Terracotta Developer Console

About Terracotta Documentation

This documentation is about Terracotta DSO, an advanced distributed-computing technology aimed at meeting special clustering requirements.

Terracotta products without the overhead and complexity of DSO meet the needs of almost all use cases and clustering requirements. To learn how to migrate from Terracotta DSO to standard Terracotta products, see Migrating From Terracotta DSO. To find documentation on non-DSO (standard) Terracotta products, see Terracotta Documentation. Terracotta release information, such as release notes and platform compatibility, is found in Product Information.

- **Introduction**
- How DSO Clustering Works
- Platform Concepts
- Hello Clustered World
- **Setup and Configuration**
- Planning for a Clustered App
- Configuring Terracotta DSO
- Configuration Reference
- Installation
- **APIs**
- Using Annotations
- Cluster Events
- Data Locality Methods
- Distributed Cache
- Clustered Async Data Processing
- **Tool Guides**
- Developer Console
- Operations Center
- tim-get (TIM Management Tool)
- Platform Statistics Recorder
- Eclipse Plugin
- Sessions Configurator
- Clustering Spring Webapp with Sessions Configurator
- Maven
- JMX
- **Testing, Tuning, and Deployment**
- Top 5 Tuning Tips
- Testing a Clustered App
- Tuning a Clustered App
- Deployment Guide
- Operations Guide
- **FAQs and Troubleshooting**
- General FAQ
- DSO Technical FAQ
- Troubleshooting Guide
- Gotchas
- Non-portable Classes
- **Reference**
- Migrating From DSO
- Concept and Architecture Guide
- Examinator Reference Application
- Clustered Data Structures Guide
- Integrating Terracotta DSO
- Clustering Spring Framework
- Integration Modules Manual
- AspectWerkz Pattern Language
- Glossary

Release: 3.6
Publish Date: November, 2011 **Documentation Archive »**

Unknown macro: {HTMLcomment}

UI stuff:

- server info:

    - Main: says "Failover mode disk-based-active-passive" even with only 1 server in cluster
- Cluster "recording on" light should be doc in Lock Profiler and Statistics Recorder, and where ever else it's triggered from
- see lock profiler blog for more info: http://blog.terracottatech.com/2008/04/lock_profiling_across_jvms_in_1.html

Terracotta Developer Console

The Terracotta Developer Console delivers a full-featured monitoring and diagnostics tool aimed at the development and testing phases of an application clustered with Terracotta. Use the Developer Console to isolate issues, discover tuning opportunities, observe application behavior with clustering, and learn how the cluster holds up under load and failure conditions. The console functions as a JMX client with a graphical user interface.

Enterprise versions of Terracotta also include the Terracotta Operations Center, a GUI operator's console offering features such as backups of shared data, client disconnect, and server shutdown controls. To learn more about the many benefits of an enterprise version of Terracotta, see our [Enterprise Products|enterprise:Products].

Using the console, you can perform the following tasks:

- Monitor and manage applications using Ehcache, Ehcache for Hibernate, Quartz Scheduler, and Web Sessions.
- Monitor the health of servers and clients under changing conditions.
- Trigger distributed garbage collection operations.
- View cluster-wide statistics.
- Record cluster statistics for later display.
- Receive console status and server log messages in a console window.
- Find exactly which classes are being shared in the cluster.

For complete documentation on the Terracotta Developer Console, see .

More information on the following topics is available in other Terracotta documentation:

- JMX Port
- Roots
- Distributed Garbage Collection
- Object Portability

Unknown macro: {HTMLcomment}

These and other console features are described below.

Many of the images in this document can be viewed in a larger format by clicking them.

Launching the Terracotta Developer Console

You can launch the Terracotta Developer Console from a command line.

