues: CT-1813

- The tapi command will now output service names with the new option `--includeServiceName`, or if there is more than one service name found.

  Issues: CT-1860

- Core Replicator
  - The thl list command now displays an approximative field size in bytes for row-based replication.

    Issues: CT-1847

- Tungsten Connector
  - Added a connector mode command to print which mode the connector is running in, either "bridge" or "proxy"

    Issues: CT-1799

- Security
  - Replaced official log4j library with a secured version in which all vulnerable classes have been removed. This prevents exposing the software following a (user) misconfiguration of the log4j properties

    Issues: CT-1810

- API
  - Added a way to retrieve service properties through the API, for example, to check a specific service parameter, run the following API call :

    ```
    shell> curl https://127.0.0.1:8097/api/v2/replicator/service/alpha/properties?filter=replicator.global.db.sslEnabled
    {
     "payloadType": "PropertiesListPayload",
     "payloadVersion": "1",
     "payload": {
     "replicator.global.db.sslEnabled": "false"
     }
    }
    ```

    Issues: CT-1802

  - Added new top level REST API call to display RMI SSL status, run the following API call :

    ```
    shell> curl https://127.0.0.1:8097/api/v2/rmiSecurity
    ```

    Issues: CT-1812

Bug Fixes

- Installation and Deployment
  - The tpm update command now properly handles v7 -> v7 upgrades.

Issues: CT-1815

- deployall script now properly displays the executable prefix for restarting services, typically mm_treplicator in Multi-Site/Active-Active replicator-only installs, as well as the full path to component binaries

  Issues: CT-1835

- tpm would fail on keystores containing special characters in alias names.

  Issues: CT-1852

- Command-line Tools

  - Worked around a ruby issue where DNS name resolution was sometimes preferred to file-based resolution in TPM. The following logic now applies: first trying file-based host name resolution, upon failure try DNS, then through ping command

    Issues: CT-1653

  - The tpm update command now returns clusters to the proper mode (i.e. *`MAINTENANCE`* or *`AUTOMATIC`*) for staging-method upgrades (INI not affected).

    Issues: CT-1784

  - The tpm policy command now returns the proper policy in a composite cluster. There is also a new cli argument, `--all`, to show the policy for all cluster services.

    Issues: CT-1787

  - The tpm update command no longer aborts during a staging deployment when the actual hostname does not match what is configured for the node names in tpm

    Issues: CT-1791

  - Improves the tpm diag command so that it waits 2 seconds for replicator thread dump to complete.

    Issues: CT-1792

  - Fixed an Undefined subroutine`call error in tapi --create

    Issues: CT-1798

  - The tpm connector command now properly passes the -e arguments to the mysql command.

    Issues: CT-1816

  - tpm now properly reports errors upon timeout executing commands. Typically, when a host is down, when upgrading or installing, tpm will now properly report that pinging this host failed.

    Issues: CT-1819

  - The tpm update command now properly determines services to be deleted.

    Issues: CT-1825

  - TPM would fail to validate data sources when using `ssl-capath [directory]` rather than `ssl-ca [file]` in `my.cnf`.

    This fixes only the validation part, automated truststore creation by TPM will not be possible at this point, a full `tungsten-truststore.ts` will have to be passed through `--java-truststore-path`

    Issues: CT-1826

  - The tapi command now auto-switches protocol (http or https) properly.

    Issues: CT-1829

  - replicator start offline would not properly pass the offline argument to systemd scripts when installed through deployall.

    Issues: CT-1836

  - undeployall script would not properly uninstall `systemd` services on some distributions including amazon linux.

    Issues: CT-1845

- Backup and Restore
  - Fixed an issue with latest xtrabackup 8.0.28

    Issues: CT-1838
  - During the backup/restore process the MySQL certificates are kept intact.

    Issues: CT-1841
  - Restore will now succeed even if only one incremental backup has been created.

    Issues: CT-1842
  - Incremental backups can now be restored more then once.

    Issues: CT-1850
- Core Replicator
  - Fixed the EnumToString and pkey filters to renew their database connections (every hour by default). This can be changed with the following properties:

    ```
    property=replicator.filter.enumtostring.reconnectTimeout=3600
    property=replicator.filter.pkey.reconnectTimeout=3600
    ```

    Issues: CT-1786
  - tpm update would fail with message `"The host is configured to be '' but it is currently operating as a 'master'"` when a dynamic replicator configuration file is found

    Issues: CT-1848
  - Fixes issues with PrimaryKeyFilter:
    - It could fail from correctly fetching primary key columns for tables with mixed upper and lower case letters in table names
    - Alter user statements could make it fail

    Issues: CT-1861
  - Fixed a `NullPointerException` that could happen if a replicator was put offline during startup. This had mostly no consequence except the exception being logged.

    Issues: CT-1863
- Tungsten Connector
  - Fixed the output of connector cluster-status by adjusting log4 appender.

    Issues: CT-1778
  - An exception could occur while forcing shutdown of router gateways.

    Issues: CT-1779
- Tungsten Manager
  - Fixed an issue where the command system summary issued in cctrl would hang the managers in the cluster.

    Issues: CT-1849

A.4. Tungsten Clustering 7.0.0 GA (29 Mar 2022)

Version End of Life. 28 Apr 2026

Release 7.0.0 is a major release introducing many new features including a fully documented API. There are a large number of bug fixes and improvements in all areas of the product, and a number of key behavior changes, most significantly being Security is now enabled by default.

The following changes have been made to Tungsten Cluster and may affect existing scripts and integration tools. Any scripts or environment which make use of these tools should check and update for the new configuration:

- Installation and Deployment
  - An RPM update now issues `--replace-release`

    Issues: CT-1708
- Command-line Tools
  - The tpm diag command now uses `tar czf` instead of the `zip` command to compress the gathered files. The `zip` command is no longer a prerequisite for tpm diag.

    Issues: CT-1253
  - tungsten_set_position has been deprecated and no longer available in this release. dsctl should be used instead.

    tungsten_provision_slave has now been renamed to tprovision

    Issues: CT-1302
  - The tungsten_post_process script functionality has been merged into the tpm post-process command. The tungsten_post_process script remains as a shell wrapper for tpm post-process.

    Issues: CT-1314
  - The tungsten_find_orphaned script now creates a log file every time it runs which is stored in the configured temporary directory (`/tmp/` by default, tpm query values temp_directory). This is to allow for easier troubleshooting and visibility during automatic execution.

    Issues: CT-1447
  - tpm now accepts chrony as a valid time synchronization software

    Issues: CT-1462
  - The tpm diag command now uses the ss (socket status) command in place of netstat on SUSE and other operating systems that have deprecated netstat.

    Issues: CT-1483
  - The tpm diag command now gathers the `/etc/os-release` file when located. Also now using the ip command on systems where ifconfig and/or route is deprecated.

    Issues: CT-1496
  - Changed output of thl purge command when no lower and upper bounds are given from 'Deleting events where' to 'Deleting all events'.

    Issues: CT-1738
- Backup and Restore
  - `--user` and `--password` options have been added to the following scripts:
    - cluster_backup
    - check_tungsten_backups
    - tungsten_monitor

    If `manager-rest-api-authentication=true` (the default if not explicitly disabled) then these two new options must be supplied otherwise the scripts will fail with the following error: "`ERROR >> Manager REST API authentication needed. Please specify the user name and password.`"

    The internal ruby `api.rb` module can now also handle the `manager-rest-api-ssl=true` (the default if not explicitly disabled) and will use https instead of http to access the REST API

    Issues: CT-1311
  - From v8 of xtrabackup, the `--stream=tar` option was removed, meaning that backups could fail if using the newer release of the Percona tools

In this release, the backups will now create the backup and then manually compress it

> **Warning**
>
> This change will increase the required disk space for backups to allow the post-backup compression to complete.

Issues: CT-1346

- tungsten_provision_slave has now been renamed to tprovision.

  Issues: CT-1436

- Additional messaging has been added to the output displayed when running tprovision.

  Issues: CT-1689

- Core Replicator
  - Replicator will now check after extracting a mysql `STOP_EVENT` whether a new binlog file was created, actually handling this event as a `ROTATE EVENT`.

    This decreases the time that would be needed after `STOP_EVENT` to extract a new event..

    Issues: CT-1349

- Tungsten Connector
  - Optimized transaction parsing by removing SQL transaction start recognition: in-transaction state is found in server status flags.

    Issues: CT-1698

- Tungsten Manager
  - A new logfile, `console.log` is now generated in the `$CONTINUENT_ROOT/tungsten/tungsten-manager/logs` directory which contains all output displayed via cctrl. This file will provide Continuent Support with more valuable information when assisting to diagnose support cases.

    Issues: CT-1499

- Other Issues
  - Log files for each component now have the same date and time stamp format.

    Issues: CT-1669

- API
  - API calls triggering configuration changes are protected by a flag, `i-am-sure=true`, in order to avoid unwanted, potentially dramatic, configuration changes. This applies to:
    - configuration/module/servicesmap
    - reset
    - offline
    - online
    - onhold
    - addDataService
    - addDataSource

    Issues: CT-1317

Known Issue

The following issues are known within this release but not considered critical, nor impact the operation of Tungsten Cluster. They will be addressed in a subsequent patch release.

- Installation and Deployment
  - After starting up Tungsten components, a `defunct` process for each running component can be found in the process listing.

    Whilst this does not cause any issues, it could generate unnecessary alerts for customers monitoring.

    The cause has been identified and affects version 7.0.0 and 7.0.1. This will be fixed in the next 7.0.2 release.

    Issues: CT-1876
- Command-line Tools
  - The check_tungsten_online command returns a Replicator offline error on active witness hosts.

    Issues: CT-1783
  - The tpm policy command may return the incorrect policy value for composite clusters.

    As a result of this known issue, the tungsten_reset_manager command will also have issues on composite clusters because it calls tpm policy.

    Issues: CT-1787

Improvements, new features and functionality

- Installation and Deployment
  - Two new tpm options have been included as part of the new API in this release.

    The options are used for setting the API admin user credentials and are as follows:

    - `rest-api-admin-user`
    - `rest-api-admin-pass`

    Issues: CT-1327
  - Support now included for MariaDB 10.3+

    Issues: CT-1276, CT-1433
  - Support has been added for Java 17 LTS

    Issues: CT-1706
- Command-line Tools
  - A new tpm option delete-service is now available to simplify the removal of clusters and/or replicator services.

    Issues: CT-210, CT-327, CT-1275
  - Prometheus exporters mysqld_exporter and node_exporter are now included with the distribution packages.

    A new command line tool tmonitor is now available for the management and testing of external Prometheus exporters (node and mysqld), and for the testing of internal exporters (Manager, Connector and Replicator).

    Issues: CT-960
  - A new tpm option `purge-thl` and a new script tungsten_purge_thl have been added to allow easier and more intelligent THL purging across all nodes in a topology.

    This allow you to purge THL files based on the following rules:

    - Gather the last applied seqno from all Replica nodes and take the lowest one
    - Find the current THL file which contains that seqno, then locate the previous one
    - Construct a thl purge command to remove thl thru the last seqno in the prev file

    The default behavior is to display the needed commands for the admin to execute manually.

    Issues: CT-1273

- tpm diag now collects routing table information via `route -n` , and has two new command-line arguments: `--include` and `--groups`.

  `--include` specifies a comma-separated list of subroutines to include. Any gather subroutine not listed will be skipped.

  `--groups` specifies a comma-separated list of subroutine groups to include. Any group not listed will be skipped.

  Issues: CT-1322

- A new sub-command has been added, tpm generate-haproxy-for-api. This read-only action will read all available INI files and dump out corresponding `haproxy.cfg` entries with properly incrementing ports; the composite parent will come first, followed by the composite children in alphabetical order.

  The tungsten_generate_haproxy_for_api script functionality has been merged into the tpm generate-haproxy-for-api command. The tungsten_generate_haproxy_for_api script remains as a shell wrapper for tpm generate-haproxy-for-api.

  Issues: CT-1342

- tungsten_send_diag now supports a new command-line argument, `--cleanup`, which will cause the removal of the diagnostic archive file generated using the `--diag` argument.

  Issues: CT-1360

- The tungsten_reset_manager command is now able to restart the Manager process when the `--start` or `-s` argument is passed in.

  Issues: CT-1401

- With the release of APIv2, a new cli tool has been introduced to allow easier access called tapi.

  In addition, the vast majority of Tungsten cli tools have been updated to optionally use the APIv2 interface when desired.

  The Nagios and Zabbix checks are also available via APIv2 using the tapi tool.

  Issues: CT-1454

- The tungsten_purge_thl command is now a wrapper for the tpm purge-thl command.

  Issues: CT-1488

- The tmonitor command now has better help text and more options to ease usage, including `--filter` to allow easy viewing of the tmonitor test output.

  Issues: CT-1585

- A new option to print the merged logs to `STDOUT` has been added to tungsten_merge_logs [`--stdout|-O`].

  The tpm command suite now properly supports the `--profile` argument to specify a Tungsten json configuration file in place of the installed `tungsten.cfg`.

  Issues: CT-1680

- The tapi command now supports the `--affinity` argument which will display all Connector-specific affinity settings, along with `--connectorstatus` to show all.

  Issues: CT-1700

- The `cctrl.log` file is now accessible from the `$CONTINUENT_ROOT/service_logs` directory

  Issues: CT-1727

- A new command [error] has been added to trepctl to output a full stack trace of the last error, if any.

```
shell> trepctl -service <serviceName> error

Event application failed: seqno=10 fragno=0 message=Table hr.regions not found in database. Unable to generate a valid statement.
com.continuent.tungsten.replicator.applier.ApplierException: Table hr.regions not found in database. Unable to generate a valid statement.
at com.continuent.tungsten.replicator.applier.JdbcApplier.getTableMetadata(JdbcApplier.java:582)
at com.continuent.tungsten.replicator.applier.JdbcApplier.fillColumnNames(JdbcApplier.java:494)
at com.continuent.tungsten.replicator.applier.JdbcApplier.getColumnInformation(JdbcApplier.java:1236)
at com.continuent.tungsten.replicator.applier.MySQLApplier.applyOneRowChangePrepared(MySQLApplier.java:418)
at com.continuent.tungsten.replicator.applier.JdbcApplier.applyRowChangeData(JdbcApplier.java:1460)
at com.continuent.tungsten.replicator.applier.JdbcApplier.apply(JdbcApplier.java:1576)
at com.continuent.tungsten.replicator.applier.ApplierWrapper.apply(ApplierWrapper.java:100)
at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.apply(SingleThreadStageTask.java:871)
```

```
at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.runTask(SingleThreadStageTask.java:601)
at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.run(SingleThreadStageTask.java:185)
at java.base/java.lang.Thread.run(Thread.java:834)
```

Issues: CT-1747

- Backup and Restore
  - Added support for mariabackup. Configurable tpm option:

    ```
    backup-method=mariabackup
    or
    backup-method=mariabackup-incremental
    ```

    Issues: CT-1100
  - Fixes an issue where tprovision could remove the SSL certs for MySQL on reprovision

    Issues: CT-1323
- Core Replicator
  - It is now possible to compress and/or encrypt THL on disk. For more information on using these features see https://docs.continuent.com/tungsten-clustering-7.0/thl-compress-encrypt.html

    Issues: CT-630
  - The replicator will now be able to handle new SQL_MODES available in later releases of MySQL and MariaDB, these are as follows:
    - MySQL: `TIME_TRUNCATE_FRACTIONAL`
    - MariaDB: `TIME_ROUND_FRACTIONAL, SIMULTANEOUS_ASSIGNMENT`

    Issues: CT-1362
  - In-Flight THL Compression is now available.

    For full details on enabling this feature, refer to this page

    Issues: CT-1420
- Tungsten Connector
  - Audit Logging is now available in the Connector: it allows for logging data transferred between client application and MySQL servers to a file, database or socket.

    For more details and steps to enable, see https://docs.continuent.com/tungsten-clustering-7.0/connector-advanced-audit-logging.html

    Issues: CT-78
  - Added support for TLSv1.3.

    Note that TLS 1.3 was introduced in jdk 8u272. So tpm will only set it by default for java 8u272 and later versions. It is still possible to force the protocol to be used via tpm flag `--tls-enabled-protocols=TLSv1.xxx`

    Issues: CT-1367
  - A new Dynamic Active/Active feature has been added within the Proxy.

    This new Proxy mode is for Composite Active/Active clusters only, and allows you to specifically configure writes to be directed to a single cluster by the use of the tpm `connector-write-affinity` flag.

    For full details on this feature, and configuring it, see [link to be added]

    Issues: CT-1540
  - connector drain [opt-timeout] has been introduced as an alias to existing connector graceful-stop command and will prevent new connections, then shutdown the connector after an optional delay.

    The `opt-timeout` parameter [seconds] can be specified to limit the wait before stopping the connector. Not passing this parameter implies infinite wait.

    Issues: CT-1644

- `connector-reset-when-affinity-back` is now available with proxy mode.

  Issues: CT-1763

- Security
  - Tungsten can now install on default CentOS and RedHat 8 with tightened security settings thanks to support of TLSv1.3

    Issues: CT-1359

  - Tungsten software now undergoes a rigorous security scan during QA. We also check included open-source/3rd-party software.

    Issues: CT-1579

- Monitoring
  - A number of new metrics have been added to the Prometheus exporters.

    Issues: CT-1266, CT-1615

- Platform Specific Deployments
  - ARM 64 bit processor support added (linux aarch64)

    > **Note**
    >
    > Note that at time of release, there is currently no xtrabackup binary available for ARM.

    Issues: CT-1619, CT-1620

- Other Issues
  - General improvements in MySQL 8.0 support.

    Issues: CT-1346

  - IPv6 host addresses are now fully supported.

    Can be enabled with the following configuration property:

    ```
    prefer-ip-stack="6"
    ```

    By default, IPv4 is enabled, which equates to the value of "4" in the above property.

    Issues: CT-1537

Bug Fixes

- Installation and Deployment
  - Default systemd configuration files for Tungsten components no longer specifies the tungsten group to execute the command. This will prevent file access issues when tungsten user belongs to several groups.

    Issues: CT-1550

  - When services are deployed with systemd and MySQL could not start due to an error, tpm would not be able to later start MySQL

    Issues: CT-1734

- Command-line Tools
  - In certain cases, tprovision would not be able to find the binary log position of the backup when taken from a primary. This has been fixed.

    Issues: CT-1085

  - Fixes a bug in tprovision when using xtrabackup version 8, due to changes in xtrabackup binaries.

    Issues: CT-1248

  - The tpm connector command now handles special characters in the password string.

    Issues: CT-1258

- The tpm update command will now exit with an error if any files not owned by the configured Tungsten OS user are found in the Tungsten installation directory.

  For example, if the OS user is tungsten and the installation directory is `/opt/continuent`, containing the file `/opt/continuent/thl/archived_thl.zip` owned by `root` would cause something like the following error to be produced, and tpm update would exit:

```
Foreign-owned files found!

Located files in the Tungsten installed directory /opt/continuent
that are not owned by the Tungsten OS user (tungsten):

/opt/continuent/thl/archived_thl.zip

Please change the ownership of these files to OS user "tungsten"
using the chown command as root via sudo, then rerun the `tpm update` command.

For example:
shell> sudo chown -R tungsten:tungsten /opt/continuent
```

  Issues: CT-1260

- The tpm diag command now behaves correctly on Connector-only nodes, where previously it would try to gather Manager and Replicator-specific items.

  Issues: CT-1284

- Fixes a security issue within the tpm diag command.

  Issues: CT-1295

- The number of created and active connections could be incorrect when listed from a composite data service

  Issues: CT-1312

- tungsten_send_diag no longer prints an error about `Use of uninitialized value $diagArgs in concatenation`.

  Issues: CT-1354

- The tpm command no longer prints an error when run with no other command-line arguments.

  Issues: CT-1373

- The tpm command no longer aborts with a `Use of uninitialized value error` when a stray `tungsten.cfg` file exists under `$CONTINUENT_ROOT`

  Issues: CT-1394

- Fixes a monitoring bug with users using `caching_sha2_password`.

  Issues: CT-1406

- tprovision (formerly tungsten_provision_slave) may fail to provision if the MySQL data directory was not accessible to the tungsten user.

  Issues: CT-1475

- The tpm generate-haproxy-for-api command no longer fails on CentOS 8.

  Issues: CT-1484

- The tmonitor command no longer fails on Debian 9 and Ubuntu.

  Issues: CT-1485

- All tpm sub-commands now handle command-line arguments more intelligently.

  Issues: CT-1487

- The tpm purge-thl command now handles command-line arguments more intelligently.

  Issues: CT-1489

- The tpm diag command now properly collects the system information file on Debian systems.

  Issues: CT-1492

- Database monitoring logs are now reporting the correct error number and SQL state when database errors occur.

  Issues: CT-1497

- The tpm update command now handles updates/upgrades more gracefully when the previous version did not have the latest tpm framework.

  Issues: CT-1506

- The tpm update command will now properly remove composite services.

  Issues: CT-1519

- The tungsten_find_orphaned command no longer fails with an `'Can't exec "/bin/sh": Argument list too long'` error when there are too many THL files to parse.

  Issues: CT-1545

- The tpm ask command no longer calls `Data::Dumper` when it is not available.

  Issues: CT-1626

- tpm now parses the MYSQL SSL related setting correctly.

  Issues: CT-1662

- Fixes an issue where the deployall command would create a root owned wrapper.log in the ./tools directory.

  Issues: CT-1664

- When MySql services where badly installed, some distribution could show a `"not-found"` status within systemctl, confusing tpm

  Issues: CT-1677

- The tpm command now communicates properly when there is no INI configuration file or staging-method `deploy.cfg` configuration defined.

  Issues: CT-1712

- The tpm diag command now handles Multi-Site/Active-Active topologies better.

  Issues: CT-1718

- tungsten_monitor.rb script no longer uses sudo to send emails if the configuration doesn't allow it.

  Issues: CT-1737

- The tpm diag command now handles zero-length `mysqld.log` files gracefully.

  Issues: CT-1740

- Backup and Restore

  - When running the cluster_backup in a Active/Active environment, when setting `require_master_backup` to false, the script would still attempt to backup the Primary as it would scan the wrong sub-service and incorrectly identify the Relay node as a candidate.

    Issues: CT-1280

  - Fixed an issue where an xtrabackup generated by the replicator would fail to be restored using trepctl restore command.

    Issues: CT-1575

- Core Replicator

  - The replicator metadata cache will now correctly handle table names when `lower_case_table_names=1` is set in the MySQL configuration.

    Issues: CT-651

  - When using parallel apply, the replicator would error with a Foreign Key constraint error if statements were issued against two or more objects that shared the same name, but with different case sensitivity, for example:

```
mysql> create table testtable;
```

```
mysql> drop table testtable;
mysql> create table TestTable;
```

Issues: CT-1259

- A change in the way MySQL logs `CREATE TABLE AS SELECT` in the Binary Logs from v8.0.20 onwards, meant these transactions would previously fail.

