WAN Replication User Guide

Innovation Release

Version 4.3.5

April 2018

TERRACOTTA

This document applies to BigMemory WAN Replication Version 4.3.5 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: BMM-WANUG-435-20180417

Table of Contents

About the WAN Replication Service....5
What Is the WAN Replication Service?....6
Concepts and Architecture....6
Master-Replica Deployment....7
Replication Modes....9

WAN Replication Demo....11
About the Demo....12
Requirements....12
Running the Demo....12
Terminating the Demo....14

Setting Up WAN Replication....15
Requirements....16
General Steps for Setting Up WAN Replication....16
Orchestrator Configuration Parameters....18
Best Practices and Configuration Considerations....19

Administering WAN Replication....23
Working with the Replication Logs....24
Using the Cleanup Utility....26
Synchronizing Replica Caches....26
About Fault Tolerance and Recovery....27
Overview of Fault Recovery....27
Automatic Recovery Scenarios....28
Manual Recovery....30

Bidirectional WAN Replication....31
About Bidirectional WAN Replication....32
Configuring Bidirectional Mode....32
Data Consistency and Conflict Resolution....32

Monitoring and Managing WAN Replication....35
Overview of Monitoring and Managing WAN Replication....36
Connecting via JMX....36
Connecting via the REST APIs....38
The Available Statistics....41

1 About the WAN Replication Service

- What Is the WAN Replication Service? 6
- Concepts and Architecture 6
- Master-Replica Deployment 7
- Replication Modes 9

What Is the WAN Replication Service?

BigMemory WAN Replication reliably replicates data on a per-cache basis across two or more regions connected by wide area networks. This allows geographically remote data centers to maintain eventually consistent views of data. With BigMemory reliability and predictability, WAN replication ensures business continuity and can be used as part of a disaster recovery plan.

BigMemory WAN Replication integrates with your application so that caches marked for WAN replication are automatically synchronized across the WAN link. The service includes:

- Fault tolerance
- Scaling (both at the local cluster and in number of WAN-linked clusters)
- Support for multiple topologies
- Operational simplicity

Concepts and Architecture

BigMemory WAN Replication uses regional Orchestrators to manage data replication.

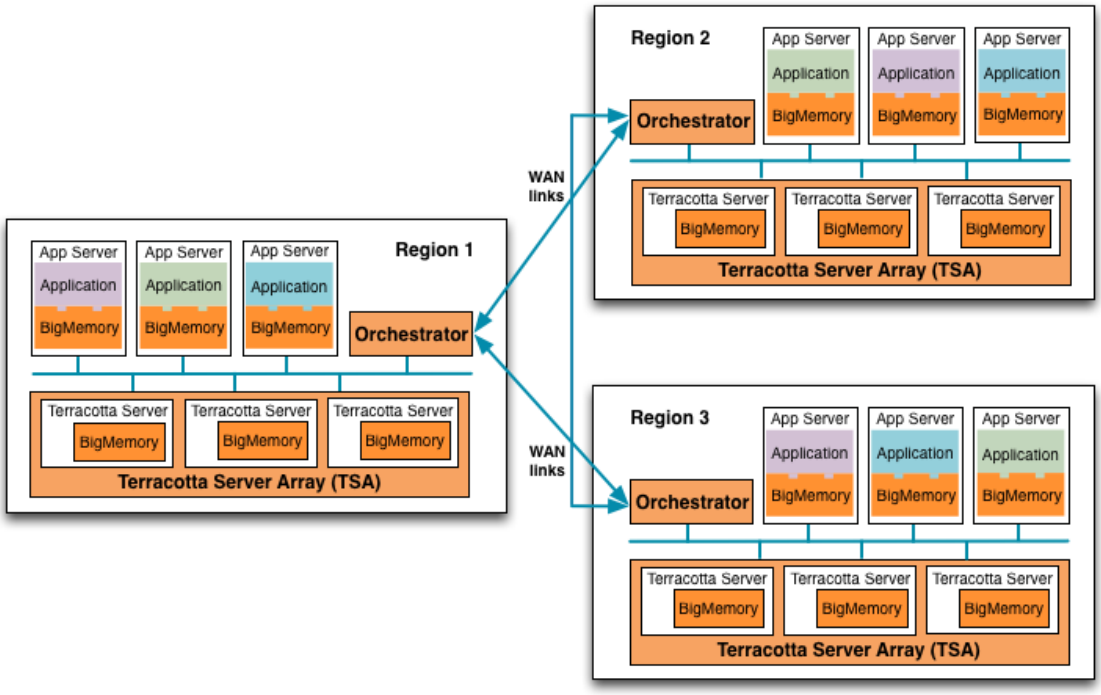
Region - A data center, typically geographically distinct and connected to other regions with a high-speed link. A region holds one or more Terracotta Server Arrays (TSAs), and it can support multiple Orchestrators.

Orchestrator - The process that coordinates communication among regions and replicates cache modifications to all regions. Orchestrators can manage multiple caches.

Region-Orchestrator Example

In the figure below, each application has access to BigMemory data held in its TSA. Each Orchestrator communicates with the TSA in its region, as well as with the Orchestrators in other regions, in order to allow data to become eventually consistent across all regions.

WAN Replication Topology



Master and Replica Caches

The Orchestrators manage replication through designated Master and Replica instances of each cache.

Master Cache - The cache that serves as the authoritative data set against which its Replica caches are synchronized. Each replicated cache will have one active Master at a time.

Replica Cache - A cache mirror that receives replication updates from the Master cache. A Replica cache becomes active only after it fully synchronizes with the Master cache.

Master-Replica Deployment

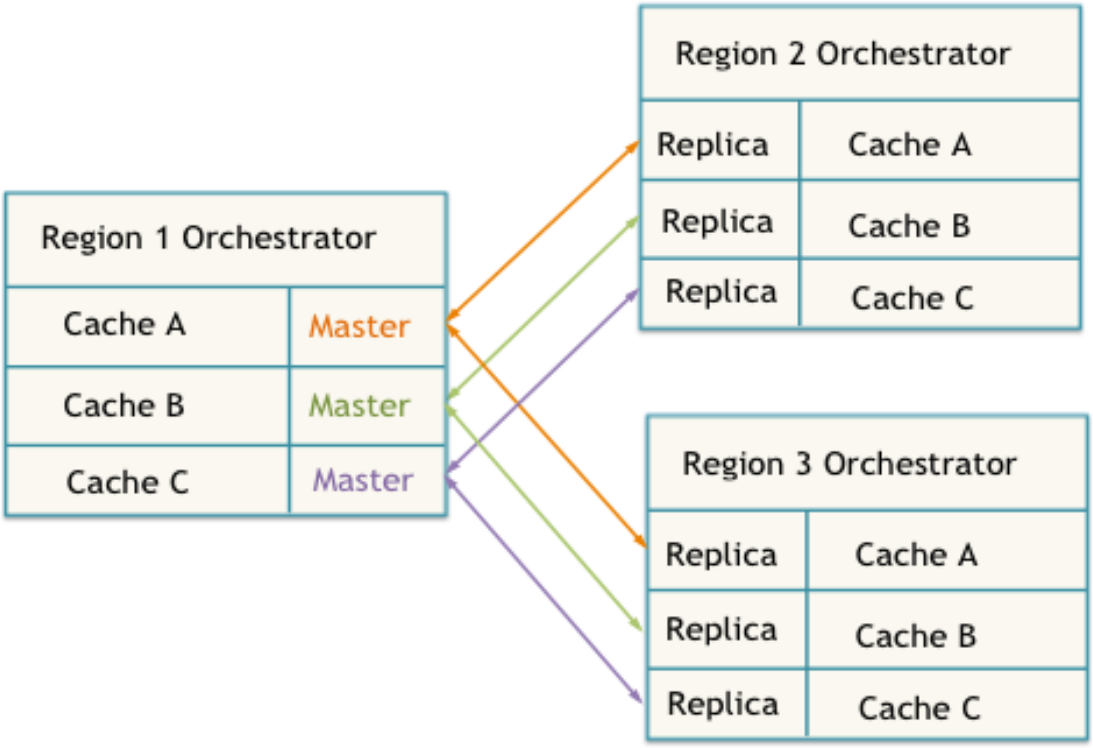
Master and Replica caches can be arranged in two basic deployments: master-region or combination-region.