Microsoft Windows

```
[PROMPT] %TERRACOTTA_HOME%\bin\dev-console.bat
```

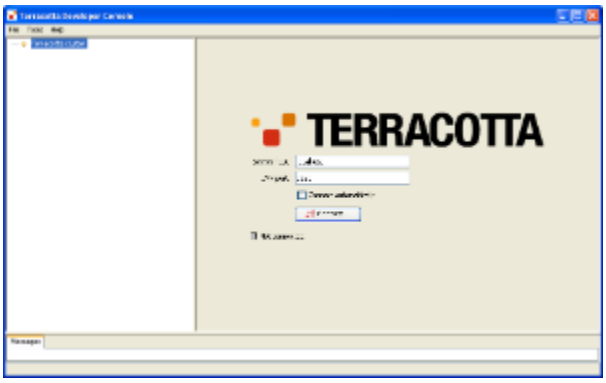
UNIX/Linux

```
[PROMPT] ${TERRACOTTA_HOME}/bin/dev-console.sh&
```

When the console first starts, it waits until every Terracotta server configured to be active has reached active status before fully connecting to the cluster.

The Console Interface

When not connected to a server, the console displays a connect/disconnect panel, message-log window, status line, and an inactive cluster node in the clusters panel.



The cluster list in the clusters panel could already be populated because of pre-existing references to previously defined Terracotta clusters. These references are maintained as Java properties and persist across sessions and product upgrades. If no clusters have been defined, a default cluster (host=localhost, jmx-port=9520) is created.

> **Setting a JMX Port**
>
> To learn more about setting JMX ports, see the Configuration Guide and Reference.

Once the console is connected to a cluster, the cluster node in the clusters panel serves as the root of an expandable/collapsible tree with nested displays and controls. More than one cluster node can appear in the clusters panel.

Console Messages

Click **Messages** in the Status panel to view messages from the console about its operations.

Menus

The following menus are available from the console's menu bar.

File

- New Cluster – Create a new cluster node in the clusters panel.
- Quit – Shut down the console application. Has no effect on the cluster except to reduce load if the console has been recording statistics or profiling locks.

Tools

- Show SVT – Opens the Terracotta Snapshot Visualization Tool.
- Options – Opens the Options dialog.

Help

- Developer Console Help – Go to the Developer Console documentation.
- Visit Terracotta Forums – Go to the community forums to post questions and search for topics.
- Contact Terracotta Technical Support – Go to the contact page for Terracotta technical support.
- Check for Updates – Automatically check for updates to Terracotta.
- Check Server Version – Automatically check for console-server version mismatch.
- About Terracotta Developer Console – Display information on Terracotta and the local host.

Context-Sensitive Help

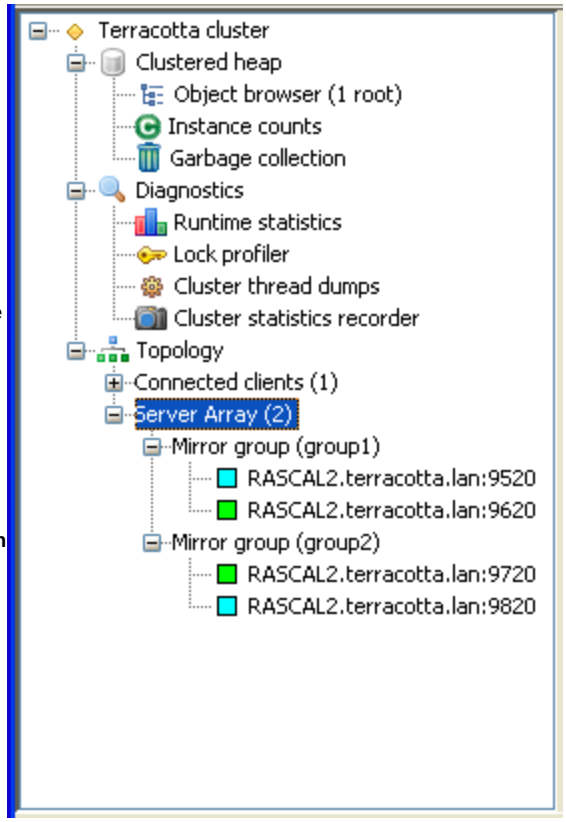
Context-sensitive help is available wherever (help button) appears in the Terracotta Developer Console. Click in a console panel to open a web-browser page containing help on the features in that panel.

Context Menus

Some console features have a context menu that is accessed by right-clicking the feature. For example, to open a context menu for creating a new cluster root in the clusters panel, right-click in the clusters panel.