  Warning

  Whilst these statements will now replicate, it mut be noted that in the event of a failure during the data load, the initial CREATE statement won't be rolled back, and therefore care must be taken when using this type of DDL.

  Note

  This only affects customers using MySQL v8.0.20+ running with `ROW` based replication. An alternative workaround to ensure correct rollback on failure, would be to run the statement with `STATEMENT` based replication for the session. This will also provide better performance for larger tables.

  Issues: CT-1301

- Fixes occurences of `NullPointerException` that would occur when bringing the replicator online before MySQL was started.

  Issues: CT-1348

- For row based events, SQL modes were not displayed in the THL output. This is now fixed.

  Issues: CT-1440

- When connecting to a THL server, a client will now connect to the next available host in its THL uri, if the first does not have the sequence number that the client requires. The client will then fail only if none of the hosts from the uri can provide the needed sequence number.

  Issues: CT-1558

- Fixed an issue when using Parallel apply that would show a `NullPointerException` in case an event could either not be found or be corrupted in THL. This will now display a correct message `Missing or corrupted event from storage`

  Issues: CT-1722

- Fixed an issue where trepctl was leaving JMX connections opened.

  Issues: CT-1752

- Added more debug information for detecting possible hanging connections while a THL client connects to the THL server. Also, added socket timeout for the connection initialization

  Issues: CT-1760

- Filters

  - Includes previously missing template file to enable easy configuration of the `dbrename` filter.

    Issues: CT-1350

  - The `BidiRemoteSlaveFilter` could fail to correctly flag fragmented events in unprivileged environments (Aurora, for example) In such an environment (multi-active, unprivileged database access), a new setting was introduced to force extraction process to read ahead to the last fragment to detect the service name (false by default). Enabled with `repl_svc_extractor_multi_frag_service_detection=true`

    Issues: CT-1351

- Tungsten Connector

  - By default, the connector will no longer transparently reconnect underlying connections to database servers when the data service changes.

    This will prevent the following case: in Composite Active/Active topologies, a given connection starts to write data to a site. The site fails, connection gets reconnected the other site and resumes writing. However the data written to the 1st site has not reached the 2nd site, thus data will not be consistent.

    Default is to reject reconnections that follow a write operation (`RW_STRICT` connection or SmartScale after a write) and to allow reconnection after a read operation (~~`RO_RELAXED` or SmartScale after a read~~) which translates to ~~`--connector-allow-cross-site-reconnects-for-writes=false`~~ and `--connector-allow-cross-site-reconnects-for-reads=true`

It is still possible to get the previous behavior (reconnecting transparently connections cross-site) by specifying both `--connector-allow-cross-site-reconnects-for-writes=true` and `--connector-allow-cross-site-reconnects-for-reads=true`, at your own risk

Issues: CT-1265

- With `@direct` r/w splitting in active/active configurations, the connector was not correctly redirecting connections to an active site after re-joining the cluster following a failure.

  Issues: CT-1400

- The Connector now properly retries specific MySQL commands when possible: `INIT_DB`, `CHANGE_USER`, `STATISTICS` and prepared statement functions.

  Issues: CT-1480

- Connector now forbids non-ssl connections when mysql server has `require_secure_transport=ON`

  Issues: CT-1666

- Connector now mirrors the MySQL default `connect_timeout` by retrieving it from the primary when starting up. This timeout will apply to all connections made from the connector to MySQL servers.

  This setting can be over-ridden by using the following tpm property

  ```
  property=connectTimeout=VALUE
  ```

  If *`VALUE`* set to autodetect, this value will mirror the MySQL `connect_timeout` system variable. Set to 0 for infinite timeout.

  Issues: CT-1726

- Connector no longer requests cross-site services information from manager. In addition to removing extra network traffic, this cures a problem of connections counted twice in data source connection statistics

  Issues: CT-1775

- Tungsten Manager

  - Fixes a `NullPointerException` error (NPE) in the manager logs.

    Issues: CT-1132

  - Fixes an edge case bug that would allow a Composite Active/Active cluster to contain 2 Primary nodes.

    Issues: CT-1474

  - Fixes a manager-internal connection check during recovery that was not properly using SSL when required.

    Issues: CT-1661

  - Early initialisation of the REST API sometimes caused the manager to hang and fail to startup correctly on new installations.

    Issues: CT-1725

  - Fixes an issue when the manager would wait longer than the timeout for replicator purge on a failover.

    Issues: CT-1733

  - Fixes an issue where datasource <datasourcename> welcome would fail to welcome a manually failed composite datasource.

    Issues: CT-1771

- Security

  - log4j libraries updated to v2.17.1 specifically to mitigate risk of exposure to the 0-day vulnerbaility detected in log4j v2.14

    Issues: CT-1703

Appendix B. Prerequisites

Before you install Tungsten Cluster, there are a number of setup and prerequisite installation and configuration steps that must have taken place before any installation can continue. Section B.2, “Staging Host Configuration” and Section B.3, “Host Configuration” must be performed on every host within your chosen cluster or replication configuration. Additional steps are required to configure explicit databases, such as Section B.4, “MySQL Database Setup”, and will need to be performed on each appropriate host.

B.1. Requirements

B.1.1. Operating Systems Support

The following Operating Systems are supported for installation of the various Tungsten components and are part of our regular QA testing processes. Other variants of Linux may work at your own risk, but use of them in production should be avoided and any issues that arise may not be supported; if in doubt we recommend that you contact Continuent Support for clarification. Windows/MAC OS is NOT supported, however appropriate Virtual Environments running any of the supported distributions listed would be supported, although only recommended for Development/Testing environments.

Virtual Environments running any of the supported distributions listed are supported, although only recommended for Development/Testing environments.

The list below also includes EOL dates published by the providers and should be taken into consideration when configuring your deployment

Table B.1. Tungsten OS Support

Distribtion	Published EOL	Notes
Amazon Linux 2	30th June 2024	
Amazon Linux 2023	June 2029	
CentOS 7	30th June 2024	
Debian GNU/Linux 10 (Buster)	June 2024	
Debian GNU/Linux 11 (Bullseye)	June 2026	
Debian GNU/Linux 12 (Bookworm)	June 2028	
Oracle Linux 8.4	July 2029	
Oracle Linux 8.9	July 2032	
RHEL 7	30th June 2024	
RHEL 8.4.0	31st May 2029	
RHEL 9	31st May 2032	
Rocky Linux 8	31st May 2029	
Rocky Linux 9	31st May 2032	See Note Below
SUSE Linux Enterprise Server 15	21st June 2028	
Ubuntu 20.04 LTS (Focal Fossa)	April 2025	
Ubuntu 22.04 LTS (Canonical)	April 2027	
Ubuntu 24 LTS		

Warning

Limitation with Rocky 9 distribution: Under high load with many short living connections, socket timeouts will appear in the Tungsten Connector, leading to potential disruption of service.

B.1.2. Database Support

Unless stated, MySQL refers to the following variants:

- MySQL Community Edition
- MySQL Enterprise Edition
- Percona MySQL

Version Support Matrix

Table B.2. MySQL/Tungsten Version Support

Database	MySQL Version	Tungsten Version	Notes
MySQL	5.7	All non-EOL Versions	Full Support
MySQL	8.0.0-8.0.34	6.1.0-6.1.3	Supported, but does not support Partitioned Tables or the use of `binlog-transaction-compression=ON` introduced in 8.0.20
MySQL	8.0.0-8.0.34	6.1.4 onwards	Fully Supported.
MySQL	8.4	8.0.0 onwards	Fully Supported.
MariaDB	10.0, 10.1	All non-EOL Versions	Full Support
MariaDB	10.2, 10.3	6.1.13-6.1.20	Partial Support. See note below.
MariaDB	Up to, and including, 10.11	7.x	Full Support

Known Issue affecting use of MySQL 8.0.21+

In MySQL release 8.0.21 the behavior of `CREATE TABLE ... AS SELECT ...` has changed, resulting in the transactions being logged differenly in the binary log. This change in behavior will cause the replicators to fail in versions up to and including v6.1.21

If you are using MySQL 8.0.21 and later it is recommended that you upgrade to v6.1.22 or later

Alternatively, the workaround for this will be to split the action into separate `CREATE TABLE ...` followed by `INSERT INTO ... SELECT FROM...` statements.

If this is not possible, then you will need to manually create the table on all nodes, and then skip the resulting error in the replicator, allowing the subsequent loading of the data to continue.

MariaDB 10.3+ Support

Full support for MariaDB version 10.3 has been certified in v7.0.0 onwards of the Tungsten products.

Version 6.1.13 onwards of Tungsten will also work, however should you choose to deploy these versions, you do so at your own risk. There are a number of issues noted below that are all resolved from v7.0.0 onwards, therefore if you choose to use an earlier release, you should do so with the following limitations acknowledged:

- tungsten_find_orphaned may fail, introducing the risk of data loss (Fixed in v6.1.13 onwards)
- SSL from Tungsten Components TO the MariaDB is not supported.
- Geometry data type is not supported.
- Tungsten backup tools will fail as they rely on xtrabackup, which will not work with newer release of MariaDB.
- tpm might fail to find correct mysql configuration file. (Fixed in 6.1.13 onwards)
- MariaDB specific event types trigger lots of warnings in the replicator log file.

MySQL "Innovation" Releases

In 2023, Oracle announced a new MySQL version schema that introduced "Innovation" releases. From this point on, patch releases would only contain bug fixes and these would be, for example, the various 8.0.x releases, whereas new features would only be introduced in the "Innovation" releases, such as 8.1, 8.2 etc (Along with Bug Fixes)

"Innovation" releases will be released quartlery, and Oracle aim to make an LTS release every two years which will bundle all of the new features, behavior changes and bug fixes from all the previous "Innovation" releases.

Oracle do not advise the use of the "Innovation" releases in a production enviornment where a known behavior is expected to ensure system stability. We have chosen to follow this advice and as such we do not certify any release of Tungsten against "Innovation" releases for

use in Production. We will naturally test against these releases in our QA environment so that we are able to certify and support the LTS release as soon as is practical. Any modifications needed to support an LTS release will not be backported to older Tungsten releases.

For more information on Oracles release policy, please read their blogpost here

B.1.3. RAM Requirements

RAM requirements are dependent on the workload being used and applied, but the following provide some guidance on the basic RAM requirements:

- Tungsten Replicator requires 2GB of VM space for the Java execution, including the shared libraries, with approximate 1GB of Java VM heapspace. This can be adjusted as required, for example, to handle larger transactions or bigger commit blocks and large packets.

  Performance can be improved within the Tungsten Replicator if there is a 2-3GB available in the OS Page Cache. Replicators work best when pages written to replicator log files remain memory-resident for a period of time, so that there is no file system I/O required to read that data back within the replicator. This is the biggest potential point of contention between replicators and DBMS servers.

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

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. This space will be required on each server.
- Each component creates log files within this staging directory. Log rotation should be configured to ensure these files do not grow out of control.
- The THL files required for installation are based on the size of the binary logs generated by MySQL. THL size is typically twice the size of the binary log. This space will be required on each machine in the cluster. The retention times and rotation of THL data can be controlled, see Section D.1.5, “The `thl` Directory” for more information, including how to change the retention time and move files during operation.

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

Because the replicator reads and writes information using buffered I/O in a serial fashion, there is no random-access or seeking.

> **Important**
>
> Disk Full issues are generally the biggest cause of cluster failures resulting in support calls. It is strongly advised that sufficient moniotring is put in place to alert on disk space related issues so that they can be remedied before an outage occurs.
>
> A disk full situation would not cause a database failover as this could result in data loss.

B.1.5. Java Requirements

All components of Tungsten are certified with Java using the following versions:

- 8
- 11
-
-
- Java 9, 10 and 13 have been tested and validated but certification and support will only cover Long Term releases.

> **Important**
>
> There are a number of known issues in earlier Java revisions that may cause performance degradation, high CPU, and/or component hangs, specifically when SSL is enabled. It is strongly advised that you ensure your Java version is one of the following MINIMUM releases:
>
> - Oracle JRE 8 Build 261
> - OpenJDK 8 Build 222
> - Oracle JRE 11 Build 8

All versions from 8u265, excluding version 13 onwards, contain a bug that can trigger unusually high CPU and/or system timeouts and hangs within the SSL protocol. To avoid this, you should add the following entry to the `wrapper.conf` file for all relevant components. This is included by default from version 6.1.15 onwards of all Tungsten products.

`wrapper.conf` can be found in the following path *`{INSTALLDIR}`*/tungsten/tungsten-*`component`*/conf, for example: `/opt/continuent/tungsten/tungsten-manager/conf`

```
wrapper.java.additional.{next available number}=-Djdk.tls.acknowledgeCloseNotify=true
```

For example:

```
wrapper.java.additional.16=-Djdk.tls.acknowledgeCloseNotify=true
```

After editing the file, each component will need restarting

Important

If your original installation was performed with Java 8 installed, and you wish to upgrade to Java 11, you will need to issue tools/tpm update --replace-release on all nodes from within the software staging path.

This is to allow the components to detect the newer Java version and adjust to avoid calls to functions that were deprecated/renamed between version 8 and version 11.

B.1.6. Cloud Deployment Requirements

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

Instance Types/Configuration

Attribute	Guidance	Amazon Example	GCP Example
Instance Type	Instance sizes and types are dependent on the workload, but larger instances are recommended for transactional databases.	`m4.xlarge` or better	`c3-standard-8` or better
Instance Boot Volume	Use block, not ephemeral storage.	EBS	
Instance Deployment	Use standard Linux distributions and bases. For ease of deployment and configuration, the use of Ansible, Puppet or other script based solutions could be used.		

Development/QA nodes should always match the expected production environment.

AWS/EC2 Deployments

- Use Virtual Private Cloud (VPC) deployments, as these provide consistent IP address support.
- When using Active Witnesses, a `micro` instance can be used for a single cluster. For composite clusters, an instance size larger than `micro` must be used.
- Multiple EBS-optimized volumes for data, using Provisioned IOPS for the EBS volumes depending on workload:

Parameter	tpm Option	tpm Value	MySQL `my.cnf` Option	MySQL Value
`/` (root)				
MySQL Data	`datasource-mysql-data-directory` [486]	`/volumes/mysql/data`	`datadir`	`/volumes/mysql/data`
MySQL Binary Logs	`datasource-log-directory` [486]	`/volumes/mysql/binlogs`	`log-bin`	`/volumes/mysql/binlogs/mysql-bin`
Transaction History Logs (THL)	`thl-directory` [519]	`/volumes/mysql/thl`		

Recommended Replication Formats

- `MIXED` is recommended for MySQL Primary/Replica topologies (e.g., either single clusters or primary/data-recovery setups).

- `ROW` is strongly recommended for Composite Active/Active setups. Without `ROW`, data drift is a possible problem when using `MIXED` or `STATEMENT`. Even with `ROW` there are still cases where drift is possible but the window is far smaller.

B.1.7. Docker Support Policy

B.1.7.1. Overview

> **Note**
>
> The details in this section refer to all versions 6.x and 7.x of Tungsten.

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

While it is possible to install and run Continuent Tungsten products (i.e. Clustering/Replicator/etc.) inside Docker containers, there are many reasons why this is not a good idea.

Version 8.0.0 of Tungsten introduced the Tungsten Operator, with full Kubernetes support for Tungsten Clustering. See the Tungsten Operator documentation for more details.

B.1.7.2. Background

As background, every database node in a Tungsten Cluster runs at least three (3) layers or services:

- MySQL Server (i.e. MySQL Community or Enterprise, MariaDB or Percona Server)
- Tungsten Manager, which handles health-checking, signaling and failover decisions (Java-based)
- Tungsten Replicator, which handles the movement of events from the MySQL Primary server binary logs to the Replica databases nodes (Java-based)

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

B.1.7.3. Current State

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

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

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

- Our product is designed to run on a full Linux OS. By design Docker does not have a full init system like SystemD, SysV init, Upstart, etc... This means that if we have a process (Replicator, Manager, Connector, etc...) that process will run as PID 1. If this process dies the container will die. There are some solutions that let a Docker container to have a 'full init' system so the container can start more processes like ssh, replicator, manager, ... all at once. However this is almost a heavyweight VM kind of behavior, and Docker wasn't designed this way.
- Requires a mutable container – to use Tungsten Clustering inside a Docker container, the Docker container must be launched as a mutable Linux instance, which is not the classic, nor proper way to use containers.
- Our services are not designed as "serverless". Serverless containers are totally stateless. Tungsten Cluster and Tungsten Replicator do not support this type of operation.
- Until we make the necessary changes to our software, using Docker as a cluster node results in a minimum 1.2GB docker image.
- Once Tungsten Cluster and Tungsten Replicator have been refactored using a microservices-based architecture, it will be much easier to scale our solution using containers.
- A Docker container would need to allow for updates in order for the Tungsten Cluster and Tungsten Replicator software to be re-configured as needed. Otherwise, a new Docker container would need to be launched every time a config change was required.
- There are known i/o and resource constraints for Docker containers, and therefore must be carefully deployed to avoid those pitfalls.
- We test on CentOS-derived Linux platforms.

B.1.7.4. Summary

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

- Unsupported at this time for all products (i.e. Clustering/Replicator/etc.) for version 6.x and 7.x
- Use at your own risk

- For a fully support solution, consider Tungsten Operator v8.0.0+. See the Tungsten Operator documentation for more details.

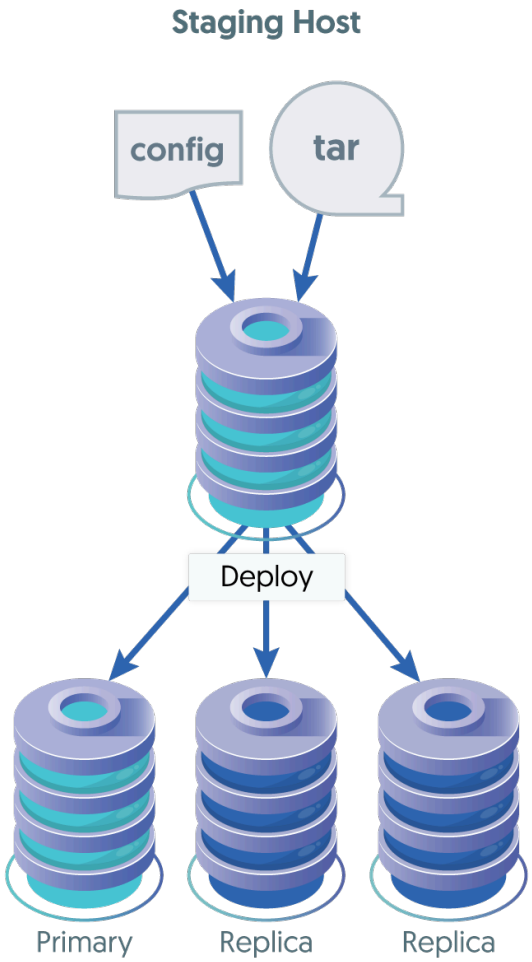
B.2. Staging Host Configuration

> **Important**
>
> The use of a Staging Host, referred to as the Staging Deployment Method, is no longer recommended and is being deprecated, with removal in a future release.
>
> For backwards compatability, the staging deployment method is still supported however it is not recommended for new deployments, and existing deployments should be converted to the INI method of deployment.

The staging host will form the base of your operation for creating your topology. The primary role of the staging host is to hold the software, and to install, transfer, and initiate the services on each of the nodes. The staging host can be a separate machine, or a machine that will be part of the cluster.

To use a staging host, you will need to configure SSH on each machine within the topology which allows the tpm tool to connect and perform the necessary installation and setup operations to create your 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 with a common SSH key. This will allow both the staging server, and each host 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 the Tungsten software. The recommended choice for MySQL installations is to create a new user, `tungsten`. You will need to create this user on each host. You can create the new user using adduser:

```
shell> sudo adduser tungsten
```

You can add the user to the `mysql` group adding the command-line option:

```
shell> sudo usermod -G mysql -a tungsten
```

2. Login as the `tungsten` user:

```
shell> su - tungsten
```

3. Create an SSH key file, but do not configure a password:

```
tungsten:shell> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/tungsten/.ssh/id_rsa):
Created directory '/home/tungsten/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/tungsten/.ssh/id_rsa.
Your public key has been saved in /home/tungsten/.ssh/id_rsa.pub.
The key fingerprint is:
e3:fa:e9:7a:9d:d9:3d:81:36:63:85:cb:a6:f8:41:3b tungsten@staging
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|             .   |
|            . .  |
|        S .. +   |
|       . o .X .  |
|        .oEO + . |
|       .o.=o. o  |
|      o=+..    . |
+-----------------+
```

This creates both a public and private keyfile; the public keyfile will be shared with the hosts to allow hosts to connect to each other.

4. Within the staging server, profiles for the different 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.

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. Operating System Version Support

For a full list of supported Operating System environments, see Table B.1, “Tungsten OS Support”.

B.3.2. Creating the User Environment

Create a new Tungsten user that will be used to manage and install the Tungsten software. The recommended choice for MySQL installations is to create a new user, `tungsten`. You will need to create this user on each host. You can create the new user using adduser:

```
shell> sudo adduser tungsten
```

You can add the user to the `mysql` group adding the command-line option:

```
shell> sudo usermod -G mysql -a tungsten
```

Login as the `tungsten` user:

```
shell> su - tungsten
```

Optionally, create an SSH key file, but do not configure a password:

```
tungsten:shell> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/tungsten/.ssh/id_rsa):
Created directory '/home/tungsten/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/tungsten/.ssh/id_rsa.
Your public key has been saved in /home/tungsten/.ssh/id_rsa.pub.
The key fingerprint is:
e3:fa:e9:7a:9d:d9:3d:81:36:63:85:cb:a6:f8:41:3b tungsten@staging
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|             .   |
|            . .  |
|        S .. +   |
|       . o .X .  |
|        .oEO + . |
|       .o.=o. o  |
|      o=+..    . |
+-----------------+
```

This creates both a public and private keyfile; the public keyfile can then be shared with all hosts in the deployment. See Section B.3.3.2, "SSH Configuration" for steps.

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. You can check this by using ulimit:

```
shell> ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
file size               (blocks, -f) unlimited
max locked memory       (kbytes, -l) unlimited
max memory size         (kbytes, -m) unlimited
open files                      (-n) 256
pipe size            (512 bytes, -p) 1
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 709
virtual memory          (kbytes, -v) unlimited
```

The system should be configured to allow a minimum of 65535 open files. You should configure both the `tungsten` user and the database user with this limit by editing the `/etc/security/limits.conf` file:

```
tungsten    -    nofile    65535
mysql       -    nofile    65535
```

In addition, the number of running processes supported should be increased to ensure that there are no restrictions on the running processes or threads:

```
tungsten    -    nproc    8096
mysql       -    nproc    8096
```

You must logout and log back in again for the ulimit changes to take effect.