Master-Region

The master-region deployment may be used to keep all Master caches in a single region. This arrangement is effective when a single application is using WAN replication, or when the Replica region is used as the backup location for disaster recovery.

The figure below shows Caches A, B, and C mastered in Region 1. Regions 2 and 3 are replicas of all caches.

Master-Region Deployment



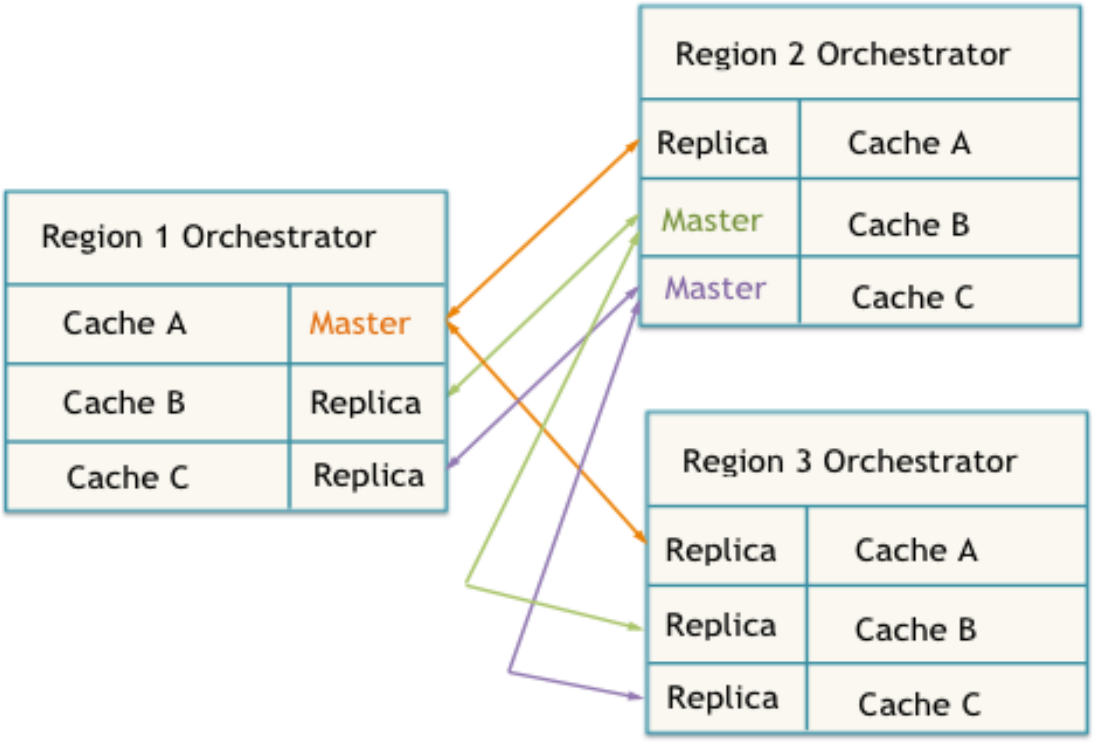
Combination-Region

The combination-region deployment may be used when Master caches are not confined to a particular region. This arrangement is effective for geographically distributed applications that concurrently use BigMemory data, with each Master cache located in the region where its application is most active.

The Orchestrator can manage non-overlapping Master and Replica caches at the same time. This means that one region may be mastering one set of caches while replicating another set of caches.

The figure below shows Cache A mastered in Region 1, and Caches B and C mastered in Region 2. Region 3 is a replica of all caches.

Combination-Region Deployment



Replication Modes

The BigMemory WAN Replication Service offers two replication modes:

- **Unidirectional mode** replicates data in one direction only, from Master to Replica, and is used for active-passive deployments such as disaster recovery.
- **Bidirectional mode** replicates data in two directions, from Master to Replica and from Replica to Master, and is used for active-active deployments where updates from more than one region should be incorporated into the Master cache's authoritative data set.

The table below summarizes the two modes:

Unidirectional Replication	Bidirectional Replication
Active-Passive deployment	Active-Active deployment

Unidirectional Replication	Bidirectional Replication
The Master cache replicates cache element modifications to the Replica caches	Both Master and Replica caches replicate cache element modifications
Replica caches operate locally but do not replicate cache element modifications	The Master cache receives replications from the Replica caches
The Master cache maintains the authoritative data set but no element tracking is necessary	The Master cache tracks each cache element and maintains the authoritative data set
No conflict resolution necessary	Conflict resolution (localized updates recommended)
Example use case: disaster recovery	Example use case: regional operations

The default mode is Unidirectional. To switch to Bidirectional mode, see "Orchestrator Configuration Parameters" on page 18. For more information about Bidirectional mode and conflict resolution, see "Bidirectional WAN Replication" on page 31.

2 WAN Replication Demo

- About the Demo 12
- Requirements 12
- Running the Demo 12
- Terminating the Demo 14

About the Demo

This sample application demonstrates the BigMemory WAN Replication service for BigMemory Max clusters. WAN replication enables two or more BigMemory Max clusters to share data in an eventually consistent manner.

The scenario simulated by this application involves two BigMemory Max clusters, named Region-1 and Region-2. Any data stored in Region-1 will be shown to be replicated to Region-2, and vice versa.

You can run the sample application on a single machine, or on machines that are actually connected via WAN. To run the demo on different machines, you need to specify the correct IP:PORT addresses in the xml configuration files.

The sample application is located under <bigmemory_kit>/code-samples/example09-wan-replication/, and the code used for the application is located under /code-samples/src/.

Requirements

In order to run the sample application, you must have Java 1.6 or higher and you must set JAVA_HOME as follows:

For Windows, use the following command:

```
set JAVA_HOME="<path to java home>"
```

For UNIX/Linux, use the following command:

```
export JAVA_HOME="<path to java home>"
```

If you intend to modify the source code of the application and build it, you will also need Maven.

Running the Demo

Run the application to demonstrate the WAN replication.

Note: The list of all commands is available by typing > help.

1. In the wan-config-region-1.xml and wan-config-region-2.xml files, replace the "wan-samples-root-dir" placeholder with the actual location of the folder (for example, $WAN_SAMPLES/).
2. Launch the Region-1 TSA using the command `start-sample-server-1.sh` (for UNIX/Linux) or `start-sample-server-1.bat` (for Windows).
3. Launch the Region-2 TSA using the command `start-sample-server-2.sh` (for UNIX/Linux) or `start-sample-server-2.bat` (for Windows).

4. Launch the Region-1 WAN Orchestrator using the command `start-orch-1.sh` (for UNIX/Linux) or `start-orch-1.bat` (for Windows).
5. Launch the Region-2 WAN Orchestrator using the command `start-orch-2.sh` (for UNIX/Linux) or `start-orch-2.bat` (for Windows).