Working with Clusters

Clusters are the highest-level nodes in the expandable cluster list displayed by the Terracotta Developer Console. A single Terracotta cluster defines a domain of Terracotta server instances and clients (application servers) being clustered by Terracotta. A single Terracotta cluster can have one or more servers and one or more clients. For example, two or more Terracotta servers configured as a server array, along with their clients, appear under the same cluster.



Adding and Removing Clusters

To add a new cluster reference, choose **New cluster** from the **File** or context menu.

The cluster topology is determined from the server specified in the connection panel's **Server Host** and **JMX Port** fields. These fields are editable when the console is not connected to the cluster.

To remove an existing cluster reference, right-click the cluster in the cluster list to open the context menu, then choose **Delete**.

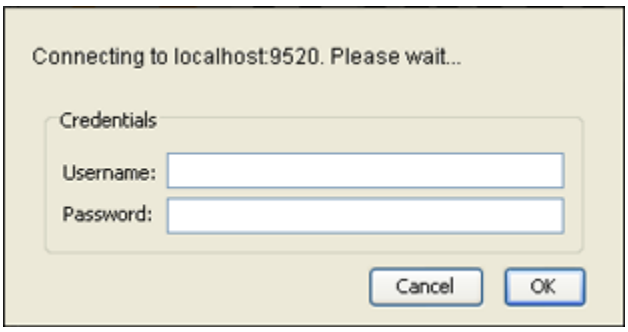
Connecting to a cluster

To connect to an existing cluster, select the cluster node in the cluster list, then click the **Connect** button in the connect/disconnect panel. You can also connect to a specific cluster by choosing **Connect** from its context menu. After a successful connection, the cluster node becomes expandable and a connection message appears in the status bar.

To automatically connect to a cluster whenever the Terracotta Developer Console starts or when at least one of the cluster's servers is running, enable **Auto-connect** in the cluster context menu. Automatic connections are attempted in the background and do not interfere with normal console operation.

Connecting to a Secured Cluster

A Terracotta cluster can be secured for JMX access, requiring authentication before access is granted. Connecting to a secured cluster prompts users to enter a username and password.



For instructions on how to secure your Terracotta cluster for JMX, see the Configuration Guide and Reference.

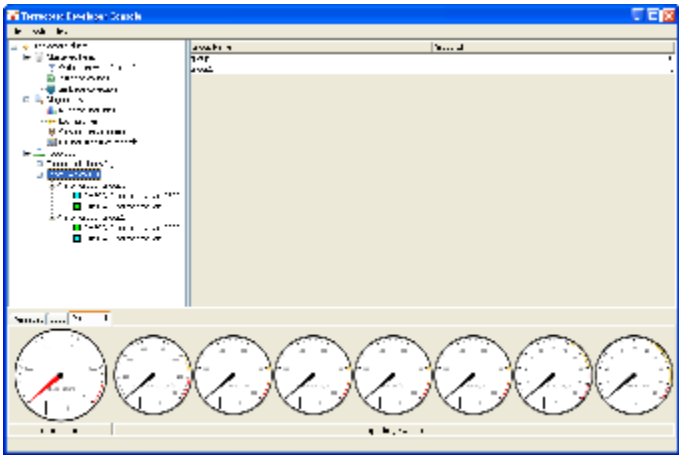
Disconnecting from a Cluster

To disconnect from a cluster, select that cluster's node in the on the clusters panel and either click the **Disconnect** button above the help panel or select **Disconnect** from the cluster context menu.

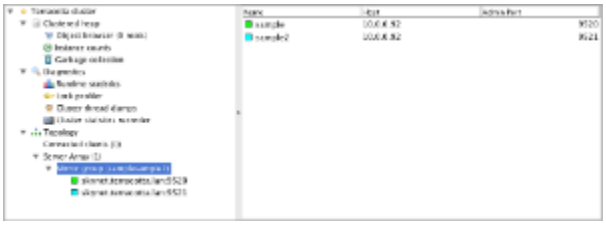
Working with Terracotta Server Arrays

Terracotta servers are arranged in *mirror groups*, each of which contains at least one active server instance. A high-availability mirror group also contains one backup server instance, sometimes called a passive server or "hot standby." Under the **Topology** node, a **Server Array** node contains all of the mirror groups in the cluster.