You may also need to set the limit settings on the specific service if your operating system uses the systemctl service management framework. To configure your file limits for the specific service:

1. Copy the MySQL service configuration file template to the configuration directory if it does not already exist:

```
shell> sudo cp /lib/systemd/system/mysql.service /etc/systemd/system/
```

> **Note**
>
> Please note that the filename `mysql.service` will vary based on multiple factors. Do check to be sure you are using the correct file. For example, in some cases the filename would be `mysqld.service`

2. Edit the proper file used above, and append to or edit the existing entry to ensure the value of `infinity` for the key `LimitNOFILE`:

   ```
   LimitNOFILE=infinity
   ```

   This configures an unlimited number of open files, you can also specify a number, for example:

   ```
   LimitNOFILE=65535
   ```

3. Reload the systemctl daemon configuration:

   ```
   shell> sudo systemctl daemon-reload
   ```

4. Now restart the MySQL service:

   ```
   shell> service mysql restart
   ```

> **Warning**
>
> On Debian/Ubuntu hosts, limits are not inherited when using su/sudo. This may lead to problems when remotely starting or restarting services. To resolve this issue, uncomment the following line within `/etc/pam.d/su`:
>
> ```
> session required pam_limits.so
> ```

Integration with AppArmor

Make sure that Apparmor, if configured, has been enabled to support access to the `/tmp` directory for the MySQL processes. For example, add the following to the MySQL configuration file (usually `/etc/apparmor.d/local/usr.sbin.mysqld`):

```
/tmp/** rwk
```

B.3.3. Configuring Network and SSH Environment

The hostname, DNS, IP address and accessibility of this information must be consistent. For the installation to operate successfully, each host must be identifiable and accessible to each other host, either by name or IP address.

Individual hosts within your deployment must be reachable and must conform to the following:

- Do not use the `localhost` or `127.0.0.1` addresses.
- Do not use Zeroconf (`.local`) addresses. These may not resolve properly or fully on some systems.
- The server hostname (as returned by the hostname) must match the names you use when configuring your service.
- The IP address that resolves on the hostname for that host must resolve to the IP address (not `127.0.0.1`). The default configuration for many Linux installations is for the hostname to resolve to the same as `localhost`:

  ```
  127.0.0.1 localhost
  127.0.0.1 host1
  ```

- Each host must be able to resolve the address for all the other hosts in the. To prevent errors within the DNS system causing timeouts or bad resolution, all hosts 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 must resolve to the same IP address on each node. For example, if `host1` resolves to `192.168.0.69` on `host1`, the same IP address must be returned when looking up `host1` on the host `host2`.

To double check this, you should perform the following tests:

1. Confirm the hostname:

   ```
   shell> uname -n
   ```

   > **Warning**
   >
   > The hostname cannot contain underscores.

2. Confirm the IP address:

   ```
   shell> hostname --ip-address
   ```

3. Confirm that the hostnames of the other hosts 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 deployment:

   ```
   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 deployment.

> Witness Hosts
>
> Tungsten Cluster includes support for verifying the network status using a *witness* host.
>
> Active Witness Hosts operate as standalone managers, and therefore require the same rights and requirements as a standard Tungsten Cluster host.

B.3.3.1. Network Ports

The following network ports should be open between specific hosts to allow communication between the different components:

Component	Source	Destination	Port	Purpose
All Services	All Nodes	All Nodes	ICMP Ping	Checking availability (Default method)
#	#	#	7	Checking availability
Database Service	Database Host	Database Host	2112	THL replication
#	#	#	7800-7805	Manager Remote Method Invocation (RMI)
#	#	#	9997	Manager Remote Method Invocation (RMI)
#	#	#	10000-10001	Replication connection listener port
#	#	#	11999-12000	Tungsten manager
Connector Service	Connector Host	Manager Hosts	11999	Tungsten manager
Connector Service	#	#	13306	Database connectivity
Client Application	Client	Connector	3306	Database connectivity for client

For composite clusters, communication between each cluster within the composite configuration can be limited to the following ports:

Component	Port	Purpose
Database service	9997	Manager Remote Method Invocation (RMI)
#	2112	THL replication
#	11999-12000	Tungsten Manager
Client Application	13306	MySQL port for Connectivity
Manager Hosts	ICMP Ping	Checking availability (default method)
#	7	Checking availability

For Composite Active/Active and Multi-Site/Active-Active clusters that communicate through replication, the communication between sites can be limited to the following ports:

Component	Port	Purpose
#	2113-211x	THL replication (See Note Below)
#	10002-10003	Replication connection listener ports
Client Application	13306	MySQL port for Connectivity
Manager Hosts	ICMP Ping	Checking availability (default method)
#	7	Checking availability

Note

Within Composite Active/Active, the THL ports configured for the cross-site replication is automatically configured to start at port 2113 and increment by 1 for each additional cross-site service. For example in a 2 cluster configuration, port 2112 would be configured for the local replication, and 2113 for the cross-site replication. In a 3 cluster configuration, port 2112 would be configured for the local replication and ports 2113 and 2114 for each of the cross-site replicators, therefore firewall rules should be configured accordingly to factor in the additional THL ports based on the deployment topology.

Within Multi-Site/Active-Active the THL ports will be configured manually by the user within the tungsten.ini configuration and these should be accounted for when configuring the firewall rules accordingly.

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.3.3.2. SSH Configuration

For password-less SSH to work between the different hosts in the deployment, you need to copy both the public and private keys between the hosts. This will allow each host, to communicate directly with each other using the designated login.

To achieve this, on each host:

1. On one host, create the `.ssh/authorized_keys` file.
2. Add the public key from every host to this file.

   ```
   shell> cat .ssh/id_rsa.pub
   ```

   Copy the output and paste as a new line into the `.ssh/authorized_keys` file. Repeat this for every host so that file that contains all of the public keys for all hosts in the entire deployment.
3. Copy this file to the `.ssh` directory on every host
4. 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 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.3.3.3. Host Availability Checks

The manager checks the availability of other hosts, for example to determine whether the host is still up, rather than just an individual service on that host. These checks must be able to be performed by one of the two available methods. Without these checks, it is possible for the availability of hosts to be falsely determined. These checks are performed using one of two protocols:

- ping — the preferred, and default, method using the system ping (ICMP) command.
- Uses the TCP/IP echo protocol on port 7. The port must be available on the source and destination host, not blocked by a system or network firewall.

The configuration of which service to use depends on the setting of the `--mgr-ping-method` [501] option during configuration. If no option is given, tpm will test ping first and then try `TCP/IP Echo` after. An error will be thrown if neither option works for all members of the dataservice.

B.3.4. Directory Locations and Configuration

On each host within the deployment you must pick, and configure, a number of directories to be used by Tungsten Clustering, as follows:

- `/tmp` Directory

  The `/tmp` directory must be accessible and executable, as it is the location where some software will be extracted and executed during installation and setup. The directory must be writable by the `tungsten` user.

  On some systems, the `/tmp` filesystem is mounted as a separate filesystem and explicitly configured to be non-executable (using the `noexec` filesystem option). Check the output from the mount command.

- Installation Directory

  Tungsten Clustering needs to be installed in a specific directory. The recommended solution is to use `/opt/continuent`. This information will be required when you configure the software.

  The directory should be created, and the owner and permissions set for the configured user:

  ```
  shell> sudo mkdir /opt/continuent
  shell> sudo chown -R tungsten: /opt/continuent
  shell> sudo chmod 700 /opt/continuent
  ```

- Software Staging Directory

  The staging directory is where you download and store the installation packages and from where you first install the software, or issue updates/upgrades. The recommended solution is to use `/opt/continuent/software`.

  The directory should be created, and the owner and permissions set for the configured user:

  ```
  shell> sudo mkdir /opt/continuent/software
  shell> sudo chown -R tungsten: /opt/continuent
  ```

- Configuration File Directory

  It is recommended to create a directory for the configuration file (`tungsten.ini`) The recommended solution is to use `/etc/tungsten`.

  The directory should be created, and the owner and permissions set for the configured user:

  ```
  shell> sudo mkdir /etc/tungsten
  shell> sudo chown -R tungsten: /etc/tungsten
  ```

- Home Directory

  The home directory of the `tungsten` user must be writable by that user.

B.3.5. Configure Software

Tungsten Clustering relies on the following software. Each host must use the same version of each tool.

Software	Versions Supported	Notes
perl	-	Check using perl -v
Perl sigtrap module	-	In most cases this should be already available, however some distributions do not ship by default, therefore if required, install using sudo yum install perl-sigtrap or via cpan install sigtrap

Software	Versions Supported	Notes
rsync	-	Check using rsync --help
Ruby	1.8.7, 1.9.3, or 2.0.0 upwards [a]	JRuby is not supported
Ruby OpenSSL Module	-	Checking using ruby -ropenssl -e 'p "works"'
Ruby Gems	-	
Ruby `io-console` module	-	Install using gem install io-console [b]
Ruby `net-ssh` module	-	Install using gem install net-ssh [c]
Ruby `net-scp` module	-	Install using gem install net-scp [d]
GNU tar	-	gtar is required for Solaris due to limitations in the native tar command
Java Runtime Environment	Java SE 8 (or compatible), Java SE 11 (or compatible) is supported in 6.1.2 and higher	Java 9 and 10 have been tested and validated but certification and support will only cover Long Term releases. See note below for more detail.
zip	-	zip is required by tpm diag in 6.1.2 and higher
readlink (GNU coreutils)	-	On most platforms, this should be available. readlink, supplied by GNU coreutils, is required by tpm diag

[a] Ruby 1.9.1 and 1.9.2 are not supported; these releases remove the execute bit during installation.

[b] `io-console` is only needed for SSH activities, and only needed for Ruby v2.0 and greater.

[c] For Ruby 1.8.7 the minimum version of net-ssh is 2.5.2, install using gem install net-ssh -v 2.5.2

For Ruby 2.5.9 and above, ensure you use v6.1.0 of net-ssh, install using gem install --force net-ssh -v 6.1.0

[d] For Ruby 1.8.7 the minimum version of net-scp is 1.0.4, install using gem install net-scp -v 1.0.4

For Ruby 2.5.9 and above, ensure you use v4.0.0 of net-scp, install using gem install --force net-scp -v 4.0.0

These tools must be installed, running, and available to all users on each host.

To check the current version for any installed tool, login as the configured user (e.g. `tungsten`), and execute the command to get the latest version. For example:

- Java

  Run java -version:

```
shell> java -version
openjdk version "1.8.0_102"
OpenJDK Runtime Environment (build 1.8.0_102-b14)
OpenJDK 64-Bit Server VM (build 25.102-b14, mixed mode)
```

> **Important**
>
> See Section B.1.5, "Java Requirements" for more detail on Java requirements and known issues with certain builds.

  On certain environments, a separate tool such as alternatives (RedHat/CentOS) or update-alternatives (Debian/Ubuntu) may need to be used to switch Java versions globally or for individual users. For example, within CentOS:

```
shell> alternatives --display
```

> **Important**
>
> It is recommended to switch off all automated software and operating system update procedures. These can automatically install and restart different services which may be identified as failures by Tungsten Clustering. Software and Operating System updates should be handled by following the appropriate Section 6.16, "Performing Database or OS Maintenance" procedures.

It also recommended to install ntp or a similar time synchronization tool so that each host in the deployment has the same physical time.

B.3.6. sudo Configuration

Tungsten requires that the user you have configured to run the server has sudo credentials so that it can run and install services as `root`.

Within Linux environments you can do this by editing the `/etc/sudoers` file using visudo and adding the following lines:

```
## Allow tungsten to run any command
tungsten ALL=(ALL) NOPASSWD: ALL
```

Warning

The above syntax is applicable to most Linux environments, however double check if your environment uses different syntax!

sudo can also be configured to handle only specific directories or files. For example, when using xtrabackup, or additional tools in the Tungsten toolkit, such as tprovision, additional commands must be added to the permitted list:

```
tungsten ALL=(ALL) NOPASSWD: /sbin/service, /usr/bin/innobackupex, /bin/rm, »
    /bin/mv, /bin/chown, /bin/chmod, /usr/bin/scp, /bin/tar, /usr/bin/which, »
    /etc/init.d/mysql, /usr/bin/test, /usr/bin/systemctl, »
    /opt/continuent/tungsten/tungsten-replicator/scripts/xtrabackup.sh, »
    /opt/continuent/tungsten/tools/tpm, /usr/bin/innobackupex-1.5.1, »
    /bin/cat, /bin/find, /usr/bin/whoami, /bin/sh, /bin/rmdir, /bin/mkdir, »
    /usr/bin/mysql_install_db, /usr/bin/mysqld, /usr/bin/xtrabackup
```

Note

On some versions of sudo, use of sudo is deliberately disabled for ssh sessions. To enable support via ssh, comment out the requirement for `requiretty`:

```
#Defaults    requiretty
```

B.3.7. SELinux Configuration

To determine the surrent state of SELinux enforcement, use the getenforce command. For example:

```
shell> getenforce
Disabled
```

To disable SELinux, use the setenforce command. For example:

```
shell> setenforce 0
```

Should your company policy enforce the use of SELinux, then you will need to configure various SELinux contexts to allow Tungsten to operate.

When SELinux is enabled, systemctl may refuse to start mysqld if the listener port or location on disk have been changed. The solution is to inform SELinux about any changed or additional resources.

Tungsten best practice is to change the default MySQL port from `3306` to `13306` so that requesting clients do not accidentally connect directly to the database without being routed by the Connector.

If using a non-standard port for MySQL and SELinux is enabled, you must also change the port context, for example:

```
shell > semanage port -a -t mysqld_port_t -p tcp 13306
```

Ensure the file contexts are set correctly for SELinux. For example, to allow MySQL data to be stored in a non-standard location (i.e. `/data`):

```
shell > semanage fcontext -a -t etc_runtime_t /data
shell > restorecon -Rv /data/

shell > semanage fcontext -a -t mysqld_db_t "/data(/.*)?"
shell > restorecon -Rv /data/*
```

B.4. MySQL Database Setup

For replication between MySQL hosts, you must configure each MySQL database server to support the required user names and core MySQL configuration.

Important

For MySQL extraction, Tungsten Clustering must have write access to the database so that status and progress information can be recorded correctly.

Note

Native MySQL replication should not be running when you install Tungsten Clustering. The replication service will be completely handled by Tungsten Replicator, and the normal replication, management and monitoring techniques will not provide you with the information you need.

B.4.1. MySQL Version Support

For a full list of MySQL Versions supported, see Table B.2, “MySQL/Tungsten Version Support”

B.4.2. MySQL Configuration

Each MySQL Server should be configured identically within the system. Although binary logging must be enabled on each host, replication should not be configured, since Tungsten Replicator will be handling that process.

The configured `tungsten` user must be able to read the MySQL configuration file (for installation) and the binary logs. Either the `tungsten` user should be a member of the appropriate group (i.e. `mysql`), or the permissions altered accordingly.

> **Important**
>
> Parsing of mysqld_multi configuration files is not currently supported. To use a mysqld_multi installation, copy the relevant portion of the configuration file to a separate file to be used during installation.

> **Important**
>
> Primary Keys
>
> Whilst it is not mandatory, it is strongly advised that all tables have Primary Keys. This will ensure replication consistency and avoid any potential risks of data drift. Without Primary Keys, when using ROW or MIXED based binlogging, the replicator will create a "sudo" primary key based on all columns of the table. This could cause degredation in performance and in some edge cases could cause replication to not apply changes to the rows that were expected.

To setup your MySQL servers, you need to do the following:

- Configure your `my.cnf` settings. The following changes should be made to the `[mysqld]` section of your `my.cnf` file:
  - By default, MySQL is configured only to listen on the localhost address (127.0.0.1). The `bind-address` parameter should be checked to ensure that it is either set to a valid value, or commented to allow listening on all available network interfaces:

    ```
    #bind-address = 127.0.0.1
    ```

  - Specify the server id

    Each server must have a unique server id:

    ```
    server-id = 1
    ```

    The best practice is for all servers to have a unique ID across all nodes.

  - (Optional) Reconfigure the default MySQL TCP/IP port

    Change the listening port to 13306. The Tungsten Connector will listen on the normal port 3306 for MySQL connections and send them to the database using port 13306.

    ```
    port = 13306
    ```

    If you are not using Tungsten Connector, the setting can remain at the default of 3306.

  - Ensure that the maximum number of open files matches the configuration of the database user. This was configured earlier at 65535 files.

    ```
    open_files_limit = 65535
    ```

  - Enable binary logs

    Tungsten Replicator operates by reading the binary logs on each machine, so logging must be enabled:

    ```
    log-bin = mysql-bin
    ```

  - Set the `sync_binlog` parameter to 1 (one).

    > **Note**
    >
    > In MySQL 5.7, the default value is 1.

    The MySQL `sync_binlog` parameter sets the frequency at which the binary log is flushed to disk. A value of zero indicates that the binary log should not be synchronized to disk, which implies that only standard operating system flushing of writes will occur. A value greater than one configures the binary log to be flushed only after `sync_binlog` events have been written. This can introduce a delay into writing information to the binary log, and therefore replication, but also opens the system to potential data loss if the binary log has not been flushed when a fatal system error occurs.

    Setting a value of value 1 (one) will synchronize the binary log on disk after each event has been written.

```
sync_binlog = 1
```

- Increase MySQL protocol packet sizes

  The replicator can apply statements up to the maximum size of a single transaction, so the maximum allowed protocol packet size must be increase to support this:

```
max_allowed_packet = 52m
```

- Configure InnoDB as the default storage engine

  Tungsten Cluster needs to use a transaction safe storage engine to ensure the validity of the database. The InnoDB storage engine also provides automatic recovery in the event of a failure. Using MyISAM can lead to table corruption, and in the event of a switchover or failure, and inconsistent state of the database, making it difficult to recover or restart replication effectively.

  InnoDB should therefore be the default storage engine for all tables, and any existing tables should be converted to InnoDB before deploying Tungsten Cluster.

```
default-storage-engine = InnoDB
```

- Configure InnoDB Settings

  Tungsten Replicator creates tables and must use InnoDB tables to store the status information for replication configuration and application:

  The MySQL option `innodb_flush_log_at_trx_commit` configures how InnoDB writes and confirms writes to disk during a transaction. The available values are:

  - A value of 0 (zero) provides the best performance, but it does so at the potential risk of losing information in the event of a system or hardware failure. For use with Tungsten Cluster™ the value should never be set to 0, otherwise the cluster health may be affected during a failure or failover scenario.

  - A value of 1 (one) provides the best transaction stability by ensuring that all writes to disk are flushed and committed before the transaction is returned as complete. Using this setting implies an increased disk load and so may impact the overall performance.

    When using Tungsten Cluster in an Composite Active/Active, fan-in or data critical cluster, the value of `innodb_flush_log_at_trx_commit` should be set to 1. This not only ensures that the transactional data being stored in the cluster are safely written to disk, this setting also ensures that the metadata written by Tungsten Cluster™ describing the cluster and replication status is also written to disk and therefore available in the event of a failover or recovery situation.

  - A value of 2 (two) ensures that transactions are committed to disk, but data loss may occur if the disk data is not flushed from any OS or hardware-based buffering before a hardware failure, but the disk overhead is much lower and provides higher performance.

    This setting must be used as a minimum for *all* Tungsten Cluster™ installations, and should be the setting for all configurations that do not require `innodb_flush_log_at_trx_commit` set to `1`.

  At a minimum `innodb_flush_log_at_trx_commit` should be set to `2`; a warning will be generated if this value is set to zero:

```
innodb_flush_log_at_trx_commit = 2
```

  MySQL configuration settings can be modified on a running cluster, providing you switch your host to *`MAINTENANCE`* mode before reconfiguring and restarting MySQL Server. See Section 6.16, “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 4.1, “Deploying Parallel Replication”.

- Increase InnoDB log file size

  The default InnoDB Redo Log file size is 48MB. This should be increased to a larger file size for performance and other reasons. Values of 512MB are common.

  To change the file size, read the corresponding information in the MySQL manual for configuring the file size information. Please see both "MySQL Redo Log" and "Optimizing MySQL InnoDB Redo Logging".

- Binary Logging Format

  Tungsten Replicator works with both statement and row-based logging, and therefore also mixed-based logging. The chosen format is entirely up to the systems and preferences, and there are no differences or changes required for Tungsten Replicator to operate. For native MySQL to MySQL Primary/Replica replication, either format will work fine.

  Depending on the exact use case and deployment, different binary log formats imply different requirements and settings. Certain deployment types and environments require different settings:

  - For Composite Active/Active deployments, use row-based logging. This will help to avoid data drift where statements make fractional changes to the data in place of explicit updates.
  - Use row-based logging for heterogeneous deployments. All deployments to Oracle, MongoDB, Vertica and others rely on row-based logging.
  - Use mixed replication if warnings are raised within the MySQL log indicating that statement only is transferring possibly dangerous statements.
  - Use statement or mixed replication for transactions that update many rows; this reduces the size of the binary log and improves the performance when the transaction are applied on the Replica.
  - Use row replication for transactions that have temporary tables. Temporary tables are replicated if statement or mixed based logging is in effect, and use of temporary tables can stop replication as the table is unavailable between transactions. Using row-based logging also prevents these tables entering the binary log, which means they do not clog and delay replication.

  The configuration of the MySQL server can be permanently changed to use an explicit replication by modifying the configuration in the configuration file:

  ```
  binlog-format = row
  ```

  > **Note**
  >
  > In MySQL 5.7, the default format is `ROW`.

  For temporary changes during execution of explicit statements, the binlog format can be changed by executing the following statement:

  ```
  mysql> SET binlog-format = ROW;
  ```

- `innodb_stats_on_metadata=0`

  Although optional, we would highly recommend setting this property as it has been shown to improve performance by preventing statistics updates every time the information_schema is queried.

  You must restart MySQL after any changes have been made.

- Ensure the `tungsten` user can access the MySQL binary logs by either opening up the directory permissions, or adding the `tungsten` user to the group owner for the directory.

B.4.3. MySQL Configuration for Active/Active Deployments

If you are inserting to the same table at the same time at two or more different sites, and using bi-directional or active/active replication, then special care must be taken to avoid primary key conflicts. Either the auto-increment keys on each need to be offset so they do not conflict, or the application needs to be be able to generate unique keys taking multiple sites into account.

> **Important**
>
> The following configuration is *required* if your application is relying upon the MySQL-native auto-increment primary key feature:

Use the `auto-increment-increment` and `auto-increment-offset` variables to affect the way that MySQL generates the next value in an auto-increment field.

For example, edit `my.cnf` on all servers:

```
# for all servers at site 1

auto-increment-increment = 10
auto-increment-offset = 1
```

```
# for all servers at site 2

auto-increment-increment = 10
auto-increment-offset = 2
```

```
# for all servers at site 3

auto-increment-increment = 10
auto-increment-offset = 3
```

> **Important**
>
> Restart MySQL on all servers.

B.4.4. MySQL 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.4.5, "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 (MySQL 5.x) is required so that the user can perform all administrative operations including setting global variables.
  - If using MySQL 8+ then the `CONNECTION_ADMIN` privilege should be used as well as the `SUPER` privilege. In future MySQL releases it is expected that `SUPER` will eventually be removed.
  - `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 deployment you must create and grant individual user/host combinations:

  ```
  mysql> CREATE USER tungsten@'client1' IDENTIFIED BY 'password';
  mysql> GRANT ALL ON *.* TO tungsten@'client1'  WITH GRANT OPTION;
  ```

  > **Note**
  >
  > If you later change the cluster configuration and add more hosts, you will need to update this configuration with each new host in the cluster.

- MySQL Application Login

  Tungsten Connector requires a user that can be used as the application user to connect to the MySQL server. The login will allow connections to the MySQL databases servers to be used in a consistent fashion across different hosts within the cluster. You must configure this user with access to your database, and then use it as the 'application' user in your cluster configuration.