   Now that the TSA and WAN services for regions 1 and 2 have started, we are ready to run the application and demonstrate the replication process.
6. Start the Region-1 application using the command `run.sh` (for UNIX/Linux) or `run.bat` (for Windows). You should see a command prompt which indicates that the application has started and is ready to accept various instructions.
7. At the command prompt type:

   ```
   > connect localhost:9510 bar
   ```

   where *bar* is a sample cache name. You can use any name you want. This will start the application's cache manager and will make the Region-1 cluster operational.
8. Start the Region-2 application, but at the command prompt, type:

   ```
   > connect localhost:9610 bar
   ```

   where *bar* is a sample cache name. You can use any name you want. This will start the application's cache manager and will make the Region-2 cluster operational.
9. Store some elements in the Region-1 TSA by typing the following command on the Region-1 application:

   ```
   > fill 100
   ```

   This command will put 100 keys named k0, k1, ... k99 with values v0, v1, ... , v99 into Region-1.
10. See if the elements have been replicated to the Region-2 TSA by typing the following command on the Region-2 application:

    ```
    > get k0
    ```

    You should see [k0, v0] as the result, which indicates that the key k0, which was inserted on Region-1, was successfully replicated to Region-2.
11. Change the k0 value from Region-2 by typing the following command:

    ```
    > put k0 baz
    ```

    This "put", which happened on Region-2, should now be visible in Region-1. Type the following command on the command prompt of Region-1:

    ```
    > get k0
    ```

    You should see `[k0, baz]` as the result, which indicates successful replication.

Terminating the Demo

To terminate and clean up the application, use the scripts `stop-*.sh` (for UNIX/Linux) or `stop-*.bat` (for Windows). Alternatively, kill all the Java processes by sending a CTRL-C signal.

3 Setting Up WAN Replication

- Requirements 16
- General Steps for Setting Up WAN Replication 16
- Orchestrator Configuration Parameters 18
- Best Practices and Configuration Considerations 19

Requirements

The following components are required:

- BigMemory Max 4.1 or higher
- A terracotta-license.key file
- JDK 1.6 or higher
- At least one Terracotta Server Array (TSA) in each region
- Reliable WAN links between all target regions

General Steps for Setting Up WAN Replication

The following steps provide an overview of setting up BigMemory WAN Replication:

1. Ensure that the Terracotta clusters that will use WAN replication can run as expected without WAN replication.

   See the *BigMemory Max Installation Guide* and the *BigMemory Max Administrator Guide* for more information on installing a cluster.

   After verifying the Terracotta clusters, shut down all processes in order to set up for WAN replication.

2. Enable WAN replication for all caches that will be replicated across your WAN.

   For each cache to be replicated, its ehcache.xml configuration file must include the `wanEnabledTSA` attribute set to "true" within the `<terracottaConfig>` element:

   ```
   <terracottaConfig wanEnabledTSA="true"/>
   ```

   Note: Caches that will share data must have the same name.

3. Configure at least one Orchestrator for each region.

   Each Orchestrator requires a dedicated wan-config.xml configuration file. The wan-config.xml specifies the caches that will be Masters and failover Masters by listing the locations of the Master caches. All of the Masters listed for a cache must be in the same region. (Replica caches do not need to be specified in the Orchestrator configuration file, as they will register with their Master caches upon startup.)

   Following is a sample wan-config.xml file:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <wan-config xmlns="http://config.wan.terracottatech.com">
     <bind host="0.0.0.0" port="9003"/>
     <logs>/path/to/mylogs</logs>
     <maxConnectionRetryCount>8</maxConnectionRetryCount>
     <replicatorIntervalMillis>125</replicatorIntervalMillis>
     <replicationMode>unidirectional</replicationMode>
     <replicaDisconnectBehavior>reconnectResync</replicaDisconnectBehavior>
   ```