To view a table of mirror groups with their group IDs, expand the **Topology** node, then click **Server Array**.



To view a table of the servers in a mirror group, expand the **Server Array** node, then click the mirror group whose servers you want to display. The table of servers includes each server's status and name, hostname or IP address, and JMX port.



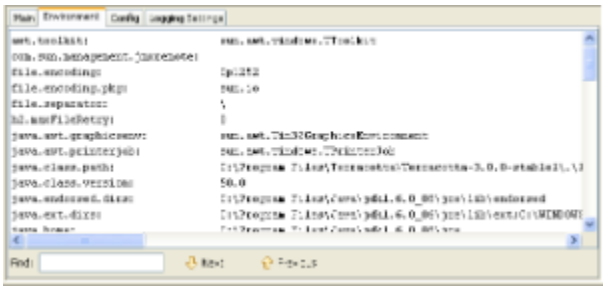
To view the servers' nodes under a mirror-group node, expand the mirror-group node.

Server Panel

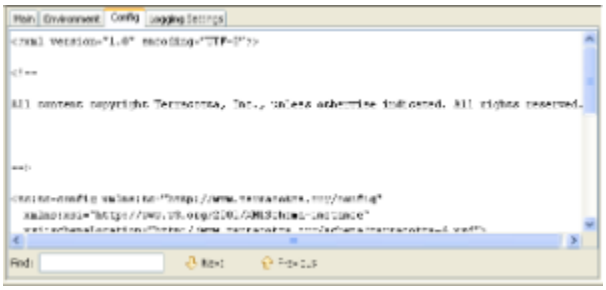
Selecting a specific server's node displays that server's panel, with the **Main**, **Environment**, **Config** , and **Logging Settings** tabs.

The **Main** tab displays the server status and a list of properties, including the server's IP address, version, license, and persistence and failover modes.

The **Environment** tab displays the server's JVM system properties and provides a **Find** tool.



The **Config** tab displays the Terracotta configuration the server is using and provides a **Find** tool.



The **Logging Settings** tab displays logging options, each of which reports on data captured in five-second intervals:

- FaultDebug – Logs the types of objects faulted from disk and the number of faults by type.
- RequestDebug – Logs the types of objects requested by clients and the number of requests by type.
- FlushDebug – Logs the types of objects flushed from clients and a count of flushes by type.
- BroadcastDebug – Logs the types of objects that changed and caused broadcasts to other clients, and a count of those broadcasts by type.
- CommitDebug – Logs the types of objects committed to disk and a count of commits by type.

Connecting and Disconnecting from a Server

The Terracotta Developer Console connects to a cluster through one of the cluster's Terracotta servers. Being connected to a server means that the console is listening for JMX events coming from that server.

> ⚠ If you have confirmed that a Terracotta server is running, but the Terracotta Developer Console is unable to connect to it, a firewall on your network could be blocking the server's JMX port.

The console is disconnected from a cluster's servers when it's disconnected from the cluster. The console is also disconnected from a server when that server is shut down, even though the server may still appear in the console as part of the cluster. A server's connection status is indicated by its status light.

Note that disconnecting from a server does not shut the server down or alter its status in the cluster. Servers can be shut down using the [stop-tc-server script].

> ✓ A server shutdown button is available in the Terracotta Operations Center.

Server Connection Status

A Terracotta server's connection status is indicated by a status light next to the server's name. The light's color indicates the server's current connection status. A cluster can have one server, or be configured with multiple servers that communicate state over the network or use a shared file-system.

The following table summarizes the connection status lights.

Status Light	Server Status	Notes
GREEN	Active	The server is connected and ready for work.
RED	Unreachable	The server, or the network connection to the server, is down.
YELLOW	Starting or Standby	A server is starting up; in a disk-based multi-server cluster, a passive server goes into standby mode until a file lock held by the active server is released.
ORANGE	Initializing	In a network-based multi-server cluster, a passive server must initialize its state before going into standby mode.
CYAN	Standby	In a network-based multi-server cluster, a passive server is ready to become active if the active server fails.