  ```
  mysql> CREATE USER app_user@'%' IDENTIFIED BY 'password!';
  mysql> GRANT ALL ON *.* TO app_user@'%';
  mysql> REVOKE SUPER ON *.* FROM app_user@'%';
  mysql8+> REVOKE CONNECTION_ADMIN ON *.* FROM app_user@'%';
  ```

  If using MySQL 8+ then the use of SUPER is deprecated in favour of the CONNECTION_ADMIN privilege, at the time of publishing, both privileges exist and should be revoked. It is advisable to check the official documentation for your specific release of MySQL to check

which roles are available and in use. If your application user is granted either one of these privileges, it is possible for that user to still make DML changes to a Read-Only replica which could result in data corruption.

Additional application user logins can be configured by using the `user.map` file within your Tungsten Cluster™ configuration.

As noted above, the creation of explicit host-specific user entries may be required.

In situations where only minimal privileges are desired for the required application user, nothing additional is needed beyond the implied usage granted by the create user command:

```
mysql> CREATE USER app_user@'%' IDENTIFIED BY 'password!';
mysql> SHOW GRANTS FOR app_user;
+----------------------------------+
| Grants for app_user@%            |
+----------------------------------+
| GRANT USAGE ON *.* TO 'app_user'@'%' |
+----------------------------------+
```

The Connector requires this user to be able to gather critical configuration information, listed below:

- An internal-only show slave status request needs access to the `trep_commit_seqno` table in the replication tracking schema:

  ```
  tungsten_%DATASERVICE%.trep_commit_seqno
  ```

- Configuration fetch:

  ```
  select @@wait_timeout;
  ```

- Configuration fetch:

  ```
  select @@version;
  ```

- Configuration fetch:

  ```
  SELECT ID FROM INFORMATION_SCHEMA.COLLATIONS WHERE COLLATION_NAME=@@collation_server LIMIT 1;
  ```

- Keepalive mechanism:

  ```
  SELECT 'KEEP_ALIVE';
  ```

- Calling a tungsten command inside the Connector during a command-line client session:

  ```
  tungsten connection status;
  ```

  will execute the following SQL query:

  ```
  SHOW STATUS LIKE 'ssl_cipher';
  ```

- Calling a tungsten command inside the Connector during a command-line client session:

  ```
  tungsten show processlist;
  ```

  will execute the following SQL query:

  ```
  SHOW FULL PROCESSLIST;
  ```

- If you configure the connector to run in Proxy mode, and you issue the `SHOW REPLICA STATUS` command [SHOW SLAVE STATUS in MySQL version up to, and includeing, 8.0], then any user executing this statement will require the select privilege on the tracking schema table `trep_commit_seqno`. The following DDL can be used as an example:

  ```
  GRANT SELECT ON tungsten_<servicename>.trep_commit_seqno TO '<user>'@'<host>';
  ```

  This will need to be executed after installation, following the initial creation of the tracking schema and tables.

B.4.5. 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 managed cloud instance such as Amazon Aurora.

For this situation, Tungsten Clustering can be configured to use an 'unprivileged' user configuration. This configuration does not require the `SUPER` privilege, but instead needs explicit privileges on the schema created by Tungsten Clustering, and on the schemas that it will update when applying events.

The capability can be enabled by using the following two options and behaviors:

- `privileged-master=false` [507]

  When privileged_master is disabled:

  - A Primary replicator will not attempt to suppress binlog writes during operations.
  - A Primary replicator will not issue a `FLUSH LOGS` command when the replicator starts.
  - The current replicator position is not updated within the `trep_commit_seqno` [693] table.

  The `tungsten` user that connects to the database must be configured to work with the MySQL service using the following grants:

```
mysql> GRANT ALL ON tungsten_alpha.* to tungsten@'%' IDENTIFIED BY 'secret';
mysql> GRANT SELECT ON *.* TO tungsten@'%' IDENTIFIED BY 'secret';
mysql> GRANT REPLICATION SLAVE ON *.* TO tungsten@'%' IDENTIFIED BY 'secret';
mysql> REVOKE SUPER ON *.* FROM tungsten@'%';
```

- `privileged-slave=false` [507]

  When privileged_slave is disabled:

  - The current replicator position is not updated within the `trep_commit_seqno` [693] table.
  - A replicator configured as a cluster-slave, will not attempt to suppress binlog writes during relay operations.

```
mysql> GRANT ALL ON tungsten_batch.* to tungsten@'%' IDENTIFIED BY 'secret';
mysql> GRANT SELECT,INSERT,UPDATE ON *.* TO tungsten@'%' IDENTIFIED BY 'secret';
mysql> GRANT REPLICATION SLAVE ON *.* TO tungsten@'%' IDENTIFIED BY 'secret';
mysql> REVOKE SUPER ON *.* FROM tungsten@'%';
```

  Optionally, `INSERT` and `UPDATE` privileges can be explicitly added to the user permissions for the tables/databases that will be updated during replication.

B.5. Prerequisite Checklist

To simplify the process of preparing your hosts, the checklist below is designed to provide a quick summary of the main prerequisites required.

A PDF version of this checklist can also be downloaded here

Host Specific

Prereq	Complete?
Create OS User – Typically called `tungsten`	
Set `ulimit` for OS User	
Configure sudoers	
Disable SELinux	
Compile `/etc/hosts`	
Setup SSH between hosts	
Create directory for installation (Typically, `/opt/continuent`)	
Create directory for software package if using tar bundle (Typically, `/opt/continuent/software`)	
Create directory for configuration file if INI Install (`/etc/tungsten`)	
Check ownership of new directories set to new OS user	
Install Perl (if not already present)	
Install Ruby	
Install Ruby gems : `net-ssh`	
Install Ruby gems : `net-scp`	
Install Ruby gems : `io-console`	
Install Java	
Install rsync	

Network Specific

Prereq	Complete?
Ensure Network ports open	

Database Specific (All Topologies)

Prereq	Complete?
Ensure `server-id` unique amongst all nodes	
Increase Open Files limits	
Ensure `bin-logging` enabled for cluster nodes, or source replicator nodes	
Review `sync_binlog` parameter	
Increase, if required, `max_allowed_packet`	
Review InnoDB settings	
Set `binlog_format` to ROW (Essential for Active/Active or heterogeneous deployments)	
Ensure `auto_increment` offsets adjusted for Active/Active deployments	
Create DB user with FULL privileges and `GRANT OPTION` – typically called `tungsten` (Used by managers and replicators)	

Database Specific for Clustering Deployments

Prereq	Complete?
Change DB port from default (eg 13306)	
Create user with FULL privileges except `SUPER`, for use by connectors – typically called `app_user`	
Ensure additional application DB user accounts have `REPLICATION CLIENT` privilege (Only if connectors configured to use `SMARTSCALE`)	

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 Error/Cause/Solution.
- Additional troubleshooting for general systems and operational issues.

C.1. Contacting Support

The support portal may be accessed at https://continuent.zendesk.com.

Continuent offers paid support contracts for Continuent Tungsten and Tungsten Replicator. If you are interested in purchasing support, contact our sales team at sales@continuent.com.

C.1.1. Support Request Procedure

Please use the following procedure when requesting support so we can provide prompt service. If we are unable to understand the issue due to lack of required information, it will prevent us from providing a timely response.

1. Please provide a clear description of the problem
2. Which environment is having the issue? [Prod, QA, Dev, etc.]
3. What is the impact upon the affected environment?
4. Identify the problem host or hosts and the role [Primary, Replica, etc]
5. Provide the steps you took to see the problem in your environment
6. Upload the resulting zip file from tpm diag, potentially run more than once on different hosts as needed. Alternatively, use the tungsten_send_diag command.
7. Provide steps already taken and commands already run to resolve the issue
8. Have you searched your previous support cases? https://continuent.zendesk.com.
9. Have you checked the Continuent documentation? https://docs.continuent.com
10. Have you checked our general knowledge base? For our Error/Cause/Solution guidance on specific issues and error messages, and how the reason can be identified and resolved, see Section C.3, “Error/Cause/Solution”.

C.1.2. Creating a Support Account

You can create a support account by logging into the support portal at https://continuent.zendesk.com. Please use your work email address so that we can recognize it and provide prompt service. If we are unable to recognize your company name it may delay our ability to provide a response.

Be sure to allow email from `helpdesk@continuent.com` and `notifications-helpdesk@continuent.com`. These addresses will be used for sending messages from Zendesk.

C.1.3. Open a Support Ticket

Login to the support portal and click on 'Submit a Request' at the top of the screen. You can access this page directly at https://continuent.zendesk.com/requests/new.

C.1.4. Open a Support Ticket via Email

Send an email to helpdesk@continuent.com from the email address that you used to create your support account. You can include a description and attachments to help us diagnose the problem.

C.1.5. Getting Updates for all Company Support Tickets

If multiple people in your organization have created support tickets, it is possible to get updates on any support tickets they open. You should see your organization name along the top of the support portal. It will be listed after the Check Your Existing Requests tab.

To see all updates for your organization, click on the organization name and then click the Subscribe link.

If you do not see your organization name listed in the headers, open a support ticket asking us to create the organization and list the people that should be included.

C.1.6. Support Severity Level Definitions

Summary of the support severity levels with initial response targets:

- Urgent: initial response within an hour

  Represents a reproducible emergency condition (i.e. a condition that involves either data loss, data corruption, or lack of data availability) that makes the use or continued use of any one or more functions impossible. The condition requires an immediate solution. Continuent guarantees a maximum one (1) hour initial response time. Continuent will continue to work with Customer until Customer's database is back in production. The full resolution and the full root cause analysis will be provided when available.

- High: initial response within four (4) hours

  Represents a reproducible, non-emergency condition (i.e. a condition that does not involve either data loss, data corruption or lack of database availability) that makes the use or continued use of any one or more functions difficult, and cannot be circumvented or avoided on a temporary basis by Customer. Continuent guarantees a maximum four (4) hours initial response time.

- Normal: initial response within one (1) business day

  Represents a reproducible, limited problem condition that may be circumvented or avoided on a temporary basis by Customer. Continuent guarantees a maximum one (1) business day initial response time.

- Low: no guaranteed initial response interval

  Represents minor problem conditions or documentation errors that are easily avoided or circumvented by Customer. Additional request for new feature suggestions, which are defined as new functionality in existing product, are also classified as low severity level. Continuent does not guarantee any particular initial response time, or a commitment to fix in any particular time frame unless Customer engages Continuent for professional services work to create a fix or a new feature.

C.2. Support Tools

C.2.1. Generating Diagnostic Information

To aid in the diagnosis of issues, a copy of the logs and diagnostic information will help the support team to identify and trace the problem. There are two methods of providing this information:

- Using tpm diag

  The tpm diag command will collect the logs and configuration information from the active installation and generate a Zip file with the diagnostic information for all hosts within it. The command should be executed from the staging directory. Use tpm query staging to determine this directory:

  ```
  shell> tpm query staging
  tungsten@host1:/home/tungsten/tungsten-clustering-7.0.3-141
  shell> cd /home/tungsten/tungsten-clustering-7.0.3-141
  shell> ./tools/tpm diag
  ```

  The process will create a file called `tungsten-diag-2014-03-20-10-21-29.zip`, with the corresponding date and time information replaced. This file should be included in the reported support issue as an attachment.

  For a staging directory installation, tpm diag will collect together all of the information from each of the configured hosts in the cluster. For an INI file based installation, tpm diag will connect to all configured hosts if ssh is available. If a warning that ssh is not available is generated, tpm diag must be run individually on each host in the cluster.

- Manually Collecting Logs

  If tpm diag cannot be used, or fails to return all the information, the information can be collected manually:

  1. Run tpm reverse on all the hosts in the cluster:

     ```
     shell> tpm reverse
     ```

  2. Collect the logs from each host. Logs are available within the `service_logs` directory. This contains symbolic links to the actual log files. The original files can be included within a tar archive by using the `-h` option. For example:

```
shell> cd /opt/continuent
shell> tar zcfh host1-logs.tar.gz ./service_logs
```

The tpm reverse and log archives can then be submitted as attachments with the support query.

C.2.2. Generating Advanced Diagnostic Information

To aid in the diagnosis of difficult issues, below are tools and procedures to assist in the data collection.

> **Warning**
>
> ONLY excute the below commands and procedures when requested by Continuent support staff.

Manager Memory Usage Script

We have provided a script to easily tell us how much memory a given manager is consuming.

Place the script on all of your manager hosts (i.e. into the tungsten OS user home directory).

> **Note**
>
> The script assumes that 'cctrl' is in the path. If not, then change the script to provide a full path for cctrl.

```
shell> su - tungsten
shell> vi tungsten_manager_memory
#!/bin/bash
memval=`echo gc | cctrl | grep used | tail -1 | awk -F: '{print $2}' | tr -d ' |'`
megabytes=`expr $memval / 1000000`
timestamp=`date +"%F %T" | tr '-' '/'`
echo "$timestamp | `hostname` | $megabytes MB"

shell> chmod 750 tungsten_manager_memory
shell> ./tungsten_manager_memory
```

This script is ideally run from cron and the output redirected to time-stamped log files for later correlation with manager issues.

Manager Thread Dump Procedure

This procedure creates a Manager memory thread dump for detailed analysis.

Run this command on manager hosts when requested by Continuent support.

This will append the detailed thread dump information to the log file named `tmsvc.log` in the `/opt/continuent/tungsten/tungsten-manager/log` directory.

```
shell> su - tungsten
shell> manager dump
shell> tungsten_send_diag -f /opt/continuent/tungsten/tungsten-manager/log/tmsvc.log -c {case_number}
```

Manager Heap Dump Procedure

This procedure creates a Manager memory heap dump for detailed analysis.

Run this command on manager hosts when requested by Continuent support.

This will create a file named `{hostname}.hprof` in the directory where you run it.

```
shell> su - tungsten
shell> jmap -dump:format=b,file=`hostname`.hprof `ps aux | grep JANINO | grep -v grep | awk '{print $2}'`
shell> tungsten_send_diag -f `hostname`.hprof -c {case_number}
```

Configuring Connector Debug Logging

This procedure allows the Connector to be configured for debug logging.

Perform this procedure on Connector hosts when requested by Continuent support.

> **Warning**
>
> Enabling Connector debug logging will decrease performance dramatically. Disk writes will increase as will disk space consumption. Do not use in production environments unless instructed to do so by Continuent support. In

any case, run in this mode for as short a period of time as possible - just long enough to gather the needed debug information. After that is done, disable debug logging.

To enable debug logging, edit the Connector configuration file `tungsten-connector/conf/log4j.properties` and turn Connector logging from INFO to DEBUG (or to TRACE for verbose logging):

```
shell> su - tungsten
shell> vi /opt/continuent/tungsten/tungsten-connector/conf/log4j.properties
# Enable debug for the connector only
logger.Connector.name=com.continuent.tungsten.connector
# WAS: logger.Connector.level=INFO
logger.Connector.level=DEBUG
logger.Connector.additivity=false
shell> connector reconfigure
```

To disable debug logging, edit the Connector configuration file `tungsten-connector/conf/log4j.properties` and revert the change from `DEBUG` to `INFO`.

C.2.3. Using tungsten_upgrade_manager

tungsten_upgrade_manager is used to correct a cctrl display bug in the Manager that causes the `useSSL` value shown via `cctrl> ls -l` to be false when it should be true after an upgrade from v6 to v7.

Warning

Only use the tungsten_upgrade_manager command when instructed to do so by Continuent Support!

Table C.1. tungsten_upgrade_manager Options

Option	Description
`--api`	Use REST APIv2 in place of cli tools where possible
`--backup`, `-b`	
`--debug`, `-d`	Impies -v also. Debug mode is VERY chatty, avoid it unless you really need it.
`--force`, `-f`	
`--help`, `-h`	
`--manager`	manager executable name or fullpath if different from default
`--start`, `-s`	
`--tpm`	tpm executable name or fullpath if different from default
`--verbose`, `-v`	Show verbose output
`--yes`, `-y`	

C.3. Error/Cause/Solution

C.3.1. MySQL is incorrectly configured

Last Updated: 2013-11-01

Condition or Error

The configuration of MySQL was wrong; it included `autocommit=0` and the wrong `server-id`

Causes

- Prerequisites were not followed correctly.

Rectifications

- Edit `my.cnf` and clean up. Restart MySQL if possible. Alternatively, set manually:

```
mysql> set GLOBAL autocommit=1;
Query OK, 0 rows affected (0.00 sec)

mysql> set GLOBAL server_id=2;
Query OK, 0 rows affected (0.01 sec)
```

C.3.2. WARNING: An illegal reflective access operation has occurred

Last Updated: 2020-01-21

Condition or Error

The following Warning may be seen in the Manager Logs when running v6.1.2 and above in conjunction with Java 9 and above

```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.drools.core.rule.builder.dialect.asm.ClassGenerator
(file:/opt/continuent/releases/tungsten-clustering-6.1.2-75_pid16553/tungsten-manager/lib/drools-core-6.3.0.Final.jar)
to method java.lang.ClassLoader.defineClass(java.lang.String,byte[],int,int)
WARNING: Please consider reporting this to the maintainers of org.drools.core.rule.builder.dialect.asm.ClassGenerator
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
```

Causes

- This is a known issue due to new encapsulation controls in Java 9+

Rectifications

- This warning does not affect operations of the Cluster and can be safely ignored

C.3.3. element 'mysql_readonly' not found in path

Last Updated: 2015-06-01

Condition or Error

We are getting the following INFO message in the `tmsvc.log` every few seconds [host1 is the Primary], logs attached:

```
INFO | jvm 1 | 2016/03/04 15:09:38 | |host1 |
INFO | jvm 1 | 2016/03/04 15:09:38 | +--------------------------------------------------------------------------+
INFO | jvm 1 | 2016/03/04 15:09:38 | |element 'mysql_readonly' not found in path |
INFO | jvm 1 | 2016/03/04 15:09:38 | |'/podsjqe/sjqedb1/conf/service/' while searching for entry |
INFO | jvm 1 | 2016/03/04 15:09:38 | |'podsjqe/sjqedb1/conf/service/mysql_readonly' |
INFO | jvm 1 | 2016/03/04 15:09:38 | +--------------------------------------------------------------------------+
```

Causes

- This is caused by the manager

Rectifications

- To prevent INFO messages being reported in the `/opt/continuent/tungsten/tungsten-manager/log/tmsvc.log` file:
  1. Put the cluster into *`MAINTENANCE`* mode
  2. Stop all managers
  3. Start all managers starting with the Primary
  4. Put the cluster into *`AUTOMATIC`* mode

C.3.4. Latency is high: master:ONLINE, progress=41331580333, THL latency=78849.733

Last Updated: 2015-06-01

Condition or Error

Latency is high: master:ONLINE, progress=41331580333, THL latency=78849.733

Causes

- There are many possible causes for this error, however, if you see the following within the log on the Primary it may indicate a specific issue:

```
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CommitSeqnoTable.updateLastCommitSeqno(CommitSeqnoTable.java:548)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.CatalogManager.updateCommitSeqnoTable(CatalogManager.java:223)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.THL.updateCommitSeqno(THL.java:593)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.thl.THLStoreApplier.apply(THLStoreApplier.java:163)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.apply(SingleThreadStageTask.java:768)
```

```
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.runTask(SingleThreadStageTask.java:501)
INFO | jvm 1 | 2016/02/09 15:01:54 | at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.run(SingleThreadStageTask.java:176)
INFO | jvm 1 | 2016/02/09 15:01:54 | at java.lang.Thread.run(Thread.java:745)
INFO | jvm 1 | 2016/02/09 15:01:54 |
```

The stack trace shows that the replicator is updating the trep_commit_seqno table which is normally a very fast operation.

The underlying reason may either be:

- It is possible that updates to MySQL are somehow getting delayed, which would slow down the operation of the replicator as it updates each status update.
- Check the block commit size, as low values will increase the number of updates to the table, and if the MySQL server updates are slow, this in turn slows down the operation of the replicator.

Rectifications

- Focus on making sure the IO system and MySQL commits are not being blocked.

  You can try increasing `repl-svc-applier-block-commit-size` [515] so the replicator has less commits to MySQL on the Primary.

  You can continue to track progress through trepctl status -name tasks [390] as you may see the *`appliedLastSeqno`* value updating less often if you increase this by a lot. Beware that increasing this value too much increases possible data loss since it creates less sync points with the Replicas.

C.3.5. Backup/Restore is not bringing my host back to normal

Last Updated: 2013-11-01

Condition or Error

A backup/restore was performed as requested, but the host is still not coming up.

Causes

- When you backup a node, the backup is stored on that physical server. The correct backup file from an active server should be used on the host being restored.

Rectifications

- You can use that backup to restore another server in two ways:
  - If the backup directory is shared between servers using NFS or a clustered file system, the commands will work like you tried.
  - You must copy the backup files between nodes. See Section 6.11.3, "Restoring from Another Replica" for instructions on that.

C.3.6. 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` [511] parameter. For example, by doubling the size, a transaction of 64GB could be handled:

```
property=replicator.extractor.dbms.transaction_frag_size=2000000
```

If you change the fragment size in this way, the service on the extractor must be reset so that the transaction can be reprocessed and the binary log is parsed again. You can reset the service by using the trepctl reset command.

C.3.7. Triggers not firing correctly on Replica

Last Updated: 2013-11-01

Condition or Error

Newly created triggers are not firing when executed

Causes

- If a new user (definer) was used to create the triggers, they may fail to be executed, raising the following warning in the logs:

```
INFO | jvm 1 | 2013/10/16 04:21:33 | WARNING: Could not execute query »
 org.drizzle.jdbc.internal.common.query.DrizzleQuery@60dc4c81: The »
 MySQL server is running with the --read-only option so it cannot »
 execute this statement
INFO | jvm 1 | 2013/10/16 04:21:33 | 2013-10-16 04:21:33,208 ERROR »
 replicator.pipeline.SingleThreadStageTask [q-to-dbms] Event »
 application failed: seqno=524545571 fragno=0 message=java.sql.SQLException: »
 Statement failed on slave but succeeded on master
INFO | jvm 1 | 2013/10/16 04:21:33 | com.continuent.tungsten.replicator.applier.ApplierException: »
 java.sql.SQLException: Statement failed on slave but succeeded on master
```

  This is an indication that the new definer does not have the required `SUPER` privilege and that a trigger is failing to run.

Rectifications

- In order to fix this issue, the new definer should be given the SUPER privilege on each server and then replication should be restarted. The `SUPER` privilege allows the user to run a statement on a Replica server where the read_only flag has been turned on. If necessary, the scope of the privilege can be restricted to an individual schema. The `GRANT` statement should be done on every database server, while the shun and recover should only be done on the Replicas.