```
  <cacheManager ehcacheConfigURI="file:///path/to/ehcache-1.xml">
    <cache name="test-cache-1">
      <master host="masterhost-A" port="9001"/>
      <master host="masterhost-B" port="9002"/>
    </cache>
  <cache name="test-cache-2">
    <master host="masterhost-A" port="9001"/>
  </cache>
  </cacheManager>
  <cacheManager ehcacheConfigURI="file:///path/to/ehcache-2.xml">
    <cache name="test-cache-3">
      <master host="masterhost-B" port="9002"/>
    </cache>
  </cacheManager>
  // Optional for version 4.1.3 and above--see step 5 below
  <userClassDirectory>/temp/user_lib</userClassDirectory>
</wan-config>
```

Note: The ehcache.xml referenced in an Orchestrator's wan-config.xml must include the location of the tc-config.xml for that Orchestrator's region. The location is specified in the `terracottaConfig` element of the ehcache.xml. For more information, refer to "Defining a Distributed Configuration" in the *BigMemory Max Configuration Guide*.

For more information about the wan-config.xml parameters, see the section "Orchestrator Configuration Parameters" on page 18.

4. Start the Terracotta Server Array in each region.
5. (Optional) If you have any non-JDK value types in your caches, and/or if you are using custom attribute extractors for Search, you will need to add these to the classpath of the Orchestrator process.
   a. Place any jars containing custom attribute extractors or custom user classes in a directory.
   b. **For versions 4.1.0 through 4.1.2 only:** Define the environment variable WAN_USER_LIB to the absolute path of the directory. For example:

      ```
      export WAN_USER_LIB=/temp/user_lib
      ```

      The WAN shell script will then add all .JAR files under the /temp/user_lib directory to the Orchestrator's classpath.

      For version 4.1.3 and above only: Starting with 4.1.3, the WAN_USER_LIB has been deprecated. The functionality previously provided by WAN_USER_LIB is now handled by the userClassDirectory tag in each Orchestrator's wan-config.xml file. Following is a sample userClassDirectory tag in a wan-config.xml file:

      ```
      <userClassDirectory>/temp/user_lib</userClassDirectory>
      ```

      The WAN shell script will then add all .JAR files under the /temp/user_lib directory.
6. Use the start-wan script to start the Orchestrators.

   To specify the configuration file location, use -f command-line option, for example:

   ```
   <bigmemory_kit>/server/bin/start-wan.sh -f /path/to/wan-config.xml
   ```

7. Start your application. You must start the instances with the Master caches first, and then start those with Replica caches.

   Upon starting the app servers, your clusters should begin replicating data across the WAN.

Orchestrator Configuration Parameters

The following parameters of the wan-config.xml configure locations for the Orchestrator instance, its logs, and its Master cache locations.

- The `bind` settings should include the URL and port of the Orchestrator instance.
- The `logs` parameter takes the path to the location where you want the Orchestrator's logs to be collected.
- Each cache to be replicated must have at least one Master instance identified in the wan-config.xml.
- The `ehcacheConfigURI` should provide the path to the ehcache.xml configuration file for the cache.
- Each `master` parameter should provide the host name and port for the instance of the Master cache.

The defaults of the parameters in the wan-config.xml are optimized for most environments, but the following parameters can be adjusted to tune operation of your BigMemory WAN deployment.

- `maxConnectionRetryCount` - Specifies the maximum number of connection attempts a disconnected Replica cache should try for each Master cache, before attempting to connect to another Master cache in the wan-config.xml. The default is 5. Note that this applies only when the `replicaDisconnect behavior` is set for `reconnectResync` (see below).
- `replicatorIntervalMillis` - Specifies how frequently the Orchestrator should send replication updates to the regions, in milliseconds. Depending upon the amount of RAM available to your Orchestrator, you may be able to optimize WAN replication by increasing the interval.
- `replicationMode` - Selects the type of replication, unidirectional or bidirectional. If unspecified, the default is unidirectional. Bidirectional mode must be specified for active-active deployments. For more information, refer to "Replication Modes" on page 9 and "Bidirectional WAN Replication" on page 31.
- `replicaDisconnectBehavior` - Specifies the behavior of the Replica caches if they become disconnected from their Master cache. The options are:
  - `reconnectResync` (default) - When a Replica is disconnected from its Master, it attempts to reconnect to the Master and to any failover Masters listed in the wan-config.xml. Upon reconnection to a Master cache, the Replica is deactivated, cleared, resynchronized to the Master cache, and then reactivated. All local

changes on the Replica region will be dropped in favor of whatever is in the Master region. Note: When a WAN-enabled cache is deactivated, it follows the nonstop behavior specified in the ehcache.xml. For more information, refer to "Configuring Nonstop Operation" in the *BigMemory Max Configuration Guide*.

  - `remainDisconnected` - When a Replica is disconnected from its Master, it remains isolated from the Master from this point on. The Replica will not attempt to connect to the failover Masters listed in the wan-config.xml. The disconnected Replica cache does remain active locally, however, but no replication will take place.

- `monitoringEnabled` - Specifies that the monitoring capability for the WAN Replication Service is enabled for each Orchestrator. If you want to disable the monitoring capability, set the `monitoringEnabled` parameter to false in each Orchestrator's wan-config.xml file.

Best Practices and Configuration Considerations

Deployment

- The BigMemory WAN Replication Service should be run on a secure, trusted network.
- It is recommended to have a dedicated server for each region's Orchestrator, equipped to handle CPU- and network-heavy processes. While the hardware recommendation is similar to that of an L2 Terracotta server, the Orchestrator's server will not need as much RAM because it will be storing less data. Hardware sizing should take into account data element count, element size, frequency of element update, and bandwidth and latency across the WAN.
- It is recommended to have least one additional Orchestrator process in each region, to serve as a failover for the master Orchestrator in that region. Note that if the region contains a replica Orchestrator, then adding a standby replica Orchestrator is not always beneficial, because this can cause some slowness on Master caches during failover of the replica Orchestrator. For more information see the section "Automatic Recovery Scenarios" on page 28.
- Master caches should be located in the region where the most writes occur. This will give the lowest number of conflicts and highest possible performance.
- If possible, locate all of your Master caches in the same region.
- Avoid deployments where there is no terminal region. If your deployment has a cyclic dependency of Masters and Replicas, it would result in at least one Master cache going down if any given region went down, and lead to longer delays when recovering.
- The deployment that offers the best performance and consistency would have:
  - Distinct caches per region, where Master caches have most of the writes, and Replicas are effectively read-only or read/write-rarely.

- Distinct sets of cache keys for each region. Non-overlapping sets of cache keys between regions avoids conflicts because the cache mutability is isolated. For example, configure a counter for each region instead of using a global counter.
- Non-simultaneous updates of shared caches and keys. This means keys are updated across the WAN, but the application ensures they don't occur simultaneously.
- To support bi-directional use cases, recommend use of distinct caches (only updated in specific region), then replicated to other region

Configuration Rules and Limitations

The following should be taken into account when configuring the components for WAN replication:

- WAN Replication does not provide data compression
- Orchestrator cannot be enabled to use off-heap
- Cache cannot be split over multiple Orchestrators
- Cache events are not replicated. The resulting element state after a cache operation is replicated.
- Write-behind queues are not replicated.
- Explicitly locked caches will still function, but the lock will only be enforced locally.
- The bulk-load state is not replicated, because it is not possible to tell from one region that another region is in bulk-load mode. Bulk loading will otherwise work as normal.
- If you are using custom value classes that are not JDK types, the classes must be included in the Orchestrator's classpath. Refer to Step 5 in the section "General Steps for Setting Up WAN Replication" on page 16.
- The following cache configurations are not supported:
  - Transactional caches
  - Non-Ehcache caches
  - Unclustered (standalone) caches
  - TSAs with different segment counts
- Cache element expiration is only performed by the Master cache, and Orchestrators enforce Master cache expiries on all Replica caches. This means that Time To Idle (TTI) settings may not achieve expected results. Only the Master will perform last accessed time updates, so even if there are gets from the Replica regions, they will not count toward resetting the idle countdown on the element. It is recommended to avoid using TTI-based expiration with WAN replication.
- If you had configured cache size settings for Replica caches, they will be overridden so that the Master can control the cache.

Configuration Best Practices

- To improve performance, consider the following:
  - Divide data into 2 caches
  - Use Orchestrator per cache
  - Use 2 stripes if feasible (with high data volumes)
  - Above is a recommendation - more effective in environments with higher volumes
- Cache configurations should be as symmetrical as possible for the same cache in different regions. The cache size, as well as other settings that can impact the eviction of an element, should be the same across regions. These settings include the Time-To-Live option and the `maxEntriesInCache` attribute (configured in the ehcache.xml file), and the `dataStorage` and `offheap` elements (configured in the tc-config.xml file). For more information, see "Automatic Resource Management" in the *BigMemory Max Administrator Guide*.
- In order to see exceptions on a client when it fails to connect to an Orchestrator, you must set the client's nonstop `timeoutBehavior` type to "exception"; otherwise, you will only get the INFO message "Waiting for the Orchestrator to mark it active" in the event of an Orchestrator failure, or when a Replica gets disconnected from its Master. For more information, see "Configuring Nonstop Operation" in the *BigMemory Max Configuration Guide*.
- Avoid repeated and sustained conflicting writes to a set of elements over the WAN. This will result in constant conflict resolution and lower performance.
- Removing all elements should be avoided where possible. Caches may be cleared, but the Orchestrator never disposes of a cache. Even if all clients dispose of a cache, the Orchestrator will continue to hold it, and so it cannot be destroyed while the Orchestrator is running. It is recommended to avoid calling cache.clear() in your application.
- For a region that is used for disaster recovery, it is recommended to enable the Fast Restart feature. For more information, see "Configuring Fast Restart" in the *BigMemory Max Configuration Guide*.

4 Administering WAN Replication

- Working with the Replication Logs 24
- Using the Cleanup Utility 26
- Synchronizing Replica Caches 26
- About Fault Tolerance and Recovery 27

Working with the Replication Logs

BigMemory WAN Replication provides logging with messages that are easily parsable by third-party log watchers or scrapers. Logging includes:

- Topology changes
- Regularly print stats
- Number of mutations.
- Message Size
- Data conflicts and repairs
- Failover/recovery status/progress

Significant WAN replication log messages regard synchronization and incremental updates. The messages below provide important markers in the WAN replication logs.

Synchronization: Master Side

- Log message issued when the Master cache is started synchronizing a Replica cache:
  - INFO [master-0] FullScanMasterSynchronizer - Master 'localhost:9001' initiated sync protocol for replica 'localhost:9003' and cache '__tc_clustered-ehcache|CacheManager|test-cache-1'
- Log message issued when the Master cache is sending a synchronization batch to the Replica cache:
  - INFO [master-sync-13] FullScanMasterSynchronizer - Master 'localhost:9001' sent SYNC_UPDATE request with 7 events to replica 'localhost:9002' for cache 'tc_clustered-ehcache|DEFAULT__|wan-test-cache1'
- Log message posting the total number of SYNC_UPDATEs that should be processed by the Replica:
  - INFO [New I/O worker #1] FullScanMasterSynchronizer - Total number of SYNC_UPDATE requests submitted to replica 'localhost:9002' is 184
- Log message issued once the synchronization between Master and Replica has completed:
  - INFO [New I/O worker #4] FullScanMasterSynchronizer - Master 'localhost:9001' successfully synchronized replica 'localhost:9002' for cache 'tc_clustered-ehcache|DEFAULT__|wan-test-cache3'

Synchronization: Replica Side

- Log message issued when the Replica cache has started synchronzing with the Master cache:

    - INFO [replica-sync-4] FullScanReplicaSynchronizer - Replica got SYNC_START request from master 'localhost:9001' for cache 'tc_clustered-ehcache|DEFAULT__|wan-test-cache3'
- Log message issued when the Replica cache has processed a batch of synchronization updates from the Master cache:
    - INFO [replica-sync-2] FullScanReplicaSynchronizer - Replica got SYNC_UPDATE request from master 'localhost:9001' for cache 'tc_clustered-ehcache|DEFAULT__|wan-test-cache1' with 8 keys
- Log message issued when the Replica cache has completed synchronization with Master cache, but the Replica is not yet activated:
    - INFO [replica-sync-5] FullScanReplicaSynchronizer - Replica got SYNC_END request from master 'localhost:9001' for cache 'tc_clustered-ehcache|DEFAULT__|wan-test-cache3'
- Log message issued when the Replica cache is successfully activated after synchronization:
    - INFO [New I/O worker #1] ReplicaCache - Replica 'localhost:9002' activated cache 'tc_clustered-ehcache|DEFAULT__|wan-test-cache3' by request from master 'localhost:9001'

Incremental Updates

Log messages for incremental updates are for Bidirectional mode only. Note that watermarks are for batches of updates, which have been batched as part of the internal WAN process.

- Log messages issued when the Master is receiving acknowledgements of a Replica successfully storing a batch of updates in the TSA:
    - INFO [master-0] UnitReplicator - Replica cache 'tc_clustered-ehcache|DEFAULT__|wan-test-cache1@localhost:9002' was at watermark '1131', is now at '1199'
    - INFO [master-0] MasterCache - Lowest watermark across all replicas for cache 'tc_clustered-ehcache|DEFAULT__|wan-test-cache1' is now '1199'
- Log messages issued when a Replica has succesfully acklowledged the storage of the updates in the TSA to the Master cache:
    - INFO [New I/O worker #4] ReplicaCache - Replica 'tc_clustered-ehcache|DEFAULT__|wan-test-cache1@localhost:9002' successfully acknowledged watermark 1131 with master 'localhost:9001'
    - INFO [New I/O worker #4] ReplicaCache - Replica 'tc_clustered-ehcache|DEFAULT__|wan-test-cache1@localhost:9002' successfully acknowledged watermark 1199 with master 'localhost:9001'

Using the Cleanup Utility

If a cache that was registered with a TSA subsequently becomes inactive, then when the cache becomes active again, the TSA will attempt to verify that it has the same configuration. Therefore, in order to change the WAN enabled/disabled status of caches that have already been registered in a TSA, the existing WAN information will need to be removed. The cleanup-wan-metadata utility, found in the /server/bin directory of the BigMemory kit, is provided for this purpose.

To run the script:

```
cleanup-wan-metadata.sh -f "configLocation"
```

The script requires the `configLocation` argument, which is the URI location to either a wan_config.xml file or an ehcache.xml file.

If the `configLocation` specifies an ehcache.xml, the script performs cleanup all the caches mentioned in it. If the `configLocation` specifies a wan_config.xml, the script will look for all the ehcache.xml locations listed in it and clean up their caches.

The cleanup is performed for every cache, whether WAN-enabled or not. This is useful when you want to convert a WAN-enabled cache to non-WAN cache, or a non-WAN cache to a WAN-enabled cache.

Note that the script does not modify ehcache.xml or wan-config.xml files in any way. It simply removes WAN-related information from Terracotta servers.

Procedure to convert caches

In a wan-enabled system, if you need to EITHER convert a wan-enabled cache to a non-wan cache OR convert a non-wan cache to a wan-enabled-cache, follow the below steps:

1. Bring down all the clients and Orchestrator.
2. Modify the wan-config.xml and the ehcache.xml files as per the new configuration requirement.
3. Run the cleanup-wan-metadata script with the new configuration as the parameter.
4. Start the Orchestrator and clients again with the new configuration files.

Synchronizing Replica Caches

The Orchestrator activates a Replica cache when it has successfully synchronized the state of the cache with the Master cache. A Master cache can be immediately activated by an Orchestrator, as there is no synchronization to perform. Orchestrators managing Master caches perform continuous health-checking to verify the active status of the Replica caches managed by the other Orchestrators.

There are two main cases when the state of a Replica cache needs to be resynchronized by the Master cache: bootstrapping a new cache and recovering after a failure.

Bootstrapping a new cache

Perform these steps when starting a cache for the first time, or after the cache was fully cleared.

1. Ensure the Replica cache configuration is the same as the Master cache.
2. Ensure the WAN configuration for the Replica cache is valid:
   - Check that there is a valid list of Master caches.
   - For a given cache, the WAN configuration should be identical across all regions.
3. Ensure the TSA is running.
4. Start the Orchestrator.

On startup, the new Replica cache will be inactive while synchronizing (clients cannot use the cache). In this mode, it is receiving incremental updates and synchronizing the full state of the cache. Once fully synchronized, the Replica cache will be active.

Recovering after failure

BigMemory WAN Replication is built with fault tolerance features to automatically handle short-term failures and most longer-term failures.

When a replica reconnects with the master, its behavior is governed by the Orchestrator configuration parameter `replicaDisconnectBehavior`. If this parameter is set to:

- `remainDisconnected`, the replica will remain disconnected from the master, and will continue to operate offline.
- `reconnectResync`, the replica will reconnect to the master and will resync the contents of its cache. In this case, all local changes on the replica region will be dropped in favor of whatever is in the master region.

About Fault Tolerance and Recovery

Overview of Fault Recovery

Recovery from most failures is accomplished automatically by the WAN replication service, however some scenarios require user intervention.

The following sections describe:

- Automatic fault recovery
- Manual fault recovery

Automatic Recovery Scenarios

Orchestrator Failover

It is recommended to run multiple Orchestrators in a region, either on the same machine or rack, or in different racks to ensure availability. Although only one master Orchestrator is mandatory, you can start one or more standby Orchestrators at any time during application run time to provide failover protection.

Standby Orchestrators are passive, so running extra Orchestrator processes will have minimal runtime impact under normal operations. Upon failure, the other regions will look for the next available Orchestrator and resume replication (there will be a full sync for replica caches after master failover).

Note: The failover process for a Replica Orchestrator is not done in the same fault-tolerant manner as for failover of a Master Orchestrator. Once the standby Replica Orchestrator takes over, it re-syncs all Replica caches from scratch, then switches to normal replication mode. This behavior can cause some slowness if the Master cache contains a very large number of entries.

Master-Replica Disconnection

When a Master cache fails, control is given to the failover Master (if any) listed in wan-config.xml. A Replica cache will not take over as a Master. If there is no failover Master, the Replicas will continue to operate in isolation (i.e., no replication will take place). When the Master re-starts, the behavior of its Replicas is governed by the `replicaDisconnectBehavior` property in wan-config.xml. By default, the Replicas will attempt to reconnect to the Master and to the failover Master listed in wan-config.xml. Upon reconnection to a Master, the Replicas are deactivated, cleared, resynchronized to the Master cache, and then reactivated. All local changes on the Replica region will be dropped in favor of whatever is in the Master region. (Similarly, even if the Master does not fail but its Replicas become disconnected from their Master, their behavior is also controlled by the `replicaDisconnectBehavior` property.) For more information, refer to "Orchestrator Configuration Parameters" on page 18.

If no failover Master is listed in wan-config.xml and the master is lost, the operator has the option to restart a Replica to act as a Master, as described below.

In a bi-directional configuration without a failover Master, you should choose your most important Replica region to take over as the Master because any changes that occurred since the Master failed will be lost in all other regions. To do this, change your wan-config.xml to reflect that the Replica is now the Master, and then restart the Replica. It would be a good idea to remove or comment out the old Master in case it comes back.

In a uni-directional configuration without a failover Master, since the "writes" are only performed in one region, you will not lose any changes. You may simply designate that Replica region as the new Master.

TSA Disconnection

Upon any disconnection from its TSA, the Orchestrator deactivates and waits the amount of time specified by the `l2.l1reconnect.timeout.millis` property. This property is described in "Automatic Client Reconnect" in the *BigMemory Max High-Availability Guide*.

When communication between the TSA and the Orchestrator is resumed:

- If the downtime was shorter than the configured reconnect timeout, then WAN replication will resume immediately, without the need for any resynchronization.
- If the downtime exceeded the configured reconnect timeout, then the Orchestrator will resynchronize all of its Master caches, and subsequently those Master caches will resynchronize all of their Replica caches.

Cache Recovery Operations

- Unidirectional cache
  - In master region, cache operations remains operational even if local orchestrator went down
  - In replica region, cache operations remains operational even if local orchestrator went down but replica region won't receive any updates from the master region. Once orchestrator comes up, it will connect to master orchestrator and deactivates the cache (cache operations blocked) and does a full sync
    - Only if replicaDisconnectBehavior = reconnectResync (default).
- Bidirectional cache
  - In master region, cache operations remains operational even if orchestrator went down
  - In replica region, cache operations wait for the local orchestrator to come up and after local orchestrator comes up, it will connect to master region and does full sync
    - Only if replicaDisconnectBehavior = reconnectResync (default).
- With standby orchestrator, standby orchestrator becomes active orchestrator automatically and
  - in replica region, both unidirectional and bidirectional cache operations wait for full sync to be completed and cache activation.
    - Only if replicaDisconnectBehavior = reconnectResync (default).