Working with Clients

Terracotta clients that are part of the cluster appear under the **Connected clients** node. To view the **Connected clients** node, expand the **Topology** node. The **Connected clients** panel displays a table of connected clients. The table has the following columns:

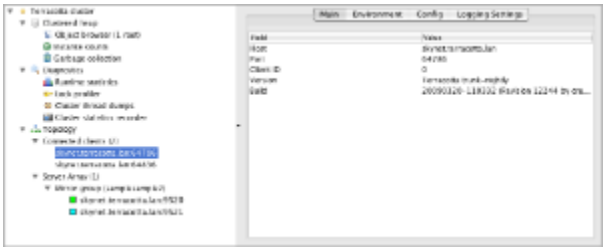
- Host - The client machine's hostname.
- Port - The client's DSO Port.
- ClientID - The client's unique ID number.
- Live Objects - The number of shared objects currently in the client's heap.

To view the client nodes that appear in the **Connected clients** panel, expand the **Connected clients** node.

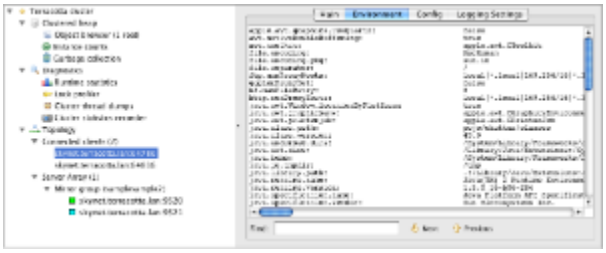
Client Panel

Selecting a specific client's node displays that client's panel, with the **Main**, **Environment**, **Config**, and **Logging** tabs.

The **Main** tab displays a list of client properties such as hostname and DSO port.

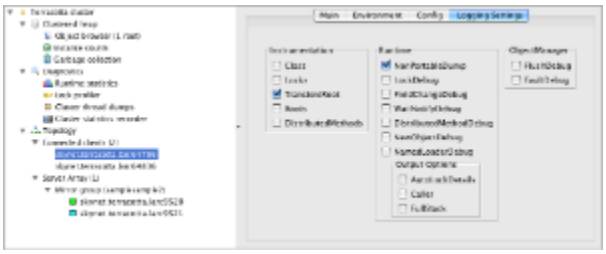


The **Environment** tab displays the client's JVM system properties and provides a **Find** tool.



The **Config** tab displays the Terracotta configuration the client is using and provides a **Find** tool.

The **Logging** tab displays options to add logging items corresponding to DSO client debugging. See the Configuration Guide and Reference for details on the various debug logging options.



Connecting and Disconnecting Clients

When started up properly, a Terracotta client is automatically added to the appropriate cluster.

When a Terracotta client is shut down or disconnects from a server, that client is automatically removed from the cluster and no longer appears in the Terracotta Developer Console .

A client disconnection button is available in the Terracotta Operations Center.

Monitoring Clusters, Servers, and Clients

The Terracotta Developer Console provides visual monitoring functions using dials, icons, graphs, statistics, counters, and both simple and nested lists. You can use these features to monitor the immediate and overall health of your cluster as well as the health of individual cluster components.

Shared Objects

Applications clustered with Terracotta use shared objects to keep data coherent. Monitoring shared objects serves as an important early-warning and troubleshooting method that allows you to:

- Confirm that appropriate object sharing is occurring;
- be alerted to potential memory issues;
- learn when it becomes necessary to tune garbage collection;
- locate the source of object over-proliferation.

The Terracotta Developer Console provides the following tools for monitoring shared objects:

- Object Browser
- Classes browser
- Client Flush and Fault Rate graphs
- Cache Miss Rate Graph
- Runtime logging

These tools are discussed in the following sections.

Object Browser

The Object Browser is a panel displaying shared object graphs in the cluster. To view the Object Browser, expand the **Clustered heap** node, then click the **Object browser** node.

The Object Browser does not refresh automatically. You can refresh it manually in any of the following ways:

- Expand or collapse any part of any object graph.
- Press the F5 key on your keyboard.
- Right-click any item on the object graph that has an object ID (for example, @1001), then select **Refresh** from the context menu.