```
mysql> grant SUPER on *.* to user;
mysql> flush privileges;
```

  Within cctrl:

```
cctrl> datasource hostname shun;
cctrl> datasource hostname recover;
```

  You should continue to review the `tungsten-replicator/log/trepsvc.log` file to see what log messages are being written there. It appears that replication is still failing and it is probably related to the same issue. If you want us to review logs to interpret the results for you, you can upload the log file here and someone will look at it.

More Information

Section C.4.1, “Triggers”

C.3.8. ERROR backup.BackupTask Backup operation failed: null

Last Updated: 2019-01-11

Condition or Error

A full Xtrabackup backup has failed, and left the datasource's replicator offline.

Causes

- A common cause of this failure is the existance of zero-length `store-*.properties` files in the `backups/{serviceName}/` directory.

Rectifications

- Simply remove any zero-byte `store-*.properties` files from the `backups/{serviceName}/` directory and retry the backup.

C.3.9. cctrl reports MANAGER(state=STOPPED)

Last Updated: 2013-11-01

Condition or Error

cctrl reports the status for the manager as `MANAGER(state=STOPPED)`

Causes

- The manager has stopped running, possibly due to a fault or error state.

Rectifications

- Restart the manager process on this server is not running. You can start it by running:

```
shell> manager start
```

Or:

```
shell> /opt/continuent/tungsten/tungsten-manager/bin/manager start
```

More Information

Section 4.2.3, “Stopping/Starting Tungsten Managers”

C.3.10. Starting replication after performing a restore because of an invalid restart sequence number

Last Updated: 2013-11-01

Condition or Error

Starting replication fails because of an invalid restart sequence number. Checking the sequence number, `trep_commit_seqno` [693] shows an empty or invalid table contents:

```
mysql> select * from tungsten.trep_commit_seqno;
+-------+--------+-----------+-----------+--------------+---------+-----------------+---------------------+
| seqno | fragno | last_frag | source_id | epoch_number | eventid | applied_latency | update_timestamp |
+-------+--------+-----------+-----------+--------------+---------+-----------------+---------------------+
| -1 | NUL L | NULL | NULL | NULL | NULL | -1 | 2013-10-27 23:44:05 |
+-------+--------+-----------+-----------+--------------+---------+-----------------+---------------------+
1 row in set (0.00 sec)
```

Causes

- The restore may have failed to correctly restore the `tungsten` tables.

Rectifications

- Retry the restore process, making sure the replicator is stopped and there are no updates to the table taking place:

1. Ensure no replicator processes are running:

   ```
   shell> replicator stop
   ```

   Ensure the `/opt/continuent/thl` directory is empty:

   ```
   shell> tpm reset-thl
   ```

2. Restore the backup using whatever backup/restore tool you used.
3. Check that `trep_commit_seqno` [693] has a valid restart position in it
4. ```
   shell> replicator start
   ```

C.3.11. WARN [KeepAliveTimerTask] - Error while sending a KEEP_ALIVE query to connection.

Last Updated: 2017-02-15

Condition or Error

Connections to MySQL through the connector report KeepAliveTimerTask errors in the `connector.log` file

Causes

- Possible causes include local scripts that kill stale connections after some fixed period of time, or the wait_timeout was changed without restarting the connector.

Rectifications
   ```

- Check for local scripts that are killing connections

  Restart the Connector

```
shell> connector restart
```

C.3.12. [S1000][unixODBC][MySQL][ODBC 5.3(w) Driver]SSL connection error: unknown error number [ISQL]ERROR: Could not SQLConnect

Last Updated: 2015-06-01

Condition or Error

We have a new server dedicated to Zabbix monitoring. Zabbix uses an ODBC connection for MySQL. When we try to connect to a Tungsten connector from the new server using ODBC we receive an error:

```
[S1000][unixODBC][MySQL][ODBC 5.3(w) Driver]SSL connection error: unknown error number
[ISQL]ERROR: Could not SQLConnect
```

Causes

- The underlying cause is related to an SSL or encryption error, either the certificate is wrong, or the ciphers being used are not supported.

  Examine the `connector.log` on the Tungsten server we are connecting to returns an error with each attempt:

```
INFO | jvm 1 | 2016/05/20 13:07:17 | WARN [MySQLProtocolHandler] - [172.16.0.120:43571] Error during transfer of authentication packet: no cipher suites
```

  Connecting from to the new server using the mysql client may work:

```
[root@zabbix etc]# mysql -uzabbix -pZ@bbix487sql -hnas-db-ct01-a.safemls.net
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 40019
Server version: 5.6.20-68.0-log-tungsten Percona Server (GPL), Release 68.0, Revision 656
```

  Connecting directly MySQL database on port 13306 using the ODBC connection may also work:

```
[root@zabbix etc]# isql -v ct01
+---------------------------------------+
| Connected! |
| |
| sql-statement |
| help [tablename] |
| quit |
| |
+---------------------------------------+
```

Rectifications

- zabbix is trying to connect to the connector with SSL encryption, but the SSL is not operating. The easiest way to bypass this is disable SSL connections for ODBC. Add the following entry in `odbc.ini` (under the section for the host you're testing):

```
useSSL = No
```

C.3.13. MySQL 8.0+, User Roles and Smartscale

Last Updated: 2021-11-29

Condition or Error

When configuring a connector to use SmartScale, you receive an error indicating users do not ave the required repl_client privilege despite the privilege being granted via a Role (New in MySQL 8.0)

Causes

- Known issue occuring in all versions that support MySQL 8.0. This is due to tpm not aware of Roles, and therefore unable to determine that privileges have been grated in this way.

Rectifications

- Providing you are certain the correct privileges have been granted, you can simply instruct tpm to ignore this particulare validate check by adding the following option to your configuration:

```
skip-validation-check=MySQLConnectorPermissionsCheck
```

C.3.14. Connector shows errors with "java.net.SocketException: Broken pipe"

Last Updated: 2013-11-01

Condition or Error

When using DirectReads, the connector reports errors with a broken pipe.

Causes

- The most likely culprit for this error is that the `wait_timeout` and/or `interactive_timeout` is too low. This causes a problem because pooled connections get timeouts and are closed by the MySQL server.

Rectifications

- Change the configuration for your MySQL server (in `my.cnf`) to increase these timeouts.

C.3.15. Services requires a reset

Last Updated: 2025-03-20

Condition or Error

The replicator service needs to be reset, for example if your MySQL service has been reconfigured, or when resetting a data warehouse or batch loading service after a significant change to the configuration.

Causes

- If the replicator stops replicating effectively, or the configuration and/or schema of a source or target in a datawarehouse loading solution has changed significantly. This will reset the service, starting extraction from the current point, and the target/Replica from the new Primary position. It will also reset all the positions for reading and writing.

Rectifications

- To reset a service entirely, without having to perform a re-installation, refer to the steps in Section 6.13, “Resetting a Tungsten Cluster Dataservice”

  To reset a service entirely, without having to perform a re-installation, refer to the steps in Section 9.24.3.18, “trepctl reset Command”

C.3.16. trepctl status hangs

Last Updated: 2013-11-01

Condition or Error

The trepctl status command hangs at the end of the output after a "cannot fork" error.

Causes

- This can be caused by THL corruption on the Replica.

  This can be also be caused by an Out-Of-Memory condition in either the replicator or in the OS itself.

Rectifications

- You can recreate the THL files on the Replica[s] ONLY. This can be achieved by deleting the existing THL files, which will cause the Replica replicator to download all of the THL data from the Primary again:

  ```
  shell> replicator stop
  shell> cd /opt/continuent
  shell> mv thl thl.old;
  shell> mkdir thl
  shell> replicator start
  shell> trepctl status
  ```

  To increase the Replicator memory, add or edit the following tpm configuration option, then run tpm update:

  ```
  repl-java-mem-size=2048
  ```

C.3.17. Replicator reports an Out of Memory error

Last Updated: 2013-11-01

Condition or Error

Replicator reports an Out of Memory error

Causes

- The configured memory sizes within the replicator are too small for the data being replicated and applied.

Rectifications

- Raise the *repl-java-mem-size* parameter, for example to 3072 (specified in megabytes) within `tungsten.ini` and issue tpm update.

C.3.18. Backup agent name not found: xtrabackup-full

Last Updated: 2015-06-01

Condition or Error

A backup was taken with xtrabackup-full from the Primary. Replica appears to not be configured for xtrabackup-full, which results in there being issues with the restore. How can we configure the Replica to use xtrabackup-full for restore?

Causes

- The underlying cause and indication is that the xtrabackup has not been installed properly on the Replica, or not installed at all, at the point when Tungsten Cluster was being installed. The following will be seen in the status output after a failed restore:

```
minimumStoredSeqNo : -1
offlineRequests : NONE
pendingError : Unable to spawn restore request
pendingErrorCode : NONE
pendingErrorEventId : NONE
pendingErrorSeqno : -1
pendingExceptionMessage: Backup agent name not found: xtrabackup-full
```

  Probably the other hosts didn't require this setting specifically because xtrabackup was installed and detected when Tungsten Cluster was installed on them.

Rectifications

- The steps you need are:

  1. Install xtrabackup if not already installed on the Replica in question
  2. Add the line below to `/etc/tungsten/tungsten.ini`:

     ```
     backup-method=xtrabackup-full
     ```

     And then run tpm update on that Replica host to update the configuration.

C.3.19. ERROR 1010 (HY000) at line 5094506: Error dropping database (can't rmdir './mysql-bin/', errno: 17)

Last Updated: 2013-11-01

Condition or Error

Loading a mysqldump into a MySQL server from a backup/restore fails.

Causes

- The problem may be that your MySQL binary logs are in a subdirectory of your MySQL `data` directory, causing MySQL to view them as a schema.

Rectifications

- Possible steps to resolution:

  1. Modify the dump file so it isn't trying to drop a schema named after the bin log directory
  2. Update the mysql configuration so the bin logs aren't in a directory in the data dir. mysql sees all directories in the data dir as a schema

C.3.20. ERROR >> host1 >> can't alloc thread

Last Updated: 2019-03-04

Condition or Error

tpm installation fails with this error

Causes

- Most common occurenece of this error is often attributed to OS permissions.

Rectifications

- - Review Section B.3.4, "Directory Locations and Configuration"
  - Ensure all installation paths are owned by the correct OS user
  - Ensure OS user configured with correct sudo rights

C.3.21. 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 OS 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.3.2, "Creating the User Environment" for guidance on how these values should be changed.

More Information

Section B.3.2, "Creating the User Environment"

C.3.22. Replicator fails to connect after updating password

Last Updated: 2013-11-01

Condition or Error

Tungsten Replicator fails to connect after changing the `tungsten` user password.

Causes

- The most likely cause is that the configuration within `~/.my.cnf` was forcing a connection to the cluster as `tungsten` user, and user change may have only been made on one host and not replicated to the other MySQL servers.

Rectifications

- First, update the credentials in `~/.my.cnf` and ensure you can connect to all the Replicas with the updated credentials.

Also check that tpm has been configured with the right password and that all servers have the right information. Errors such as:

```
ERROR>>host1>>Unable to connect to the MySQL server using »
 tungsten@host1:3306 (WITH PASSWORD) (MySQLLoginCheck)
```

Indicate that the password may not have been replicated properly. Check the following:

1. Check the user configuration information within each MySQL server and compare the values:

   ```
   mysql> select * from mysql.user where user='tungsten';
   ```

2. For any node that is not up to date, update the password manually:

   ```
   shell> mysql -u root -ppassword -P 3306 -h host1
   mysql> UPDATE `mysql`.`user` SET Password=PASSWORD('secret') WHERE User='tungsten';
   mysql> flush privileges;
   ```

3. Update the tpm and Tungsten Cluster configuration:

   ```
   datasource-password=secret
   ```

4. Restart the replicators:

   ```
   shell> replicator restart
   ```

   Then put the replicators offline/online to refresh the configuration:

   ```
   [LOGICAL] /alpha > datasource host1 offline
   DataSource 'host1@alpha' is now OFFLINE
   [LOGICAL] /alpha > datasource host1 online
   Setting server for data source 'host1' to READ-ONLY
   ```

C.3.23. Cluster remains in MAINTENANCE mode after tpm update

Last Updated: 2021-10-14

Condition or Error

Afer issuing ./tools/tpm update --replace-release from a remote staging host, the cluster policy remains in *MAINTENANCE* mode.

Causes

- Known issue occuring from v6.1.5 onwards.

Rectifications

- Following the update, log into cctrl and check the cluster policy mode. If the cluster is still in *MAINTENANCE* and this is NOT expected, issue:

  ```
  cctrl> set policy automatic
  ```

C.3.24. Replication latency very high

Last Updated: 2013-11-01

Condition or Error

The latency of updates on the Replicas is very high

Causes

- First the reason and location of the delay should be identified. It is possible for replication data to have been replicated quickly, but applying the data changes is taking a long time. Using row-based replication may increase the latency due to the increased quantity of data that must be transferred.

Rectifications

- Check the replication format:

  ```
  shell> grep binlog_format /etc/my.cnf
  binlog_format=ROW
  ```

  Slow Replicas can be the cause, but it may require some configuration changes.

C.3.25. ERROR 2013 (HY000) at line 583: Lost connection to MySQL server during query

Last Updated: 2013-11-01

Condition or Error

Client was disconnected during a query with the error number.

Causes

- Usually this means that the MySQL server has closed the connection or the server has restarted. The exact cause will be more difficult to determine.

Rectifications

- We need a bit more information to provide assistance.

  1. Were you connected through the Tungsten Connector?
  2. Did anything else happen on the servers?
  3. If you were connected through the Tungsten Connector, please upload the tungsten-connector/log/connector.log file from the server you were connected to.

C.3.26. pendingExceptionMessage": "Unable to update last commit seqno: Incorrect datetime value: '2025-03-13 02:02:26' for column 'update_timestamp' at row 1

Last Updated: 2015-06-01

Condition or Error

The following error is reported when applying an event:

```
pendingExceptionMessage": "Unable to update last commit seqno: Incorrect datetime value: '2025-03-13 02:02:26' for column 'update_timestamp' at row 1
```

Causes

- The underlying reason for this error is the format and value of the datetime value that is being represented are either incompatible with the current SQL mode within MySQL, or the datetime combination is one that occurs during a DST switch, which may be incompatible with the SQL mode.

Rectifications

- The solution is to update the SQL mode so that explicit changes are ignored when applying the data, rather than using the information defined during the session. To update the settings. Because the problem will be short lived and specific to the data being applied it can be done temporarily:

  - Edit file `/opt/continuent/tungsten/tungsten-replicator/conf/static-endtest.properties`
  - Find this line:

    ```
    replicator.applier.dbms.ignoreSessionVars=autocommit
    ```

  - Change the line to:

    ```
    replicator.applier.dbms.ignoreSessionVars=autocommit|sql_mode
    ```

  - Restart the replicator using:

    ```
    shell> replicator restart
    ```

  - Wait for the replicator to come online, and process the change that originally caused the problem. Once the data has been replicated, revert the settings in the file back to the old value and restart the replicator again.

C.3.27. Lots of entries added to replicator log

Last Updated: 2015-06-01

Condition or Error

The logging level used by Tungsten Cluster creates a lot of entries, including `WARN`, this generates a lot of information that is difficult to find the real errors and problems. How do i change the logging level?

Causes

- By default, Tungsten Cluster reports a lot of information and detail, including `INFO` and other levels of detail that may generate a lot of information. For example:

```
INFO | jvm 1 | 2016/02/18 10:16:46 | 2016-02-18 15:16:46,789 [brm - remote-to-thl-0] WARN filter.ReplicateFilter »
Ignoring query : No schema found for this query from event 4020717251 (SET @current_db_user := NULL...)
INFO | jvm 1 | 2016/02/18 10:16:46 | 2016-02-18 15:16:46,789 [brm - remote-to-thl-0] WARN filter.ReplicateFilter »
Ignoring query : No schema found for this query from event 4020717257 (SET @disabled_trigger := NULL...)
INFO | jvm 1 | 2016/02/18 10:16:46 | 2016-02-18 15:16:46,789 [brm - remote-to-thl-0] WARN filter.ReplicateFilter »
Ignoring query : No schema found for this query from event 4020717257 (SET @cols := ''...)
INFO | jvm 1 | 2016/02/18 10:16:46 | 2016-02-18 15:16:46,789 [brm - remote-to-thl-0] WARN filter.ReplicateFilter »
Ignoring query : No schema found for this query from event 4020717257 (SET @current_user_fk := NULL...)
INFO | jvm 1 | 2016/02/18 10:16:46 | 2016-02-18 15:16:46,789 [brm - remote-to-thl-0] WARN filter.ReplicateFilter »
Ignoring query : No schema found for this query from event 4020717257 (SET @current_db_user := NULL...)
INFO | jvm 1 | 2016/02/18 10:16:46 | 2016-02-18 15:16:46,789 [brm - remote-to-thl-0] WARN filter.ReplicateFilter »
Ignoring query : No schema found for this query from event 4020717259 (SET @disabled_trigger := NULL...)
INFO | jvm 1 | 2016/02/18 10:16:46 | 2016-02-18 15:16:46,789 [brm - remote-to-thl-0] WARN filter.ReplicateFilter »
Ignoring query : No schema found for this query from event 4020717259 (SET @cols := ''...)
INFO | jvm 1 | 2016/02/18 10:16:46 | 2016-02-18 15:16:46,789 [brm - remote-to-thl-0] WARN filter.ReplicateFilter »
Ignoring query : No schema found for this query from event 4020717259 (SET @current_user_fk := NULL...)
INFO | jvm 1 | 2016/02/18 10:16:46 | 2016-02-18 15:16:46,789 [brm - remote-to-thl-0] WARN filter.ReplicateFilter »
Ignoring query : No schema found for this query from event 4020717259 (SET @current_db_user := NULL...)
```

Rectifications

- The logging level used to report status and other information, and that is written into the log, can be changed to reduce or lower the reporting level. To do this:

  1. Edit the `~tungsten_home/tungsten/tungsten-replicator/conf/log4j.properties`
  2. Find the following line:

     ```
     log4j.logger.com.continuent.tungsten.replicator.filter.ReplicateFilter=DEBUG, stdout
     ```

  This will change the logging level so that only entries at `DEBUG` and higher will be output.

C.3.28. Error: could not settle on encryption_client algorithm

Last Updated: 2015-06-01

Condition or Error

The following error is reported when trying to connect:

```
Error: could not settle on encryption_client algorithm
```

Causes

- Can be due to missing an acceptable cipher on any one of the hosts.

Rectifications

- This is a list of acceptable ciphers:

```
aes128-cbc
3des-cbc
blowfish-cbc
cast128-cbc
aes192-cbc
aes256-cbc
rijndael-cbc@lysator.liu.se
idea-cbc
none
arcfour128
arcfour256
```

These can be configured in `/etc/ssh/sshd_config` under Ciphers.

Try adding a supported cipher {`aes256-cbc`_ to the end of the ciphers in your ssh server config file. Note that SSH and OpenSSL ciphers are mapped, for example like the following:

```
// Maps the SSH name of a cipher to it's corresponding OpenSSL name
SSH_TO_OSSL = {
```

```
 "3des-cbc" => "des-ede3-cbc",
 "blowfish-cbc" => "bf-cbc",
 "aes256-cbc" => "aes-256-cbc",
 "aes192-cbc" => "aes-192-cbc",
 "aes128-cbc" => "aes-128-cbc",
 "idea-cbc" => "idea-cbc",
 "cast128-cbc" => "cast-cbc",
 "arcfour128" => "rc4",
 "arcfour256" => "rc4",
 "arcfour512" => "rc4",
 "none" => "none"
}
```

C.3.29. Unable to update the configuration of an installed directory

Last Updated: 2013-08-07

Condition or Error

Running an update or configuration with tpm returns the error 'Unable to update the configuration of an installed directory'

Causes

- Updates to the configuration of a running cluster must be performed from the staging directory where Tungsten Cluster was originally installed.

Rectifications

- Change to the staging directory and perform the necessary commands with tpm. To determine the staging directory, use:

  ```
  shell> tpm query staging
  ```

  Then change to the staging directory and perform the updates:

  ```
  shell> ./tools/tpm configure ....
  ```

More Information

Chapter 2, *Deployment*

C.3.30. ERROR 1580 (HY000) at line 5093787: You cannot 'DROP' a log table if logging is enabled

Last Updated: 2013-11-01

Condition or Error

Loading a mysqldump into a MySQL server from a backup/restore fails.

Causes

- This appears to be a bug in MySQL that causes mysqldump loads to fail.

Rectifications

- You should be able to import the dump by switching off the slow query log globally before running the import:

  ```
  mysql> SET GLOBAL slow_query_log=0
  ```

C.3.31. The Primary replicator stopped with a JDBC error.

Last Updated: 2015-06-01

Condition or Error

The Primary replicator stopped with a JDBC error.

Causes

- The error log may show a more detailed failure with the JDBC error message:

  ```
  INFO | jvm 1 | 2016/02/08 17:16:24 | 2016-02-08 17:16:24,627 [qktdb - pool-2-thread-1] ERROR management.tungsten.TungstenPlugin Unable to start replicat
  INFO | jvm 1 | 2016/02/08 17:16:24 | java.lang.NumberFormatException: For input string: "0000002417562130"
  ```

```
INFO | jvm 1 | 2016/02/08 17:16:24 | at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
INFO | jvm 1 | 2016/02/08 17:16:24 | at java.lang.Integer.parseInt(Integer.java:495)
INFO | jvm 1 | 2016/02/08 17:16:24 | at java.lang.Integer.valueOf(Integer.java:582)
INFO | jvm 1 | 2016/02/08 17:16:24 | at com.continuent.tungsten.replicator.extractor.mysql.MySQLExtractor.setLastEventId(MySQLExtractor.java:1139)
INFO | jvm 1 | 2016/02/08 17:16:24 | at com.continuent.tungsten.replicator.extractor.ExtractorWrapper.setLastEventId(ExtractorWrapper.java:243)
INFO | jvm 1 | 2016/02/08 17:16:24 | at com.continuent.tungsten.replicator.extractor.ExtractorWrapper.setLastEvent(ExtractorWrapper.java:219)
INFO | jvm 1 | 2016/02/08 17:16:24 | at com.continuent.tungsten.replicator.pipeline.StageTaskGroup.prepare(StageTaskGroup.java:210)
INFO | jvm 1 | 2016/02/08 17:16:24 | at com.continuent.tungsten.replicator.pipeline.Stage.prepare(Stage.java:272)
INFO | jvm 1 | 2016/02/08 17:16:24 | at com.continuent.tungsten.replicator.pipeline.Pipeline.prepare(Pipeline.java:274)
INFO | jvm 1 | 2016/02/08 17:16:24 | at com.continuent.tungsten.replicator.conf.ReplicatorRuntime.prepare(ReplicatorRuntime.java:642)
INFO | jvm 1 | 2016/02/08 17:16:24 | at com.continuent.tungsten.replicator.management.tungsten.TungstenPlugin.online(TungstenPlugin.java:391)
INFO | jvm 1 | 2016/02/08 17:16:24 | at com.continuent.tungsten.replicator.management.OpenReplicatorManager$OfflineToSynchronizingAction.doAction(OpenRe
INFO | jvm 1 | 2016/02/08 17:16:24 | at com.continuent.tungsten.fsm.core.StateMachine.applyEvent(StateMachine.java:220)
INFO | jvm 1 | 2016/02/08 17:16:24 | at com.continuent.tungsten.fsm.event.EventProcessor.run(EventProcessor.java:78)
INFO | jvm 1 | 2016/02/08 17:16:24 | at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:471)
INFO | jvm 1 | 2016/02/08 17:16:24 | at java.util.concurrent.FutureTask.run(FutureTask.java:262)
INFO | jvm 1 | 2016/02/08 17:16:24 | at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
INFO | jvm 1 | 2016/02/08 17:16:24 | at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
INFO | jvm 1 | 2016/02/08 17:16:24 | at java.lang.Thread.run(Thread.java:745)
```

The underlying reason for the error is that MySQL has created a binlog over 2GB and the replicator could not process the event due to the limit of a Java integer.

Rectifications

- The solution for this error, if the log is a rotate event (use mysqlbinlog) is to reposition the replicator using tungsten_set_position.

C.3.32. 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 (`repl-buffer-size` [477]) 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:

  ```
  repl-buffer-size=10
  ```

  Altering the size of each fragment can also help, as it reduces the memory required to hold the data before it is written to disk and sent out over the network to Replica replicators. Reducing the fragment size will reduce the memory footprint. The size is controlled by the `replicator.extractor.dbms.transaction_frag_size` [511] parameter:

  ```
  property=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.