  - in master region, both unidirectional and bidirectional cache operations continue to operate and new active orchestrator syncs all replica orchestrator.
    - Only if replicaDisconnectBehavior = reconnectResync (default).

Master Region Failure

If the master region fails entirely, there will most likely be data writes that were not completed (updated to Replica Region), though application may believe they were. As described in this section, recommended action would be to promote Replica region caches to master.

Manual Recovery

If a TSA failure takes place in a region with Master caches, upon recovery the Master caches cannot automatically force a resynchronization of all live Replicas across the WAN, because that could result in data loss. In this case, when an entire region is down (for example, your data center is offline), you must use a manual recovery process. You will need to designate new Master caches, update the configuration, and restart the Orchestrators.

In preparation for disaster recovery, two Orchestrator configurations should be ready:

Orchestrator Configuration	Region A	Region B
wan-config-1.xml	Master	Replica
wan-config-2.xml	Replica	Master

If Region A goes down, then Configuration 2 can be applied in Region B. Ideally, Region B would already be serving as a read-only or backup region. When Region B becomes the Master, it can then be used to resynchronize Region A.

5 Bidirectional WAN Replication

- About Bidirectional WAN Replication 32
- Configuring Bidirectional Mode 32
- Data Consistency and Conflict Resolution 32

About Bidirectional WAN Replication

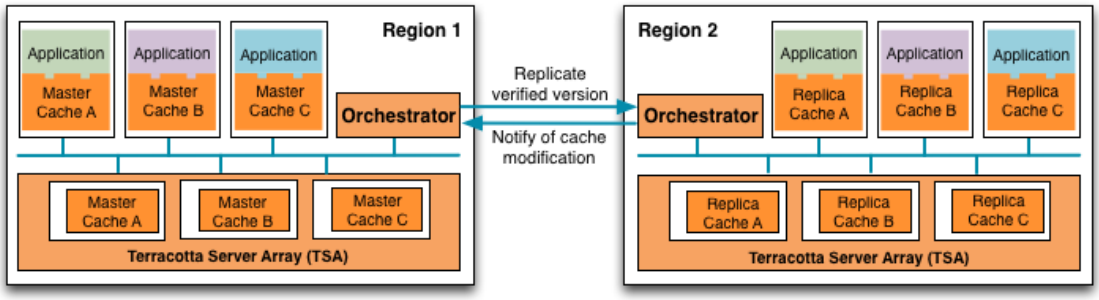
The BigMemory WAN Replication Service offers two replication modes: Unidirectional and Bidirectional. Unidirectional mode replicates data in one direction only, from Master to Replica, and is used for active-passive deployments such as disaster recovery. Bidirectional mode replicates data in two directions, from Master to Replica and from Replica to Master, and is used for active-active deployments where updates from more than one region should be incorporated into the Master cache's authoritative data set.

For comparison of unidirectional and bidirectional modes, see the section "Replication Modes" on page 9.

With Bidirectional mode, Replica caches notify their Master cache of any modification they receive, and the Master cache either applies or rejects the modification based upon its tracking of each cache element.

In the figure below, Master caches in Region 1 are distributed between the application servers and the Terracotta servers of the TSA. Cache modifications in either region are communicated through the Orchestrators, and the Master caches determine which modifications become part of the authoritative data set.

Bidirectional replication with Master caches in Region 1 and Replica caches in Region 2



Configuring Bidirectional Mode

The default mode is Unidirectional, and Bidirectional mode must be specified in the wan-config.xml. For configuration information, refer to the section "Orchestrator Configuration Parameters" on page 18.

Data Consistency and Conflict Resolution

With Bidirectional mode, the WAN replication service balances high performance with eventual data consistency through synchronization among Orchestrators. Two

additional strategies are employed for data consistency: element versioning and optimistic local modifications.

Note: When a key is getting updated simultaneously at high frequency from multiple regions, it is possible that even with conflict resolution, the key values will diverge. Therefore we recommend in bi-directional configurations to avoid conflicts when possible, in particular to use distinct caches (only updated in specific region), then replicated to other region. This will allow data to be replicated and available in multiple regions, but avoid any data conflict.

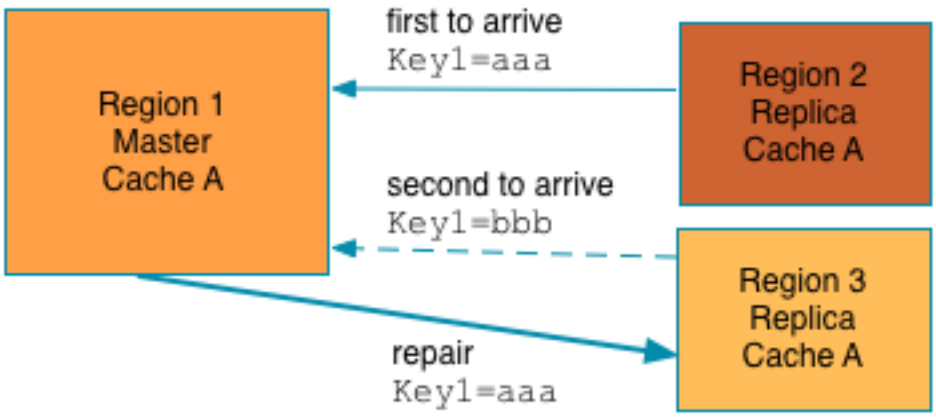
Element versioning

Element versioning is the core component of data consistency and conflict resolution. All modifications (puts/updates/deletes) to an element cause an internal version to be incremented once successfully applied. Versioning ensures there are no out-of-order updates applied to the Master cache (no duplicates, no backward counting, and no gaps), and versioning is used to resolve simultaneous conflicting updates.

Conflict resolution is invoked for two updates arriving for the same key with the same version, to determine which update is applied and which is rejected. The decision is made based upon the order in which the Master cache processes the updates. The first processed update wins.

The diagram below shows concurrent updates from Regions 2 and 3 on the same key (Key1) being resolved by the Master cache in Region 1. Region 2 (Key1=aaa) wins because its update arrives at the Master cache first. This results in a repair of Region 3, where Key1=bbb is repaired to Key1=aaa.

For the same version of the element, the Master cache repairs the second-to-arrive value to match the first-to-arrive value



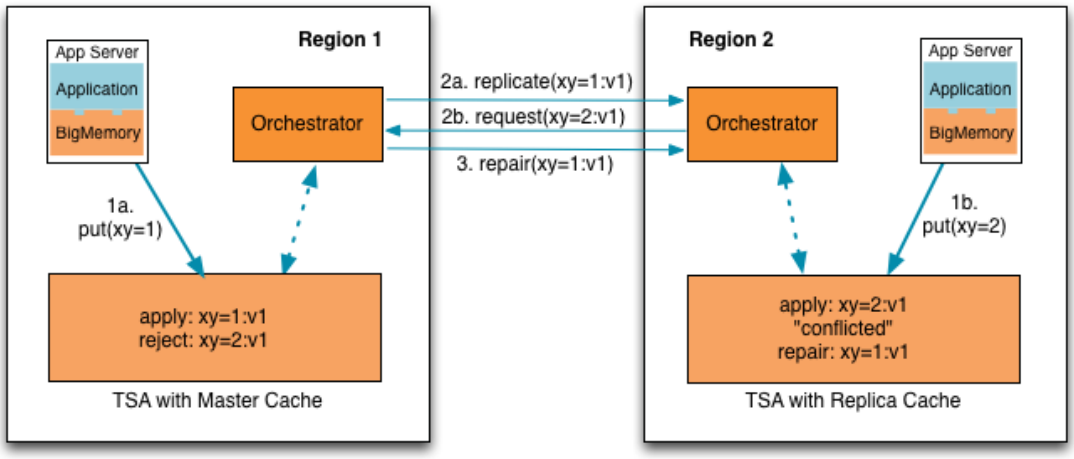
Optimistic local modifications with Master repair

To improve performance, local modifications (put/remove) are optimistically applied to the cache. Subsequent reads of the cache by the local application will return the expected modified value.

If there is a race to modify an element in the Master cache, the Master cache can force a repair of the Replica's value. A Master repair means that one of the racing modifications was rejected, and the Master cache has sent a repair message that causes the local element to now have the winning state. This can mean that a put may be overwritten with the winning value, or a losing remove may cause the local remove to be reverted.

In the following diagram, Region 1 performs a put (xy=1 at version 1), while Region 2 simultaneously performs put (xy=2 also at version 1). Because Region 1 contains the Master cache, it replicates xy=1:v1 to Region 2. And when the Master cache rejects the put from the Replica cache, it automatically resolves and sends a "repair" of the value back to Region 2.

Process of a Master cache repairing a Replica cache



After a repair, if there are any rejected values in any local caches in the region, they will be cleared with the existing process for non-replicated caches. For more information about expiration and eviction, refer to "Managing Data Life" in the *BigMemory Max Configuration Guide*.

6 Monitoring and Managing WAN Replication

- Overview of Monitoring and Managing WAN Replication 36
- Connecting via JMX 36
- Connecting via the REST APIs 38
- The Available Statistics 41

Overview of Monitoring and Managing WAN Replication

Terracotta enables you to monitor information about the following aspects of the WAN Replication Service:

- Your WAN's topology and configuration.

  In addition, Terracotta issues topology event alerts when a replica cache connects to or disconnects from a master cache.

- Each WAN-enabled cache, including performance statistics and details of their deployment, such as conflicts and status. Terracotta gathers statistics from all Orchestrators and displays them for each cache.

There are two ways to connect to the WAN Replication Service for monitoring:

- Connect via JMX. Terracotta exposes JMX MBeans to gather the statistics. You can use any JMX monitoring application such as JConsole or VisualVM to view the statistics.
- Connect via the REST APIs that Terracotta provides.

Note: This monitoring capability is enabled by default for each Orchestrator. If you want to disable the monitoring capability, set the `monitoringEnabled` parameter to false in each Orchestrator's wan-config.xml file. See "Orchestrator Configuration Parameters" on page 18.

Connecting via JMX

This section describes how to:

- "Set up to connect via JMX" on page 36.
- "Use JMX MBeans to display statistics in your JMX monitoring application" on page 37.
- "Programmatically access the statistics through JMX" on page 38 if you have configured your Orchestrator process to listen for JMX via RMI.

Setting up to Connect via JMX

You can access the statistics through JMX in either of the following ways:

- Via RMI on the Orchestrator process.

  To set up JMX to listen through RMI on the Orchestrator process, add the following arguments to the JAVA_OPTS environment variable before executing `start-wan.sh` :