  Finally, if you do need to increase the memory allocated to the repliction, you will need to adjust the `repl-java-mem-size` [496] property

C.4. Known Issues

C.4.1. Triggers

Tungsten Replicator does not automatically shut off triggers on Replicas. This can create problems on Replicas as the trigger will run twice. Typical symptoms are duplicate key errors, though other problems may appear.

There is no simple one-answer-fits-all solution as the behaviour of MySQL and Triggers will differ based on various conditions.

- When using `ROW` Based Binary Logging, MySQL will log all data changes in the binary log, including any data changes performed as a result of a trigger firing
- When using `MIXED` Based Binary Logging...
  - if the Trigger is deemed to be non-deterministic then MySQL will behave based on the `ROW` Based Logging rules and log all data changes, including any data changes performed as a result of a trigger firing.
  - if the Tigger is deemed to be deterministic, then MySQL will behave based on `STATEMENT` Based Logging rules and ONLY log the statement issued by the client and NOT log any changes as a result of the trigger firing

The mixed behaviour outlined above presents challenges for Tungsten Replicator because MySQL does not flag transactions as being the result of a trigger firing or a client application. Therefore, it is not possible for the replicator to make a decision either.

This means, that if you are running with `MIXED` Based Binary Logging enabled, then there may be times when you would want the triggers on the target to fire, and times when you don't. Therefore the recommendations are as follows:

Tungsten Clustering Deployments

- Switch to `ROW` Based Binary Logging, and either
  - Implement the `is_Primary()` function outlined below, or
  - Use the `replicate.ignore` filter to ignore data changes to tables altered by Triggers (ONLY suitable if the filtered tables are solely managed by the Trigger)

Tungsten Replicator Deployments

- If source instance is running in `ROW` Based Binary Logging mode
  - Drop triggers on target. This is practical in fan-in topologies for reporting or other cases where you do not need to failover to the Replica at a later time. Optionally also implement the `dropddl.js` JavaScript filter (Available in Tungsten Replicator v6.1.2 onwards) to prevent CREATE/DROP TRIGGER DDL being replicated, or
  - Implement the `is_Primary()` function outlined below, or
  - Use the `replicate.ignore` filter to ignore data changes to tables altered by Triggers (ONLY suitable if the filtered tables are solely managed by the Trigger)
- If source instance is running in `MIXED` Based Binary Logging mode
  - Use the `replicate.ignore` filter to ignore data changes to tables altered by Triggers (ONLY suitable if the filtered tables are solely managed by the Trigger), or
  - Switch to `ROW` Based Binary Logging and follow recommendations above

The `is_Primary()` approach is simple to implement. First, create a function like the following that returns false if we are using the Tungsten user, as would be the case on a Replica.

```
create function is_Primary()
    returns boolean
    deterministic
    return if(substring_index(user(),'@',1) != 'tungsten',true, false);
```

Next add this to triggers that should not run on the Replica, as shown in the next example. This suppresses trigger action to insert into table bar except on the Primary.

```
delimiter //
create trigger foo_insert after insert on foo
  for each row begin
    if is_Primary() then
      insert into bar set id=NEW.id;
    end if;
  end;
//
```

As long as applications do not use the Tungsten account on the Primary, the preceding approach will be sufficient to suppress trigger operation.

Alternatively, if you are implementing the `is_Primary()` within a clustering deployment, you could check the database read_only parameter. In a clustered deployment, the Replica databases will be in read_only mode and therefore the trigger could be coded to only fire when the database read_only mode is OFF

C.5. Troubleshooting Timeouts

C.6. Troubleshooting Backups

- Operating system command failed

  Backup directory does not exist.

```
...
INFO | jvm 1 | 2013/05/21 09:36:47 | Process timed out: false
INFO | jvm 1 | 2013/05/21 09:36:47 | Process exception null
INFO | jvm 1 | 2013/05/21 09:36:47 | Process stderr: Error: »
    The directory '/opt/continuent/backups/xtrabackup' is not writeable
...
```

- Backup Retention

  The number of backups retained is set in the backup retention field. To set it at installation time, use the `--backup-retention=N` option where N is the number of backups to retain.

  You can check the number of currently retained backups by looking at the replicator.properties file and searching for the following property:

```
replicator.storage.agent.fs.retention=3
```

  The default is 3 backups retained at any given time.

C.7. Running Out of Diskspace

```
...
pendingError           : Event application failed: seqno=156847 »
    fragno=0 message=Unable to store event: seqno=156847
pendingErrorCode       : NONE
pendingErrorEventId    : mysql-bin.000025:0000000024735754;0
pendingErrorSeqno      : 156847
pendingExceptionMessage: Unable to store event: seqno=156847
...
```

The above indicates that the THL information could not be stored on disk. To recover from this error, make space available on the disk, or move the THL files to a different device with more space, then set the replicator service online again.

For more information on moving THL files to a different disk, see Section 6.18.1, “Moving the THL File Location”; for information on moving the backup file location, see Section D.1.1.4, “Relocating Backup Storage”.

C.8. Troubleshooting SSH and tpm

When executing tpm, ssh is used to connect and install the software on other hosts in the cluster. If this fails, and the public key information is correct, there are a number of operations and settings that can be checked. Ensure that you have followed the Section B.3.3.2, “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` [443] option to specify the alternative port.

- Make sure that the user specified in the `--user` [521] to tpm is allowed to connect to your cluster nodes.

C.9. Troubleshooting Data Differences

It can sometimes become necessary to identify table and data differences due to unexpected behaviour or failures. There are a number of third party tools that can help identify and fix however a lot of them assume native replication is in place, the following explains the recommended methods for troubleshooting a Tungsten Environment based on MySQL as the source and target technologies.

C.9.1. Identify Structural Differences

If you suspect that there are differences to a table structure, a simple method to resolve this will be to compare schema DDL.

Extract DDL on the Primary node, specifying the schema in place of {DB}:

```
shell> mysqldump -u root -p --no-data -h localhost --databases {DB} >Primary.sql
```

Repeat the same on the Replica node:

```
shell> mysqldump -u root -p --no-data -h localhost --databases {DB} >Replica.sql
```

Now, using diff, you can compare the results

```
shell> diff Primary.sql Replica.sql
```

Using the output of diff, you can then craft the necessary SQL statements to re-align your structure

C.9.2. Identify Data Differences

It is possible to use pt-table-checksum from the Percona Toolkit to identify data differences, providing you use the syntax described below for bypassing the native replication checks. First of all, it is advisable to familiarise yourself with the product by reading through the providers own documentation here:

https://www.percona.com/doc/percona-toolkit/2.2/pt-table-checksum.html

Once you are ready, ensure you install the latest version to the persona toolkit on all nodes, next execute the following on the Primary node:

```
shell> pt-table-checksum --set-vars innodb_lock_wait_timeout=500 \
--recursion-method=none \
--ignore-databases=mysql \
--ignore-databases-regex=tungsten* \
h=localhost,u=tungsten,p=secret
```

On first run, this will create a database called percona, and within that database a table called checksums. The process will gather checksum information on every table in every database excluding the mysql and tungsten related schemas. You can now execute the following SQL Statement on the Replica to identify tables with data differences:

```
SELECT db, tbl, SUM(this_cnt) AS total_rows, COUNT(*) AS chunks
FROM percona.checksums
WHERE (
 master_cnt <> this_cnt
 OR master_crc <> this_crc
 OR ISNULL(master_crc) <> ISNULL(this_crc))
GROUP BY db, tbl;
```

This `SELECT` will return any tables that it detects are different, it won't show you the differences, or indeed how many, this is just a basic check. To identify and fix the changes, you could use pt-table-sync, however this product would by default assume native replication and also try and fix the problems for you. In a tungsten environment this would not be recommended, however by using the `--print` switch you can gather the SQL needed to be executed to fix the mistakes. You should run this, and review the output to determine whether you want to manually patch the data together or consider using tungsten_provision_slave to retrovision a node in the case of large quantities of differences.

To use pt-table-sync, first identify the tables with differences on each Replica, in this example, the SELECT statement above identified that there was a data difference on the departments table within the employees database on db2. Execute the pt-table-sync script on the Primary, passing in the database name, table name and the Replica host that the difference exists on:

```
shell> pt-table-sync --databases employees --tables departments --print h=db1,u=tungsten,p=secret,P=13306 h=db2
```

The first h= option should be the Primary, also the node you run the script from, the second h= option relates to the Replica that the difference exists on. Executing the script will output SQL statements that can be used to patch the data, for example the above statement produces the following output:

```
UPDATE `employees`.`departments`
SET `dept_name`='Sales'
WHERE `dept_no`='d007'
LIMIT 1
/*percona-toolkit src_db:employees src_tbl:departments src_dsn:P=13306,h=db1,p=...,u=tungsten
dst_db:employees dst_tbl:departments dst_dsn:P=13306,h=db2,p=...,u=tungsten
lock:0 transaction:1 changing_src:0 replicate:0 bidirectional:0 pid:24524 user:tungsten host:db1*/;
```

The UPDATE statements could now be issued directly on the Replica to correct the problem.

Warning

Generally, changing data directly on a Replica is not recommended, but every environment is different. before making any changes like this *always* ensure you have a FULL backup, and it would be recommended to shun the Replica node (if in a clustered environment) before making any changes so as not to cause any potential interruption to connected clients

C.10. Comparing Table Data

The Percona Toolkit includes a tool called pt-table-checksum that enables you to compare databases on different databases using a checksum comparison. This can be executed by running the checksum generation process on the Primary:

```
shell> pt-table-checksum --set-vars innodb_lock_wait_timeout=500 \
    --recursion-method=none \
    --ignore-databases=mysql \
    --ignore-databases-regex=tungsten* \
    h=localhost,u=tungsten,p=secret
```

Using MySQL, the following statement must then be executed to check the checksums generated on the Primary:

```
mysql> <userinput>SELECT db, tbl, SUM(this_cnt) AS total_rows, COUNT(*) AS chunks \
    FROM percona.checksums WHERE ( master_cnt <> this_cnt OR master_crc \
    <> this_crc OR ISNULL(master_crc) <> ISNULL(this_crc)) GROUP BY db, tbl;</userinput>
```

Any differences will be reported and will need to manually corrected.

C.11. Troubleshooting Memory Usage

Appendix D. Files, Directories, and Environment

D.1. The Tungsten Install Directory

Any Tungsten Clustering installation creates an installation directory that contains the software and the additional directories where active information, such as the transaction history log and backup data is stored. A sample of the directory is shown below, and a description of the individual directories is provided in Table D.1, "Continuent Tungsten Directory Structure".

```
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-clustering-7.0.3-141_pid31409
```

The directories shown in the table are relative to the installation directory, the recommended location is `/opt/continuent`. For example, the THL files would be located in `/opt/continuent/thl`.

Table D.1. Continuent Tungsten Directory Structure

Directory	Description
`backups`	Default directory for backup file storage
`conf`	Configuration directory with a copy of the current and past configurations
`relay`	Location for relay logs if relay logs have been enabled.
`releases`	Contains one or more active installations of the Continuent Tungsten software, referenced according to the version number and active process ID.
`service-logs`	Logging information for the active installation
`share`	Active installation information, including the active JAR for the MySQL connection
`thl`	The Transaction History Log files, stored in a directory named after each active service.
`tungsten`	Symbolic link to the currently active release in `releases`.

Some advice for the contents of specific directories within the main installation directory are described in the following sections.

D.1.1. The `backups` Directory

The `backups` directory is the default location for the data and metadata from any backup performed manually or automatically by Tungsten Clustering. 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 Mar  3 16:09 .
drwxrwxr-x 3 tungsten tungsten      4096 Mar  3 11:51 ..
-rw-r--r-- 1 tungsten tungsten        71 Mar  3 16:09 storage.index
-rw-r--r-- 1 tungsten tungsten 133907646 Mar  3 16:09 store-0000000001-mysqldump_2025-03-04_16-08_42.sql.gz
-rw-r--r-- 1 tungsten tungsten       317 Mar  3 16:09 store-0000000001.properties
```

The `storage.index` contains the backup file index information. The actual backup data is stored in the GZipped file. The properties of the backup file, including the tool used to create the backup, and the checksum information, are location in the corresponding `.properties` file. Note that each backup and property file is uniquely numbered so that you can identify and restore a specific backup.

Different backups scripts and methods may place their backup information in a separate subdirectory. For example, xtrabackup stores backup data into `/opt/continuent/backups/xtrabackup`.

For full details of taking Backups, and restoring, see Section 6.10, "Creating a Backup"

D.1.1.1. Automatically Deleting Backup Files

Tungsten will automatically remove old backup files. This is controlled by the `repl-backup-retention` [476] setting and defaults to 3. Use the tpm update command to modify this setting. Following the successful creation of a new backup, the number of backups will be compared to

the retention value. Any excess backups will be removed from the `/opt/continuent/backups` directory or whatever directory is configured for `repl-backup-directory` [475].

The backup retention will only remove files starting with `store`. If you are using a backup method that creates additional information then those files may not be fully removed until the next backup process begins. This includes xtrabackup-full, xtrabackup-incremental and any snapshot based backup methods. You may manually clean these excess files if space is needed before the next backup method. If you delete information associated with an existing backup, any attempts to restore it will fail.

D.1.1.2. Manually Deleting Backup Files

If you no longer need one or more backup files, you can delete the files from the filesystem. You must delete both the SQL data, and the corresponding properties file. For example, from the following directory:

```
shell> ls -al /opt/continuent/backups
total 764708
drwxrwxr-x 2 tungsten tungsten      4096 Apr 16 13:57 .
drwxrwxr-x 3 tungsten tungsten      4096 Apr 16 13:54 ..
-rw-r--r-- 1 tungsten tungsten        71 Apr 16 13:56 storage.index
-rw-r--r-- 1 tungsten tungsten    517170 Apr 15 18:02 store-0000000004-mysqldump-1332463738918435527.sql
-rw-r--r-- 1 tungsten tungsten       311 Apr 15 18:02 store-0000000004.properties
-rw-r--r-- 1 tungsten tungsten    517170 Apr 15 18:06 store-0000000005-mysqldump-2284057977980000458.sql
-rw-r--r-- 1 tungsten tungsten       310 Apr 15 18:06 store-0000000005.properties
-rw-r--r-- 1 tungsten tungsten 781991444 Apr 16 13:57 store-0000000006-mysqldump-3081853249977885370.sql
-rw-r--r-- 1 tungsten tungsten       314 Apr 16 13:57 store-0000000006.properties
```

To delete the backup files for index 4:

```
shell> rm /opt/continuent/backups/alpha/store-0000000004*
```

See the information in Section D.1.1.3, “Copying Backup Files” about additional files related to a single backup. There may be additional files associated with the backup that you will need to manually remove.

Warning

Removing a backup should only be performed if you know that the backup is safe to be removed and will not be required. If the backup data is required, copy the backup files from the backup directory before deleting the files in the backup directory to make space.

D.1.1.3. Copying Backup Files

The files created during any backup can be copied to another directory or system using any suitable means. Once the backup has been completed, the files will not be modified or updated and are therefore safe to be moved or actively copied to another location without fear of corruption of the backup information.

There are multiple files associated with each backup. The number of files will depend on the backup method that was used. All backups will use at least two files in the `/opt/continuent/backups` directory.

```
shell> cd /opt/continuent/backups
shell> scp store-[0]*6[\.-]* host3:$PWD/
store-0000000001-full_xtrabackup_2024-08-16_15-44_86                  100%   70     0.1KB/s   00:00
store-0000000001.properties                                           100%  314     0.3KB/s   00:00
```

Note

Check the ownership of files if you have trouble transferring files or restoring the backup. They should be owned by the `tungsten` system user to ensure proper operation.

If xtrabackup-full method was used, you must transfer the corresponding directory from `/opt/continuent/backups/xtrabackup`. In this example that would be `/opt/continuent/backups/xtrabackup/full_xtrabackup_2024-08-16_15-44_86`.

```
shell> cd /opt/continuent/backups/xtrabackup
shell> rsync -aze ssh full_xtrabackup_2024-08-16_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_2024-10-16_20-55_73
drwxr-xr-x 7 tungsten tungsten 4096 Oct 17 20:55 full_xtrabackup_2024-10-17_20-55_1
drwxr-xr-x 7 tungsten tungsten 4096 Oct 18 20:55 incr_xtrabackup_2024-10-18_20-55_38
drwxr-xr-x 7 tungsten tungsten 4096 Oct 19 20:57 incr_xtrabackup_2024-10-19_20-57_76
drwxr-xr-x 7 tungsten tungsten 4096 Oct 20 20:58 full_xtrabackup_2024-10-20_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_2024-10-21_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_2024-10-21_20-58_97`, it must be copied along with `full_xtrabackup_2024-10-20_20-57_41`.
- To restore `incr_xtrabackup_2024-10-19_20-57_76`, it must be copied along with `incr_xtrabackup_2024-10-18_20-55_38` and `full_xtraback-up_2024-10-17_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` [475] 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 tools/tpm install or tools/tpm update --replace-release), 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-clustering-6.1.25-6__pid16184/
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 tungsten-clustering-7.0.2-161__pid14577/
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 tungsten-clustering-7.1.4-10_pid23747/
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 tungsten-clustering-8.0.0-34_pid24978/
```

The latest release currently in use can be determined by checking the symbolic link, `tungsten` within the installation directory. For example:

```
shell> ll /opt/continuent
total 40
drwxr-xr-x 9 tungsten mysql 4096 May 23 16:19 ./
drwxr-xr-x 3 root     root  4096 Apr 29 16:09 ../
drwxr-xr-x 2 tungsten mysql 4096 May 30 13:27 backups/
drwxr-xr-x 2 tungsten mysql 4096 May 23 16:19 conf/
drwxr-xr-x 3 tungsten mysql 4096 May 10 19:09 relay/
drwxr-xr-x 5 tungsten mysql 4096 May 23 16:19 releases/
drwxr-xr-x 2 tungsten mysql 4096 May 10 19:09 service_logs/
drwxr-xr-x 2 tungsten mysql 4096 May 23 16:18 share/
drwxr-xr-x 3 tungsten mysql 4096 May 10 19:09 thl/
lrwxrwxrwx 1 tungsten mysql   63 May 23 16:19 tungsten -> /opt/continuent/releases/tungsten-clustering-8.0.0-34_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-clustering-8.0.0-34_pid24978/tungsten-replicator/var/treplicator.pid
/opt/continuent/releases/tungsten-clustering-8.0.0-34_pid24978/tungsten-connector/var/tconnector.pid
/opt/continuent/releases/tungsten-clustering-8.0.0-34_pid24978/tungsten-manager/var/tmanager.pid
```

Directories within the `releases` directory that are no longer being used can be safely removed. The clean_release_directory command can be used to sinmplify this process.

D.1.3. The `service_logs` Directory

The `service_logs` directory contains links to the log files for the currently active release. The directory contains the following links:

- `connector.log` — a link to the Tungsten Connector log.
- `tmsvc.log` — a link to the Tungsten Cluster manager log.

- `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 Clustering. Unlike other directories, the `share` directory is not overwritten or replaced during installation or update using tpm. This means that the directory can be used to hold information, such as filter configurations, without the contents being removed when the installation is updated.

The `share` directory is also used as the default location fo all SSL related certificates and keys when security is enabled

D.1.5. The `thl` Directory

The transaction history log (THL) retains a copy of the SQL statements from each Primary host, and it is the information within the THL that is transferred between hosts and applied to the database. The THL information is written to disk and stored in the `thl` directory:

```
shell> ls -al /opt/continuent/thl/alpha/
total 2291984
drwxrwxr-x 2 tungsten tungsten      4096 Apr 16 13:44 .
drwxrwxr-x 3 tungsten tungsten      4096 Apr 15 15:53 ..
-rw-r--r-- 1 tungsten tungsten         0 Apr 15 15:53 disklog.lck
-rw-r--r-- 1 tungsten tungsten 100137585 Apr 15 18:13 thl.data.0000000001
-rw-r--r-- 1 tungsten tungsten 100134069 Apr 15 18:18 thl.data.0000000002
-rw-r--r-- 1 tungsten tungsten 100859685 Apr 15 18:26 thl.data.0000000003
-rw-r--r-- 1 tungsten tungsten 100515215 Apr 15 18:28 thl.data.0000000004
-rw-r--r-- 1 tungsten tungsten 100180770 Apr 15 18:31 thl.data.0000000005
-rw-r--r-- 1 tungsten tungsten 100453094 Apr 15 18:34 thl.data.0000000006
-rw-r--r-- 1 tungsten tungsten 100379260 Apr 15 18:35 thl.data.0000000007
-rw-r--r-- 1 tungsten tungsten 100294561 Apr 16 12:21 thl.data.0000000008
-rw-r--r-- 1 tungsten tungsten 100133258 Apr 16 12:24 thl.data.0000000009
-rw-r--r-- 1 tungsten tungsten 100293278 Apr 16 12:32 thl.data.0000000010
-rw-r--r-- 1 tungsten tungsten 100819317 Apr 16 12:34 thl.data.0000000011
-rw-r--r-- 1 tungsten tungsten 100250972 Apr 16 12:35 thl.data.0000000012
-rw-r--r-- 1 tungsten tungsten 100337285 Apr 16 12:37 thl.data.0000000013
-rw-r--r-- 1 tungsten tungsten 100535387 Apr 16 12:38 thl.data.0000000014
-rw-r--r-- 1 tungsten tungsten 100378358 Apr 16 12:40 thl.data.0000000015
-rw-r--r-- 1 tungsten tungsten 100198421 Apr 16 13:32 thl.data.0000000016
-rw-r--r-- 1 tungsten tungsten 100136955 Apr 16 13:34 thl.data.0000000017
-rw-r--r-- 1 tungsten tungsten 100490927 Apr 16 13:41 thl.data.0000000018
-rw-r--r-- 1 tungsten tungsten 100684346 Apr 16 13:41 thl.data.0000000019
-rw-r--r-- 1 tungsten tungsten 100225119 Apr 16 13:42 thl.data.0000000020
-rw-r--r-- 1 tungsten tungsten 100390819 Apr 16 13:43 thl.data.0000000021
-rw-r--r-- 1 tungsten tungsten 100418115 Apr 16 13:43 thl.data.0000000022
-rw-r--r-- 1 tungsten tungsten 100388812 Apr 16 13:44 thl.data.0000000023
-rw-r--r-- 1 tungsten tungsten  38275509 Apr 16 13:47 thl.data.0000000024
```

THL files are created on both the Primary and Replicas within the cluster. THL data can be examined using the thl command.

The THL is written into individual files, which are by default, no more than 1 GByte in size each. From the listing above, you can see that each file has a unique file index number. A new file is created when the file size limit is reached, and given the next THL log file number. To determine the sequence number that is stored within log, use the thl command:

```
shell> thl index
LogIndexEntry thl.data.0000000001(0:106)
LogIndexEntry thl.data.0000000002(107:203)
LogIndexEntry thl.data.0000000003(204:367)
LogIndexEntry thl.data.0000000004(368:464)
LogIndexEntry thl.data.0000000005(465:561)
LogIndexEntry thl.data.0000000006(562:658)
LogIndexEntry thl.data.0000000007(659:755)
LogIndexEntry thl.data.0000000008(756:1251)
LogIndexEntry thl.data.0000000009(1252:1348)
LogIndexEntry thl.data.0000000010(1349:1511)
LogIndexEntry thl.data.0000000011(1512:1609)
LogIndexEntry thl.data.0000000012(1610:1706)
LogIndexEntry thl.data.0000000013(1707:1803)
LogIndexEntry thl.data.0000000014(1804:1900)
LogIndexEntry thl.data.0000000015(1901:1997)
LogIndexEntry thl.data.0000000016(1998:2493)
LogIndexEntry thl.data.0000000017(2494:2590)
LogIndexEntry thl.data.0000000018(2591:2754)
LogIndexEntry thl.data.0000000019(2755:2851)
LogIndexEntry thl.data.0000000020(2852:2948)
LogIndexEntry thl.data.0000000021(2949:3045)
LogIndexEntry thl.data.0000000022(3046:3142)
LogIndexEntry thl.data.0000000023(3143:3239)
LogIndexEntry thl.data.0000000024(3240:3672)
```

The THL files are retained for seven days by default, although this parameter is configurable. Due to the nature and potential size required to store the information for the THL, you should monitor the disk space and usage.

The purge is continuous and is based on the date the log file was written. Each time the replicator finishes the current THL log file, it checks for files that have exceeded the defined retention configuration and spawns a job within the replicator to delete files older than the retention policy. Old files are only removed when the current THL log file rotates.

For working with THL files, the thl should be used. For further reading on the usage of this command and the various actions and tasks that can be performed, see the following sections:

- Viewing the contents of files: Section 9.20.5, “thl list Command”
- Purging files: Section 9.20.6, “thl purge Command”
- Moving files and adjusting Retention: Section 6.18, “Managing THL Files”
- Compressing and Encrypting THL: Section 6.18.4, “THL Encryption and Compression”

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  9 tungsten mysql  4096 May 23 16:18 tungsten-connector
drwxr-xr-x 14 tungsten mysql  4096 May 23 16:18 tungsten-manager
drwxr-xr-x 11 tungsten mysql  4096 May 23 16:18 tungsten-replicator
```

Table D.2. Continuent Tungsten `tungsten` Sub-Directory Structure

Directory	Description
`bristlecone`	Contains the bristlecone load-testing tools.
`cluster-home`	Home directory for the main tools, configuration and libraries of the Tungsten Clustering installation.
`cookbook`	Cookbook installation and testing tools.
`INSTALL`	Text file describing the basic installation process for Tungsten Clustering
`README.LICENSES`	Software license information.
`tools`	Directory containing the tools for installing and configuring Tungsten Clustering.
`tungsten-connector`	Installed directory of the Tungsten Connector installation.
`tungsten-manager`	Installed directory of the Tungsten Manager installation.
`tungsten-replicator`	Installed directory of the Tungsten Replicator installation.

D.1.6.1. The `tungsten-connector` Directory

This directory holds all of the files, libraries, configuration and other information used to support the installation of Tungsten Connector.

D.1.6.1.1. The `tungsten-connector` Directory

This directory holds library files specific to Tungsten Connector. When perform patches or extending functionality specifically for Tungsten Connector, JAR files can be placed into this directory.

D.1.6.2. The `tungsten-manager` Directory

This directory holds all of the files, libraries, configuration and other information used to support the installation of Tungsten Manager.

D.1.6.2.1. The `tungsten-manager/lib` Directory

This directory holds library files specific to Tungsten Manager. When perform patches or extending functionality specifically for Tungsten Manager, JAR files can be placed into this directory.

D.1.6.3. The `tungsten-replicator` Directory

This directory holds all of the files, libraries, configuration and other information used to support the installation of Tungsten Manager.

D.1.6.3.1. The `tungsten-replicator/lib` Directory

This directory holds library files specific to Tungsten Replicator. When perform patches or extending functionality specifically for Tungsten Replicator, for example when adding JDBC libraries for other databases, the JAR files can be placed into this directory.

D.1.6.3.2. The `tungsten-replicator/scripts` Directory

This directory contains scripts used to support Tungsten Replicator operation.

D.2. Log Files

Each component witin a cluster generates it's own log files. These log files are all written their own directory within the installation directory structure. In addition, symbolic links are generated for easier access to light weight logs more suited for general user use.

For example, this is the listing of the default log directory, /opt/continuent/service_logs:

```
connector.log -> /opt/continuent/tungsten/tungsten-connector/log/connector.log
connector-user.log -> /opt/continuent/tungsten/tungsten-connector/log/connector-user.log
manager-user.log -> /opt/continuent/tungsten/tungsten-manager/log/manager-user.log
mysqldump.log -> /opt/continuent/tungsten/tungsten-replicator/log/mysqldump.log
replicator-user.log -> /opt/continuent/tungsten/tungsten-replicator/log/replicator-user.log
tmsvc.log -> /opt/continuent/tungsten/tungsten-manager/log/tmsvc.log
trepsvc.log -> /opt/continuent/tungsten/tungsten-replicator/log/trepsvc.log
xtrabackup.log -> /opt/continuent/tungsten/tungsten-replicator/log/xtrabackup.log
```

As you can see, each log file is a symlink to the user-level log for each of the layers, the main detailed log for each component, along with logs for backups, if they exist.

D.3. Environment Variables

- `$CONTINUENT_PROFILES`

  This environment variable is used by tpm as the location for storing the `deploy.cfg` file that is created by tpm during a tpm configure or tpm install operation. For more information, see Section 10.4, “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 10.4, “tpm Staging Configuration”.

- `$CONTINUENT_ROOT`

  The `$CONTINUENT_ROOT` variable is created by the `env.sh` file that is created when installing Tungsten Cluster. When defined, the variable will contain the installation directory of the corresponding Tungsten Cluster installation.

  On hosts where multiple installations have been created, the variable can be used to point to different installations.

Appendix E. Terminology Reference

Tungsten Cluster involves a number of different terminology that helps define different parts of the product, and specific areas of the output information from different commands. Some of this information is shared across different tools and systems.

This appendix includes a reference to the most common terms and terminology used across Tungsten Cluster.

E.1. Transaction History Log (THL)

The Transaction History Log (THL) stores transactional data from different data servers in a universal format that is then used to exchange and transfer the information between replicator instances. Because the THL is stored and independently managed from the data servers that it reads and writes, the data can be moved, exchanged, and transmuted during processing.

The THL is created by any replicator service acting as a Primary, where the information is read from the database using the native format, such as the MySQL binary log, and 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 Clustering installation. The exact location can configured using `thl-log-directory`. 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` [519] 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` [520] configuration parameter. The files can be manually purged or moved. See Section 9.20.6.1, “Purging THL Log Information on a Replica”.

The THL can be viewed and managed by using the thl command. For more information, see Section 9.20, “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# = 5 / FRAG# = 0 (last frag)
- FILE = thl.data.0000000001
- TIME = 2025-03-19 11:07:32.0
- EPOCH# = 0
- EVENTID = mysql-bin.000002:0000000000002081;1748
- SOURCEID = db1
- METADATA = [mysql_server_id=10;mysql_thread_id=1748;dbms_type=mysql;tz_aware=true;service=alpha;shard=hr]
- 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 = 'NO_ENGINE_SUBSTITUTION,ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,IGNORE_SPACE,NO_ZERO_DATE,NO_ZERO_IN_DATE', »
character_set_client = 33, collation_connection = 33, collation_server = 255]
```

```
- SCHEMA = hr
- SQL(0) = insert into regions values (7, 'Europe')
```

The sample above shows the information for the SQL executed on a MySQL server. The *EVENTID* [680] shows the MySQL binary log from which the statement has been read. The MySQL server has stored the information in the binary log using ROW mode; MIXED or STATEMENT mode stores the actual SQL statement that was executed. thl command output is provided in Table E.1, "THL Event Format".

Table E.1. THL Event Format

Displayed Field	Internal Name	Data type	Size	Description
-	record_length	Integer	4 bytes	Length of the full record information, including this field
-	record_type	Byte	1 byte	Event record type identifier
-	header_length	Unsigned int	4 bytes	Length of the header information
SEQ# [679]	seqno	Unsigned long	8 bytes	Log sequence number, a sequential value given to each log entry
FRAG# [679]	fragno	Unsigned short	2 bytes	Event fragment number. An event can consist of multiple fragments of SQL or row log data
-	last_frag	Byte	1 byte	Indicates whether the fragment is the last fragment in the sequence
EPOCH# [679]	epoch_number	Unsigned long	8 bytes	Event epoch number. Used to identify log sections within the Primary THL
SOURCEID [680]	source_id	UTF-8 String	Variable (null terminated)	Event source ID, the hostname or identity of the dataserver that generated the event
EVENTID [680]	event_id	UTF-8 String	Variable (null terminated)	Event ID; in MySQL, for example, the binlog filename and position that contained the original event
SHARDID [681]	shard_id	UTF-8 String	Variable (null terminated)	Shard ID to which the event belongs
TIME [680]	tstamp	Unsigned long	8 bytes	Time of the commit that triggered the event
FILE	-	String	-	Filename of the THL file containing the event
-	data_length	Unsigned int	4 bytes	Length of the included event data
-	event	Binary	Variable	Serialized Java object containing the SQL or ROW data
METADATA [680]	Part of event	-	-	Metadata about the event
TYPE [680]	Part of event	-	-	Internal storage type of the event
OPTIONS [680]	Part of event	-	-	Options about the event operation
SCHEMA [681]	Part of event	-	-	Schema used in the event
SQL [681]	Part of event	-	-	SQL statement or row data
-	crc_method	Byte	1 byte	Method used to compute the CRC for the event.
-	crc	Unsigned int	4 bytes	CRC of the event record (not including the CRC value)

- *SEQ#* [679] and *FRAG#* [679]

  Individual events within the log are identified by a sequential *SEQUENCE* [679] 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 deployment. It can be used to determine whether the Replicas and Primary are in sync, and can be used to identify individual transactions within the replication stream.

- *EPOCH#* [679]

  The *EPOCH* [679] value is used a check to ensure that the logs on the Replica and the Primary match. The *EPOCH* [679] is stored in the THL, and a new *EPOCH* [679] is generated each time a Primary goes online. The *EPOCH* [679] value is then written and stored in the THL along-

side each individual event. The *EPOCH* [679] acts as an additional check, beyond the sequence number, to validate the information between the Replica and the Primary. The *EPOCH* [679] value is used to prevent the following situations:

- In the event of a failover where there are events stored in the Primary log, but which did not make it to a Replica, the *EPOCH* [679] acts as a check so that when the Primary rejoins as the Replica, the *EPOCH* [679] numbers will not match the Replica and the new Primary. The trapped transactions be identified by examining the THL output.
- When a Replica joins a Primary, the existence of the *EPOCH* [679] prevents the Replica from accepting events that happen to match only the sequence number, but not the corresponding *EPOCH* [679].

Each time a Tungsten Replicator Primary goes online, the *EPOCH* [679] number is incremented. When the Replica connects, it requests the *SEQUENCE* [679] and *EPOCH* [679], and the Primary confirms that the requested *SEQUENCE* [679] has the requested *EPOCH* [679]. If not, the request is rejected and the Replica gets a validation error:

```
pendingExceptionMessage: Client handshake failure: Client response validation failed: »
    Log epoch numbers do not match: client source ID=west-db2 seqno=408129 »
    server epoch number=408128 client epoch number=189069
```

When this error occurs, the THL should be examined and compared between the Primary and Replica to determine if there really is a mismatch between the two databases. For more information, see Section 6.9, “Managing Transaction Failures”.

- *SOURCEID* [680]

  The *SOURCEID* [680] is a string identifying the source of the event stored in the THL. Typically it is the hostname or host identifier.

- *EVENTID* [680]

  The *EVENTID* [680] is a string identifying the source of the event information in the log. Within a MySQL installed, the *EVENTID* [680] 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* [680]

  When the source information is committed to the database, that information is stored into the corresponding binary log. 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* [680] value as stored in the THL is used to compute latency information when reading and applying data on a Replica.

- *METADATA* [680]

  Part of the binary *EVENT* payload stored within the event fragment, the metadata is collected and stored in the fragment based on information generated by the replicator. The information is stored as a series of key/value pairs. Examples of the information stored include:

  - MySQL server ID
  - Source database type
  - Name of the Replicator service that generated the THL
  - Any 'heartbeat' operations sent through the replicator service, including those automatically generated by the service, such as when the Primary goes online
  - The name of the shard to which the event belongs
  - Whether the contained data is safe to be applied through a block commit operation

- *TYPE* [680]

  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* [680]

  Part of the *EVENT* binary payload, the *OPTIONS* [680] 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* [680] block for each *SQL* [681] statement stored in the event.

- *SCHEMA* [681]

  Part of the *EVENT* structure, the *SCHEMA* [681] provides the database or schema name in which the statement or row data was applied.

- *SHARDID* [681]

  When using parallel apply, provides the generated shard ID for the event when it is applied by the parallel applier thread. data.

- *SQL* [681]

  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)
- FILE = thl.data.0000000001
- TIME = 2025-01-27 11:49:45.0
- EPOCH# = 2500
- EVENTID = mysql-bin.000007:0000000625753960;0
- SOURCEID = host1
- METADATA = [mysql_server_id=1687011;dbms_type=mysql;service=firstrep;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- SQL(0) = SET INSERT_ID = 3
- OPTIONS = [##charset = ISO8859_1, autocommit = 1, sql_auto_is_null = 0, »
    foreign_key_checks = 1, unique_checks = 1, sql_mode = '', character_set_client = 8, »
    collation_connection = 8, collation_server = 8]
- SCHEMA = test
- SQL(1) = INSERT INTO user VALUES (null, 'Charles', now()) /* ___SERVICE___ = [firstrep] */
```

  For row based events, the information is further defined by the individual row data, including the action type (`UPDATE`, `INSERT` or `DELETE`), *SCHEMA* [681], *TABLE* [681] and individual ROW data. For each ROW, there may be one or more *COL* [681] (column) and identifying *KEY* [681] event to identify the row on which the action is to be performed.

  The same statement when recorded in *ROW* [681] format:

```
SEQ# = 3582 / FRAG# = 0 (last frag)
- FILE = thl.data.0000000001
- TIME = 2025-03-19 11:07:32.0
- EPOCH# = 2500
- EVENTID = mysql-bin.000007:0000000625753710;0
- SOURCEID = host1
- METADATA = [mysql_server_id=1687011;dbms_type=mysql;service=firstrep;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = test
 - TABLE = user
 - ROW# = 0
  - COL(1: ) = 2
  - COL(2: ) = Charles
  - COL(3: ) = 2025-03-09 11:07:19.0
```

E.2. Generated Field Reference

When using any of the tools within Tungsten Clustering status information is output using a common set of fields that describe different status information. These field names and terms are constant throughout all of the different tools. A description of each of these different fields is provided below.

E.2.1. Terminology: Fields *masterConnectUri*

The URI being used to extract THL information. On a Primary, the information may be empty, or may contain the reference to the underlying extractor source where information is being read.

On a Replica, the URI indicates the host from which THL data is being read:

```
masterConnectUri : thl://host1:2112/
```

In a secure installation where SSL is being used to exchange data, the URI protocol will be `thls`:

```
masterConnectUri : thls://host1:2112/
```

E.2.2. Terminology: Fields *masterListenUri*

The URI on which the replicator is listening for incoming Replica requests. On a Primary, this is the URI used to distribute THL information.

```
masterListenUri : thls://host1:2112/
```

E.2.3. Terminology: Fields *accessFailures*

E.2.4. Terminology: Fields *active*

E.2.5. Terminology: Fields *activeSeqno*

E.2.6. Terminology: Fields *appliedLastEventId*

The event ID from the source database of the last corresponding event from the stage that has been applied to the database.

MySQL

When extracting from MySQL, the output from trepctl shows the MySQL binary log file and the byte position within the log where the transaction was extracted:

```
shell> trepctl status
Processing status command...
NAME VALUE
---- -----
appliedLastEventId : mysql-bin.000064:0000000002757461;0
...
```

E.2.7. Terminology: Fields *appliedLastGoodLatency*

If the replicator is in an offline or error state, the appliedLatency will show a value of -1. This property will reflect the last known applied latency prior to the offline event.

When the service is online, this will reflect the same values as appliedLatency.

The value is written to disk every 10 secs so as not to overload the replicator. This can be adjusted by setting the following property `svc-applier-last-applied-write-interval=20` [515]

E.2.8. Terminology: Fields *appliedLastGoodSeqno*

If the replicator is in an offline or error state, the appliedLastSeqno will show a value of -1. This property will reflect the last known applied seqno prior to the offline event.

When the service is online, this will reflect the same values as appliedLastSeqno.

The value is written to disk every 10 secs so as not to overload the replicator. This can be adjusted by setting the following property `svc-applier-last-applied-write-interval=20` [515]

E.2.9. Terminology: Fields *appliedLastSeqno*

The last sequence number for the transaction from the Tungsten stage that has been applied to the database. This indicates the last actual transaction information written into the Replica database.

```
appliedLastSeqno : 212
```

When using parallel replication, this parameter returns the minimum applied sequence number among all the channels applying data.

E.2.10. Terminology: Fields *appliedLatency*

The *appliedLatency* is the latency between the commit time of the source event and the time the last committed transaction reached the end of the corresponding pipeline within the replicator.

Within a Primary, this indicates the latency between the transaction commit time and when it was written to the THL. In a Replica, it indicates the latency between the commit time on the Primary database and when the transaction has been committed to the destination database. Clocks must be synchronized across hosts for this information to be accurate.

```
appliedLatency : 0.828
```

The latency is measure in seconds. Increasing latency may indicate that the destination database is unable to keep up with the transactions from the Primary.

In replicators that are operating with parallel apply, *appliedLatency* indicates the latency of the trailing channel. Because the parallel apply mechanism does not update all channels simultaneously, the figure shown may trail significantly from the actual latency.

E.2.11. Terminology: Fields *applier.class*

Classname of the current applier engine

E.2.12. Terminology: Fields *applier.name*

Name of the current applier engine

E.2.13. Terminology: Fields *applyTime*

E.2.14. Terminology: Fields *autoRecoveryEnabled*

Indicates whether autorecovery has been enabled by setting the `--auto-recovery-max-attempts` [474]. The field indicates the value as either `true` or `false` accordingly.

E.2.15. 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.2.16. Terminology: Fields *averageBlockSize*

E.2.17. Terminology: Fields *blockCommitRowCount*

E.2.18. Terminology: Fields *cancelled*

E.2.19. Terminology: Fields *channel*

E.2.20. 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.2.21. 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.2.22. Terminology: Fields *commits*

E.2.23. Terminology: Fields *committedMinSeqno*

E.2.24. Terminology: Fields *criticalPartition*

E.2.25. Terminology: Fields *currentBlockSize*

E.2.26. Terminology: Fields *currentEventId*

Event ID of the transaction currently being processed

E.2.27. Terminology: Fields *currentLastEventId*

E.2.28. Terminology: Fields *currentLastFragno*

E.2.29. Terminology: Fields *currentLastSeqno*

E.2.30. Terminology: Fields *currentTimeMillis*

The current time on the host, in milliseconds since the epoch. This information can used to confirm that the time on different hosts is within a suitable limit. Internally, the information is used to record the time when transactions are applied, and may therefore the *appliedLatency* figure.

E.2.31. Terminology: Fields *dataServerHost*

E.2.32. Terminology: Fields *discardCount*

E.2.33. Terminology: Fields *doChecksum*

E.2.34. Terminology: Fields *estimatedOfflineInterval*

E.2.35. Terminology: Fields *eventCount*

E.2.36. Terminology: Fields *extensions*

E.2.37. Terminology: Fields *extractTime*

E.2.38. Terminology: Fields *extractor.class*

E.2.39. Terminology: Fields *extractor.name*

E.2.40. Terminology: Fields *filter.#.class*

E.2.41. Terminology: Fields *filter.#.name*

E.2.42. Terminology: Fields *filterTime*

E.2.43. Terminology: Fields *flushIntervalMillis*

E.2.44. Terminology: Fields *fsyncOnFlush*

E.2.45. Terminology: Fields *headSeqno*

E.2.46. Terminology: Fields *intervalGuard*

E.2.47. Terminology: Fields *lastCommittedBlockSize*

The *lastCommittedBlockSize* contains the size of the last block that was committed as part of the block commit procedure. The value is only displayed on appliers and defines the number of events in the last block. By comparing this value to the configured block commit size, the commit type can be determined.

For more information, see Section 13.1, “Block Commit”.

E.2.48. Terminology: Fields *lastCommittedBlockTime*

The *lastCommittedBlockSize* contains the duration since the last committed block. The value is only displayed on appliers and defines the number of seconds since the last block was committed. By comparing this value to the configured block interval, the commit type can be determined.

For more information, see Section 13.1, “Block Commit”.

E.2.49. Terminology: Fields *latestEpochNumber*

E.2.50. Terminology: Fields *logConnectionTimeout*

E.2.51. Terminology: Fields *logDir*

E.2.52. Terminology: Fields *logFileRetainMillis*

E.2.53. Terminology: Fields *logFileSize*

E.2.54. Terminology: Fields *maxChannel*

E.2.55. Terminology: Fields *maxDelayInterval*

E.2.56. Terminology: Fields *maxOfflineInterval*

E.2.57. Terminology: Fields *maxSize*

E.2.58. 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.2.59. 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.2.60. Terminology: Fields *name*

E.2.61. Terminology: Fields *offlineRequests*

Contains the specifications of one or more future offline events that have been configured for the replicator. Multiple events are separated by a semicolon:

```
shell> trepctl status
...
inimumStoredSeqNo : 0
offlineRequests : Offline at sequence number: 5262;Offline at time: 2025-01-26 00:00:00 EST
pendingError : NONE
```

E.2.62. Terminology: Fields *otherTime*

E.2.63. Terminology: Fields *pendingError*

E.2.64. Terminology: Fields *pendingErrorCode*

E.2.65. Terminology: Fields *pendingErrorEventId*

E.2.66. Terminology: Fields *pendingErrorSeqno*

The sequence number where the current error was identified

E.2.67. Terminology: Fields *pendingExceptionMessage*

The current error message that caused the current replicator offline

E.2.68. Terminology: Fields *pipelineSource*

The source for data for the current pipeline. On a Primary, the pipeline source is the database that the Primary is connected to and extracting data from. Within a Replica, the pipeline source is the Primary replicator that is providing THL data.

E.2.69. Terminology: Fields *processedMinSeqno*

E.2.70. Terminology: Fields *queues*

E.2.71. Terminology: Fields *readOnly*

E.2.72. Terminology: Fields *relativeLatency*

The relativeLatency is the latency between now and timestamp of the last event written into the local THL. This information gives an indication of how fresh the incoming THL information is. On a Primary, it indicates whether the Primary is keeping up with transactions generated on the Primary database. On a Replica, it indicates how up to date the THL read from the Primary is.

A large value can either indicate that the database is not busy, that a large transaction is currently being read from the source database, or from the Primary replicator, or that the replicator has stalled for some reason.

An increasing *relativeLatency* on the Replica may indicate that the replicator may have stalled and stopped applying changes to the dataserver.

E.2.73. Terminology: Fields *resourcePrecedence*

E.2.74. Terminology: Fields *rmiPort*

E.2.75. Terminology: Fields *role*

The current role of the host in the corresponding service specification. Primary roles are *master* and *slave*.

E.2.76. 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.2.77. Terminology: Fields *serializationCount*

E.2.78. Terminology: Fields *serialized*

E.2.79. 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.2.80. 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.2.81. Terminology: Fields *shard_id*

E.2.82. Terminology: Fields *simpleServiceName*

A simplified version of the *serviceName*.

E.2.83. Terminology: Fields *siteName*

E.2.84. Terminology: Fields *sourceId*

E.2.85. Terminology: Fields *stage*

E.2.86. Terminology: Fields *started*

E.2.87. Terminology: Fields *state*

E.2.88. Terminology: Fields *stopRequested*

E.2.89. Terminology: Fields *store.#*

E.2.90. Terminology: Fields *storeClass*

E.2.91. Terminology: Fields *syncInterval*

E.2.92. Terminology: Fields *taskCount*

E.2.93. Terminology: Fields *taskId*

E.2.94. Terminology: Fields *timeInCurrentEvent*

Shows the time that the replicator has been processing the current event. When processing very large transactions this can be used to determine whether the replicator has stalled or is still actively extracting or applying the information.

E.2.95. Terminology: Fields *timeInStateSeconds*

E.2.96. Terminology: Fields *timeoutMillis*

E.2.97. Terminology: Fields *totalAssignments*

E.2.98. Terminology: Fields *transitioningTo*

E.2.99. Terminology: Fields *uptimeSeconds*

E.2.100. Terminology: Fields *version*

Appendix F. Internals

Tungsten Clustering includes a number of different systems and elements to provide the core services and functionality. Some of these are designed only to be customer-configured. Others should be changed only on the advice of Continuent support. This chapter covers a range of different systems hat are designated as internal features and functionality.

This chapter contains information on the following sections of Tungsten Clustering:

- Section F.1, "Extending Backup and Restore Behavior" — details on how the backup scripts operate and how to write custom backup scripts.
- Section F.2, "Character Sets in Database and Tungsten Cluster" — covers how character sets affect replication and command-line tool output.
- Section F.3, "Understanding Replication of Date/Time Values" — covers how timezone differences can impact replications.
- Section F.4, "Tungsten Clustering Schemas" — covers the internal tracking schema.

F.1. Extending Backup and Restore Behavior

The backup and restore system within Tungsten Clustering is handled entirely by the replicator. When a backup is initiated, the replicator on the specified datasource is asked to start the backup process.

The backup and restore system both use a modular mechanism that is used to perform the actual backup or restore operation. This can be configured to use specific backup tools or a custom script.

F.1.1. Backup Behavior

When a backup is requested, the Tungsten Replicator performs a number of separate, discrete, operations designed to perform the backup operation.

The backup operation performs the following steps:

1. Tungsten Replicator identifies the filename where properties about the backup will be stored. The file is used as the primary interface between the underlying backup script and Tungsten Replicator.
2. Tungsten Replicator executes the configured backup/restore script, supplying any configured arguments, and the location of a properties file, which the script updates with the location of the backup file created during the process.
3. If the backup completes successfully, the file generated by the backup process is copied into the configured Tungsten Cluster directory (for example `/opt/continuent/backups`.
4. Tungsten Replicator updates the property information with a CRC value for the backup file and the standard metadata for backups, including the tool used to create the backup.

A log is created of the backup process into a file according to the configured backup configuration. For example, when backing up using mysqldump the log is written to the log directory as `mysqldump.log`. When using a custom script, the log is written to `script.log`.

As standard, Tungsten Replicator supports two primary backup types, mysqldump and xtrabackup. A third option is based on the incremental version of the xtrabackup tool. The use of external backup script enables additional backup tools and methods to be supported.

To create a custom backup script, see Section F.1.3, "Writing a Custom Backup/Restore Script" for a list of requirements and samples.

F.1.2. Restore Behavior

The restore operation operates in a similar manner to the backup operation. The same script is called (but supplied with the `-restore` command-line option).

The restore operation performs the following steps:

1. Tungsten Replicator creates a temporary properties file, which contains the location of the backup file to be restored.
2. Tungsten Replicator executes the configured backup/restore script in restore mode, supplying any configured arguments, and the location of the properties file.
3. The script used during the restore process should read the supplied properties file to determine the location of the backup file.
4. The script performs all the necessary steps to achieve the restore process, including stopping the dataserver, restoring the data, and restarting the dataserver.

5. The replicator will remain in the `OFFLINE` state once the restore process has finished.

F.1.3. Writing a Custom Backup/Restore Script

The synopsis of the custom script is as follows:

`SCRIPT` {`-backup-restore`} -properties `FILE` -options `OPTIONS`

Where:

- `-backup` — indicates that the script should work in the backup mode and create a backup.
- `-restore` — indicates that the scrip should work in the restore mode and restore a previous backup.
- `-properties` — defines the name of the properties file. When called in *backup* mode, the properties file should be updated by the script with the location of the generated backup file. When called in *restore* mode, the file should be examined by the script to determine the backup file that will be used to perform the restore operation.
- `-options` — specifies any unique options to the script.

The custom script must support the following:

- The script must be capable of performing both the backup and the restore operation. Tungsten Replicator selects the operation by providing the `-backup` or `-restore` option to the script on the command-line.
- The script must parse command-line arguments to extract the operation type, properties file and other settings.
- Accept the name of the properties file to be used during the backup process. This is supplied on the command-line using the format:

  ```
  -properties FILENAME
  ```

  The properties file is used by Tungsten Replicator to exchange information about the backup or restore.
- Must parse any additional options supplied on the command-line using the format:

  ```
  -options ARG1=VAL1&ARG2=VAL2
  ```
- Must be responsible for executing whatever steps are required to create a consistent snapshot of the dataserver
- Must place the contents of the database backup into a single file. If the backup process generates multiple files, then the contents should be packaged using tar or zip.

  The script has to determine the files that were generated during the backup process and collect them into a single file as appropriate.
- Must update the supplied properties with the name of the backup file generated, as follows:

  ```
  file=BACKUPFILE
  ```

  If the file has not been updated with the information, or the file cannot be found, then the backup is considered to have failed.

  Once the backup process has completed, the backup file specified in the properties file will be moved to the configured backup location (for example `/opt/continuent/backups`).
- Tungsten Replicator will forward all `STDOUT` and `STDERR` from the script to the log file `script.log` within the log directory. This file is recreated each time a backup is executed.
- Script should have an exit (return) value of 0 for success, and 1 for failure. The script is responsible for handling any errors in the underlying backup tool or script used to perform the backup, but it must then pass the corresponding success or failure condition using the exit code.

A sample Ruby script that creates a simple text file as the backup content, but demonstrates the core operations for the script is shown below:

```
#!/usr/bin/env ruby
require "/opt/continuent/tungsten/cluster-home/lib/ruby/tungsten"
require "/opt/continuent/tungsten/tungsten-replicator/lib/ruby/backup"
class MyCustomBackupScript < TungstenBackupScript
  def backup
    TU.info("Take a backup with arg1 = #{@options[:arg1]} and myarg = #
{@options[:myarg]}")
    storage_file = "/opt/continuent/backups/backup_" +
Time.now.strftime("%Y-%m-%d_%H-%M") + "_" + rand(100).to_s()
    # Take a backup of the server and store the information to
storage_file
    TU.cmd_result("echo 'my backup' > #{storage_file}")
```

```
    # Write the filename to the final storage file
    TU.cmd_result("echo \"file=#{storage_file}\" > #
{@options[:properties]}")
end
  def restore
    storage_file = TU.cmd_result(". #{@options[:properties]}; echo
$file")
    TU.info("Restore a backup from #{storage_file} with arg1 = #
{@options[:arg1]} and myarg = #{@options[:myarg]}")
    # Process the contents of storage_file to restore into the database
server
end
```

An alternative script using Perl is provided below:

```
#!/usr/bin/perl

use strict;
use warnings;
use Getopt::Long;
use IO::File;

my $argstring = join(' ',@ARGV);

my ($backup,$restore,$properties,$options) = (0,0,'','');

my $result = GetOptions("backup" => \$backup,
   "restore" => \$restore,
   "properties=s" => \$properties,
   "options=s" => \$options,
    );

if ($backup)
{
    my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time);
    my $backupfile = sprintf('mcbackup.%04d%02d%02d-%02d%02d%02d-%02d.dump',
        ($year+1900),$mon,$mday,$hour,$min,$sec,$$);

    my $out = IO::File->new($backupfile,'w') or die "Couldn't open the backup file: $backupfile";

# Fake backup data

    print $out "Backup data!\n";

    $out->close();

# Update the properties file
    my $propfile = IO::File->new($properties,'w') or die "Couldn't write to the properties file";
    print $propfile "file=$backupfile\n";
    $propfile->close();
}

if ($restore)
{
    warn "Would be restoring information using $argstring\n";
}

exit 0;
```

F.1.4. Enabling a Custom Backup Script

To enable a custom backup script, the installation must be updated through tpm to use the script backup method. To update the configuration:

1. Create or copy the backup script into a suitable location, for example `/opt/continuent/share`.
2. Copy the script to each of the datasources within your dataservice.
3. Update the configuration using tpm. The `backup-method` [475] should be set to *`script`*, and the directory location set using the `backup-script` [476] option:

   ```
   backup-method=script
   backup-script=/opt/continuent/share/mcbackup.pl
   backup-online=true
   ```

   The `backup-online` [475] 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:

For example, within cctrl:

```
[LOGICAL:EXPERT] /alpha > datasource host2 backup script
```

> **Note**
>
> Note that the name of the backup method is `script`, not the actual name of the script being used.

F.2. Character Sets in Database and Tungsten Cluster

Character sets within the databases and within the configuration for Java and the wrappers for Tungsten Clustering must match to enable the information to be extracted and viewed.

For example, if you are extracting with the UTF-8 character set, the data must be applied to the target database using the same character set. In addition, the Tungsten Replicator should be configured with a corresponding matching character set. For installations where replication is between identical database flavours (for example, MySQL and MySQL) no explicit setting should be made. For heterogeneous deployments, the character set should be set explicitly.

When installing and using Tungsten Clustering, be aware of the following aspects when using character sets:

- When installing Tungsten Clustering, use the `java-file-encoding` [495] 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 providers documentation appropriate for the release you are using.

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 `repl_time_zone` property 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.
- The thl utility prints time-related data using the default GMT time zone. This can be altered using the `-timezone` option.

Best Practices

We recommend the following steps to ensure successful replication of time-related data.

- Standardize all DBMS server and host time zones to UTC. This minimizes time zone inconsistencies between applications and data stores. The recommendation is particularly important when replicating between different DBMS types, such as MySQL to Hadoop.
- Use the default time zone settings for Tungsten Replicator. Do not change the time zones unless specifically recommended by Continuent support.
- If you cannot standardize on UTC at least ensure that time zones are set consistently on all hosts and applications.

Arbitrary time zone settings create a number of corner cases for database management beyond replication. Standardizing on UTC helps minimize them, hence is strongly recommended.

F.4. Tungsten Clustering Schemas

Tungsten Replicator keeps track of replication for a dataservice within a set of tables inside a schema in both source and target databases. This schema is referred to as the tracking schema and should never be updated directly or altered in anyway.

The schema, by default will be called `tungsten_SERVICE` where `SERVICE` relates to the servicename designated during installation.

Within Composite Active/Passive clusters, the schema name will be based on the name of the composite dataservice, for example `tungsten_global`

The following tables are created automatically upon startup of Tungsten Replicator:

- The **`consistency`** [693] table.

  This table is written to as a result of issuing trepctl check and will trigger a consistency check on the source and target databases.

  Structure of the `consistency` [693] table:

```
+------------+-----------+------+-----+---------+-------+
| Field      | Type      | Null | Key | Default | Extra |
+------------+-----------+------+-----+---------+-------+
| db         | char(64)  | NO   | PRI | NULL    |       |
| tbl        | char(64)  | NO   | PRI | NULL    |       |
| id         | int       | NO   | PRI | NULL    |       |
| row_offset | int       | NO   |     | NULL    |       |
| row_limit  | int       | NO   |     | NULL    |       |
| this_crc   | char(40)  | YES  |     | NULL    |       |
| this_cnt   | int       | YES  |     | NULL    |       |
| master_crc | char(40)  | YES  |     | NULL    |       |
| master_cnt | int       | YES  |     | NULL    |       |
| ts         | timestamp | YES  |     | NULL    |       |
| method     | char(32)  | YES  |     | NULL    |       |
+------------+-----------+------+-----+---------+-------+
```

- The **`heartbeat`** [693] table.

  The `heartbeat` [693] table is used to track heartbeats and check replication is operational. Additionally this can be used to check latency by comparing differences between the `source_tstamp` [693] and `target_tstamp` [693]

  This table is written too when trepctl heartbeat is issued from the primary node or when a cluster heartbeat is issued from within cctrl.

  This table should typically only ever have 1 row, it is replaced each time a heartbeat is initiated.

  Structure of the `heartbeat` [693] table:

```
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| id           | bigint      | NO   | PRI | NULL    |       |
| seqno        | bigint      | YES  |     | NULL    |       |
| eventid      | varchar(128)| YES  |     | NULL    |       |
| source_tstamp| timestamp   | YES  |     | NULL    |       |
| target_tstamp| timestamp   | YES  |     | NULL    |       |
| lag_millis   | bigint      | YES  |     | NULL    |       |
| salt         | bigint      | YES  |     | NULL    |       |
| name         | varchar(128)| YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

- The **`trep_commit_seqno`** [693] table.

  The `trep_commit_seqno` [693] table is used by the replicator to track the position of replication.

  On a primary node, there will only ever be one row that is continually updated with the last position of the extractor process for every extracted event. It will record the specific binlog position and the `seqno` [693] associated with it.

  This enables the rpelicator to restart from the correct position should the replicators be stopped for any reason.

  On a replica, this table will typically only have 1 row, unless parallel appluy has been enabled, in which case there will be 1 row for each parallel channel.

  On a replica, the frequency of this table being updated is controlled by the `svc-applier-block-commit-size`. The setting of this option (default 10) controls the number of events that will be committed to the target prior to the update of the `trep_commit_seqno` [693]. There may be conditions where this update happens more frequently, depending on size of transactions and whether forced commits are taking place.

  Structure of the `trep_commit_seqno` [693] table:

```
+-------------------+-------------+------+-----+---------+-------+
| Field             | Type        | Null | Key | Default | Extra |
+-------------------+-------------+------+-----+---------+-------+
| task_id           | int         | NO   | PRI | NULL    |       |
| seqno             | bigint      | YES  |     | NULL    |       |
| fragno            | smallint    | YES  |     | NULL    |       |
| last_frag         | char(1)     | YES  |     | NULL    |       |
| source_id         | varchar(128)| YES  |     | NULL    |       |
| epoch_number      | bigint      | YES  |     | NULL    |       |
```

```
| eventid           | varchar(128) | YES  |     | NULL    |       |
| applied_latency   | int          | YES  |     | NULL    |       |
| update_timestamp  | timestamp    | YES  |     | NULL    |       |
| shard_id          | varchar(128) | YES  |     | NULL    |       |
| extract_timestamp | timestamp    | YES  |     | NULL    |       |
| connection_id     | bigint       | YES  |     | NULL    |       |
+-------------------+--------------+------+-----+---------+-------+
```

- The **trep_shard** [694] table.

  Unless parallel apply is enabled, this table will not be used and will be empty.

  When parallel apply is enabled, this table will contain a list of the channels and any specific custom sharding that has been configured.

  For more information see Section 4.1.7, “Controlling Assignment of Shards to Channels”

  Structure of the trep_shard [694] table:

```
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| shard_id | varchar(128) | NO   | PRI | NULL    |       |
| master   | varchar(128) | YES  |     | NULL    |       |
| critical | tinyint      | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
```

- The **trep_shard_channel** [694] table.

  Unless parallel apply is enabled, this table will not be used and will be empty.

  When parallel apply is enabled, this table will contain a list of the channels and any specific custom sharding that has been configured.

  For more information see Section 4.1.7, “Controlling Assignment of Shards to Channels”

  Structure of the trep_shard_channel [694] table:

```
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| shard_id | varchar(128) | NO   | PRI | NULL    |       |
| channel  | int          | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
```

Appendix G. Frequently Asked Questions (FAQ)

The following sections provide the questions and answers to questions often asked by customers and in forums.

G.1. General Questions

G.1.1. On a Tungsten Replicator Replica, how do I set both the local Replica THL listener port and the upstream Primaries THL listener port?

You need to specify two options: `thl-port` [520] to set the Replica THL listener port and `master-thl-port` [499] to define the upstream Primary THL listener port. Otherwise `thl-port` [520] alone sets BOTH.

G.1.2. How do I update the IP address of one or more hosts in the cluster?

To update the IP address used by one or more hosts in your cluster, you must perform the following steps:

1. If possible, switch the node into SHUNNED mode.
2. Reconfigure the IP address on the machine.
3. Update the hostname lookup, for example, by editing the IP configuration in `/etc/hosts`.
4. Restart the networking to reconfigure the service.
5. On the node that has changed IP address, run the following from the software staging directory:

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

6. On all other nodes within the cluster:
   a. Update the hostname lookup for the new node, for example, by updating the IP configuration in `/etc/hosts`.
   b. Place the cluster into *`MAINTENANCE`* mode.
   c. Execute the following from the software staging directory:

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

G.1.3. How do I update the password for the replication user in the cluster?

If you need to change the password used by Tungsten Cluster to connect to a dataserver and apply changes, the password can be updated first by changing the information within the your dataserver, and then by updating the configuration using tpm update. The new password is not checked until the Tungsten Replicator process is starting. Changing the password and then updating the configuration will keep replication from failing.

1. Within cctrl set the *`MAINTENANCE`* policy mode:

```
cctrl> set policy maintenance
```

2. Within MySQL, update the password for the user, allowing the change to be replicated to the other datasources:

```
mysql> SET PASSWORD FOR tungsten@'%' = PASSWORD('new_pass');
```

3. Update the values for `datasource-password=new_pass` [510] in your `/etc/tungsten/tungsten.ini` and issue tpm update
4. Set the policy mode in cctrl back to *`AUTOMATIC`* :

```
cctrl> set policy automatic
```

G.1.4. One of my hosts is regularly a number of seconds behind my other Replicas?

The most likely culprit for this issue is that the time is different on the machine in question. If you have ntp or a similar network time tool installed on your machine, use it to update the current time across *all* the hosts within your deployment:

```
shell> ntpdate pool.ntp.org
```

Once the command has been executed across all the hosts, trying sending a heartbeat on the Primary to Replicas and checking the latency:

```
shell> trepctl heartbeat
```

G.1.5. Does the replicate filter (i.e. replicate.do and replicate.ignore) address both DML and DDL?

Both filters replicate.do and replicate.ignore will either do or ignore both DML and DDL

DDL is currently ONLY replicated for MySQL to MySQL topologies, or within MySQL Clusters, although it would be advisable not to use ignore/do filters in a clustered environment where data/structural integrity is key.

With replicate.do, all DML and DDL will be replicated ONLY for any database or table listed as part of the do filter.

With replicate.ignore, all DML and DDL will be replicated except for any database or table listed as part of the ignore filter.

G.1.6. How do you change the replicator heap size after installation?

You can change the configuration by adding/adjusting the following property in the `/etc/tungsten/tungsten.ini` and then issuing tpm update

```
java-mem-size=2048
```

G.1.7. What effect do triggers have on replication, either row or statement-based?

The use of triggers can cause many issues with replication, and the effects can differ between different binlog-format settings [Such as ROW vs STATEMENT]

For a full explanation see Section C.4.1, “Triggers”

G.2. Cloud Deployment and Management

G.2.1. Do we support a 3-node cluster spread across three AWS Availability Zones?

This is a normal deployment pattern for working in AWS reduce risk. A single cluster works quite well in this topology.

G.2.2. What are the best settings for the Tungsten connector intelligent proxy?

Standard settings work out of the box. Fine tuning can be done by working with the specific customer application during a Proof-Of-Concept or Production roll-out.

G.2.3. Do you handle bandwidth/traffic management to the DB servers?

This is not something currently supported.