```
JAVA_OPTS="-Dcom.sun.management.jmxremote.port=<number>
-Dcom.sun.management.jmxremote.authenticate=false
```

```
-Dcom.sun.management.jmxremote.ssl=false"
```

Note: Any additional Orchestrator process started on the same machine will fail with an "Address is in use" error if the JAVA_ OPTS environment variable is still visible. Thus JAVA_ OPTS should be set inside the script that starts the process, or else the port number should be changed.

- By communicating with the Orchestrator process directly. Simply start your WAN Region and connect your monitoring application to the Orchestrator process.

Using JMX Beans to Display Statistics

The Orchestrators expose one MBean for each cache and one MBean for each replica. Once connected to an Orchestrator process via JMX, you can use the Terracotta WAN-specific MBeans within the org.terracotta.wan domain to display statistics about:

- The overall state of the master cache and its replicas.
- A particular replica cache.

Displaying the Overall Statistics for a Cache and its Replicas

Use the following ObjectName form to display the overall statistics for a master cache and its replicas:

```
org.terracotta.wan:type=WAN,cacheManager="<name>",cache="<name>"
```

where:

- `cacheManager` specifies the name of the cache's cache manager, enclosed in quotation marks.
- `cache` specifies the name of the cache, enclosed in quotation marks.

For example:

The following ObjectName displays the overall cache-wide statistics for the cache `"test-cache-1"` in the cacheManager `"test"`.

```
org.terracotta.wan:type=WAN,cacheManager="test",cache="test-cache-1"
```

Displaying Statistics for a Particular Replica

To display only the statistics for a particular replica of the cache, append replica=<host_port> to the ObjectName, as follows:

```
org.terracotta.wan:type=WAN,cacheManager="<name>",cache="<name>",replica=<host_port>
```

where *<host_port>* specifies the host name and port of the replica. For example, `myhost_1234`. Do not enclose the value in quotation marks.

For example:

The following ObjectName displays statistics only for the cache's replica which has the hostname `myhost` and listens on port `1234` (as configured in the wan.xml for the replica).

```
org.terracotta.wan:type=WAN,cacheManager="test",cache="test-cache-1",
replica=myhost_1234
```

Programmatic Access through JMX via RMI

As a convenience, some classes have been added for programmatic access to the remote statistics. If you have configured your Orchestrator process to listen for JMX via RMI, you may use the RMIStatisticConnection class. In this example the Orchestrator is running on `"localhost"`, with an RMI port of `9999`. The cache to connect to is `"test-cache-1"` in the `"test"` cache manager, as in previous examples.

```
// Create the connection
StatisticConnection rmiStatisticConnection = new RMIStatisticConnection(
    "localhost", 9999, "test", "test-cache-1");
// Get a statistic
Gauge<String> g = rmiStatisticConnection.getStatistic(
    StatisticName.CONFIG_REPLICATION_MODE);
// Do something with the statistic
System.out.println(g.getValue());
// Close the connection
rmiStatisticsConnection.close();
```

For gauges whose values are non-numeric, calling the getValue() method again will fetch the latest value (i.e., you can call it in a loop and expect the value to change). The statistics client classes are inside wan-core.jar. In order to use it, pull in wan-core.jar as a Terracotta Maven Plugin dependency.

For gauges whose values are numeric, you can expect getValue() to eventually return the most recent value, but with a certain delay, depending on your cluster's configuration. For example, your implementation might throttle value updates; by default, the value is cached for 10 seconds, no matter how many actual updates take place during that interval.

Connecting via the REST APIs

The REST APIs are enabled automatically by default. No setup is required.

The REST endpoint for accessing WAN statistics is not available directly from the Orchestrator process. Rather, it is available from the Terracotta Management Server (TMS) and the first stripe of the Terracotta Server Array (TSA).

For example, the following would return a JSON object containing all the WAN-replicated cache managers in the TSA located on `"localhost"` with a management port of `46266`.

```
http://localhost:46266/tc-management-api/v2/agents/wan/cacheManagers
```

This would return:

```
{
    "agentId": "embedded",
    "apiVersion": "v2",
    "entities": [
      {
                "agentId": "127.0.0.1_63613",
                "cacheManagerName": "test",
                "cacheNames": [
                  "test-cache-1"
                ]
```

```
      }
    ],
    "exceptionEntities": [
    ]
}
```

To narrow the view to just a particular cache manager, you can add `names=<cache manager name>` to the above URL as follows:

```
http://localhost:46266/tc-management-api/v2/agents/wan/cacheManagers;names=test
```

To get the list of caches inside the cache manager, you can further drill down by appending `/caches tonames=<cache manager name>` as follows:

```
http://localhost:46266/tc-management-api/v2/agents/wan/cacheManagers;names=test/caches
```

This would return:

```
{
    "agentId": "embedded",
    "apiVersion": "v2",
    "entities": [
       {
           "agentId": "127.0.0.1_63613",
           "cacheManagerName": "test",
           "cacheName": "test-cache-1",
           "attributes": {
               "cluster-listener.buffer.size": 0,
               "topology.replicas": [
                   "localhost:11477"
               ],
               "topology.master": "localhost:43051",
               "config.replication-mode": "UNIDIRECTIONAL"
              }
       }
    ],
    "exceptionEntities": [
    ]
}
```

The statistics returned from the REST interface are the same as those returned from the JMX interface. They are found inside the attributes map in the returned JSON object.

If you want to drill down into a replica's statistics for a given cache, you can append `/replicas tonames=<cache manager name>/caches`. The following would give you the full list of connected replicas with their relevant statistics:

```
http://localhost:46266/tc-management-api/v2/agents/wan/cacheManagers;names=test/
caches/replicas
```

The following would return only the statistics for the given replica at localhost_25394.

```
http://localhost:46266/tc-management-api/v2/agents/wan/
cacheManagers;names=test/caches/replicas;names=localhost_25394
```

For each level of statistics, it is also possible to narrow the view to show only a single statistical attribute by specifying the name of attribute with the `?show=` parameter. For example, to restrict the returned attributes to only show the `sync.tps.Count` attribute, specify this:

```
http://localhost:46266/tc-management-api/v2/agents/wan/cacheManagers;
names=test/caches/replicas;names=localhost_25394?show=sync.tps.Count
```

Ths would return:

```
{
    "agentId": "embedded",
    "apiVersion": "v2",
    "entities": [
       {
           "agentId": "127.0.0.1_63613",
           "cacheManagerName": "test",
           "cacheName": "test-cache-1",
           "attributes": {
               "sync.tps.Count": 0,
           },
           "replica": "localhost_25394"
       }
    ],
    "exceptionEntities": [
    ]
}
```

As a convenience, it is also possible to select attributes based on a prefix. This would be useful for returning all the views of a given statistic, or for returning all statistics in a given category. For example:

```
http://localhost:46266/tc-management-api/v2/agents/wan/cacheManagers;
names=test/caches;names=test-cache-1?showPrefix=topology
```

The above example would return all attributes beginning with `topology`, as follows:

```
{
  "agentId": "localhost_62890",
  "entities": [
       {
          "agentId": "embedded",
          "cacheManagerName": "test",
          "cacheName": "test-cache-1",
          "attributes": {
                                "topology.replicas": [],
                                "topology.master": "localhost:32567"
                                   }
       }
    ],
   "exceptionEntities": [],
   "apiVersion": "v2"
}
```

You can view the configuration for a given orchestrator as follows:

```
http://localhost:46266/tc-management-api/v2/agents/wan/config
```

This would return:

```
{
   "agentId":"embedded",
   "apiVersion":"v2",
   "entities":[
      {
         "agentId":"127.0.0.1_63613",
         "config":
             "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n
                <wan-config xmlns=\"http://config.wan.terracottatech.com\">\n
                   <bind host=\"10.60.29.127\" port=\"9001\"/>\n
                   <logs>logs</logs>\n
                   <monitoringEnabled>true</monitoringEnabled>\n\n
```

```
                    <cacheManager ehcacheConfigURI=\"file://<path to ehcache>/ehcache.xml\">\n
                       <cache name=\"one\">\n
                          <master host=\"masterip\" port=\"9001\"/>\n
                       </cache>\n
                       <cache name=\"two\">\n
                          <master host=\"master ip\" port=\"9001\"/>\n
                       </cache>\n
                    </cacheManager>\n
                 </wan-config>\n"
      }
   ],
   "exceptionEntities":[
   ]
}
```

The Available Statistics

You can monitor the following information about the WAN Replication Service.

The Master Cache Statistics

The following statistics report on the state of the master cache. They are collected by the master Orchestrator.

Statistic	Description
cluster-listener.buffer.size	Shows the size of the cluster listener buffer.
config.replication-mode	Shows the replication mode (unidirectional or bidirectional).
conflict.count	Shows the number of conflicts that were resolved.
conflict.table.size	Shows the number of element modification entries (puts/updates/deletes) that are currently outstanding. These entries may or may not have had a conflict; it simply means that they have not yet been replicated to all the replicas in the WAN. This number could be higher than conflict.count.
replication.tps	Shows the average transaction rate of the replicas and the count. This number will tend towards the slowest replica in the cluster because the cluster can only replicate as fast as the slowest replica. This is broken down as follows:

Statistic	Description
	■ replication.tps.OneMinuteRate: The average transactions per second (tps) rate during the previous one minute. ■ replication.tps.FiveMinuteRate: The average tps rate during the previous five minutes. ■ replication.tps.FifteenMinuteRate: The average tps rate during the previous fifteen minutes. ■ replication.tps.MeanRate: The average tps rate for your application's entire lifetime. ■ replication.tps.Count: Total number of transactions during your application's entire lifetime.
topology.master	Shows the host name and address of the cache's master Orchestrator.
topology.replicas	Shows the host name and address of each of the cache's replica Orchestrators that are connected.
topology.standby-masters	Shows the host name and address of each standby (or mirror) server in your cluster.

The Replica Statistics

The following statistics report on the overall state of the replicas of the master cache. They are collected by the replica Orchestrator.

Statistic	Description
config.replication-mode	Shows the replication mode (unidirectional or bidirectional).
replica.tps	Shows the average update rate of the replicas and the total number, broken down as follows: ■ replica.tps.OneMinuteRate: The average transactions per second (tps) rate during the previous one minute. ■ replica.tps.FiveMinuteRate: The average tps rate during the previous five minutes. ■ replica.tps.FifteenMinuteRate: The average tps rate during the previous fifteen minutes.

Statistic	Description
	■ replica.tps.MeanRate: The average tps rate for your application's entire lifetime. ■ replica.tps.Count: Total number of transactions during your application's entire lifetime.
topology.master	Shows the host name and address of the cache's master Orchestrator.
topology.replicas	Shows the host name and address of each of the cache's replica Orchestrators that are connected.
topology.standby-masters	Shows the host name and address of each standby (or mirror) server in your cluster.

The Replica-Specific Statistics

The following statistics report on a particular replica cache. They are collected by the master Orchestrator.

Statistic	Description
replica.status	Shows the connection status of this replica.
replica.tps	Shows the incremental update rate of this replica, in transactions per second (tps), broken down as follows: ■ replica.tps.OneMinuteRate: The average tps during the previous one minute. ■ replica.tps.FiveMinuteRate: The average tps during the previous five minutes. ■ replica.tps.FifteenMinuteRate: The average tps during the previous fifteen minutes. ■ replica.tps.MeanRate: The average tps for your application's entire lifetime. ■ replica.tps.Count: Total number of transactions during your application's entire lifetime.
replicator.buffer.size	Shows the number of entries in the outbound buffer that are going to this replica.

Statistic	Description
sync.status	Shows the synchronization status of this replica.
sync.tps	Shows the synchronization rate of this replica, in transactions per second (tps), broken down as follows: ■ sync.tps.OneMinuteRate: The average tps during the previous one minute. ■ sync.tps.FiveMinuteRate: The average tps during the previous five minutes. ■ sync.tps.FifteenMinuteRate: The average tps during the previous fifteen minutes. ■ sync.tps.MeanRate: The average tps during your application's entire lifetime. ■ sync.tps.Count: Total number of transactions for your application's entire lifetime.
topology.master	Shows the host name and address of the cache's master Orchestrator.
topology.replicas	Shows the host name and address of each of the cache's replica Orchestrators that are connected.
topology.standby-masters	Shows the host name and address of each standby (or mirror) server in your cluster.