The following are important aspects of the object graph display:

- The top-level objects in an object graph correspond to the shared roots declared in the Terracotta server's configuration file.
- Objects referencing other objects can be expanded or collapsed to show or hide the objects they reference.
- Objects in the graph that are a collections type, and reference other objects, indicate the number of referenced objects they display when expanded.
  This number is given as a ratio in the format `[X/Y]`, where X is the number of child elements being displayed and Y is the total number of child elements in the collection. Collections also have **More** and **Less** items in their context menus for manual control over the number of child elements displayed. By default, up to ten children (fields) are displayed when you expand a collections-type object in the object graph.
- A delay can occur when the object browser attempts to display very large graphs.
- An entry in the graph that duplicates an existing entry has an "up" arrow next to it.
  Click the up arrow to go to the existing entry.

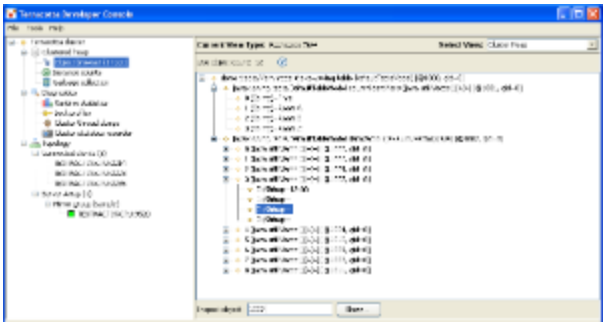
- An entry in the graph called "Collected" with no data indicates an object that was made known to the console but no longer exists on the graph. The collected object will eventually disappear from the graph on refresh.
- Each element in the object appears with unique identifying information, as appropriate for its type. Each object appears with its fully qualified name.

To inspect a portion of an object graph, follow these steps:

1. Find the object ID of the object at the root of the portion you want to inspect. An object ID has the format @<integer>. For example, @1001 can be an object ID.
2. Enter that object ID in **Inspect object**.
3. Click **Show...**. A window opens containing the desired portion of the graph.

Cluster Object Browsing

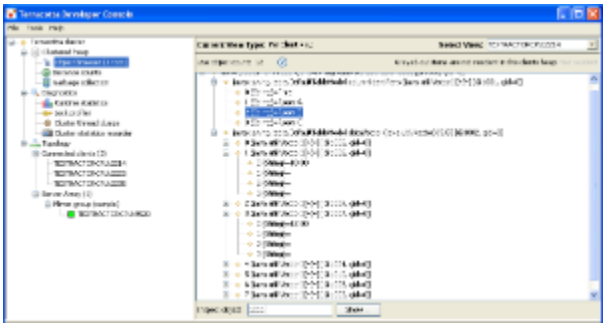
To browse the shared-object graphs for the entire cluster, select **Cluster Heap** from the **Select View** menu. All of the shared objects in the cluster-wide heap are graphed, but the browser doesn't indicate which clients are sharing them. To see object graphs specific to a client, see Client Object Browsing.



The browser panel displays a running total of the live objects in the cluster. This is the number of objects currently found in the cluster-wide heap; however, this total does not correspond to the number of objects you see in the object graph because certain objects, including literals such as strings, are not counted. These uncounted objects appear in the object graph without an object ID.

Client Object Browsing

To browse the shared-object graphs in a specific client, select the client from the **Select View** menu. All of the shared object graphs known to that client are graphed, but the ones not being shared by the client are grayed out.



The browser panel displays a running total of the live objects in the client. This is the number of objects currently found in the client heap; however, this total does not correspond to the number of objects you see in the object graph because the following types of objects are not counted:

- Objects *not* being shared by the client
  These unshared objects are grayed out in the object-browser view.
- Literals such as strings
  These objects appear in the object graph without an object ID.

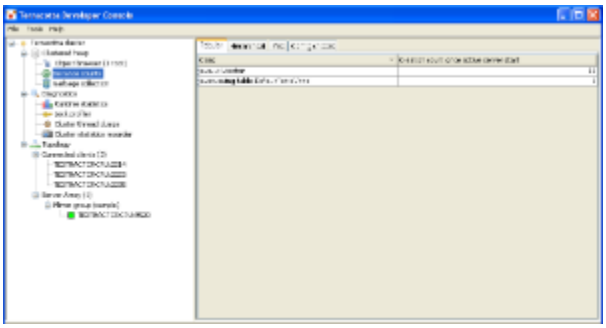
Classes Browser

Terracotta allows for transparent, clustered object state synchronization. To accomplish this feature, some of your application classes are adapted into new classes that are cluster-aware. Snapshots of the set of all such adapted classes known to the server are displayed in the **Instance counts** panel. The panel has the following tabs:

- **Tabular** – Lists all the adapted classes in a spreadsheet view, including the class name and a count of the number of instances of the class that have been created since the server started. Click the column title to sort along that column's contents.
- **Hierarchical** – Presents an expandable/collapsable Java package view of the adapted classes, along with a count of the number of instances of the class that have been created since the server started.
- **Map** – Displays an area map distinguishing the most (and least) heavily used adapted classes.

- **Config snippet** – A snippet from the <application> section of the Terracotta configuration file showing the how the instrumented classes are configured.

To refresh the values in the classes browser, select **Refresh** from the **Instance counts** context menu.



Client Flush and Fault Rate Graphs

Client flush and fault rates are a measure of shared data flow between Terracotta servers and clients. These graphs can reflect trends in the flow of shared objects in a Terracotta cluster. Upward trends in flow can indicate insufficient heap memory, poor locality of reference, or newly changed environmental conditions. For more information, see Client Flush Rate and Client Fault Rate.

Cache Miss Rate Graph

The Cache Miss Rate measures the number of client requests for an object that cannot be met by a server's cache and must be faulted in from disk. An upward trend in this graph can expose a bottleneck in your cluster. For more information, see Cache Miss Rate.

Runtime Logging of New Shared Objects

You can log the creation of all new shared objects by following these steps:

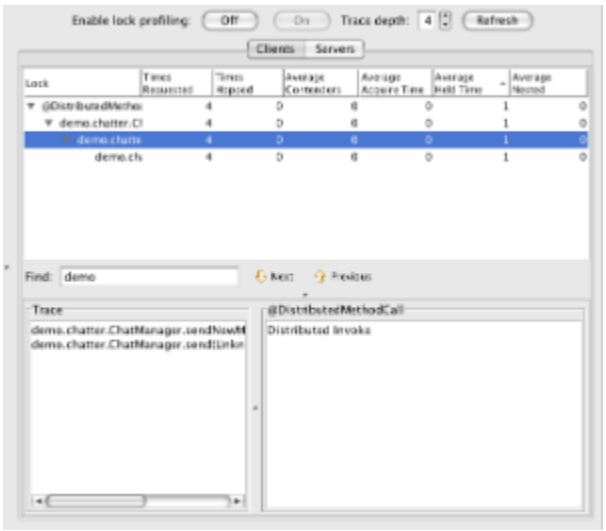
1. Select the target client in the cluster list.
2. Click the **Logging Settings** tab.
3. Enable **NewObjectDebug** from the Runtime list.

During development or debugging operations, logging new objects can reveal patterns that introduce inefficiencies or errors into your clustered application. However, during production it is recommended that this type of intensive logging be disabled.

See the Configuration Guide and Reference for details on the various debug logging options.

Lock profiler

The Terracotta runtime system can gather statistics about the distributed locks set by the Terracotta configuration. These statistics provide insight into the workings of a distributed application and aid the discovery of highly-contented access to shared state. Statistics are displayed in a table.



Using the Lock Profiler

To enable or disable lock-statistics gathering, follow these steps:

1. Expand the **Diagnostics** node, then click the **Lock profiler** node.
2. Click **On** to enable statistics gathering.
   Click **Off** to disable statistics gathering.
3. Specify the **Trace depth** (lock code-path trace depth) to set the number of client call-stack frames to analyze.
   See Trace Depth for more information.
4. Click **Clients** to view lock statistics for Terracotta clients, or click **Servers** to view lock statistics for Terracotta servers.
   Client-view lock statistics are based on the code paths in the clustered application that result in a lock being taken out. Server-view lock statistics concern the cluster-wide nature of the distributed locks. See Lock Element Detail for more information.
5. Click **Refresh** to display the latest statistics.

Gathering and recording statistics can impact a cluster's performance. If statistics are being gathered, you are alerted in the cluster list by a flashing icon next to the affected cluster.

Lock Names

Each lock has a corresponding identifier, the lock-id. For a named lock the lock-id is the lock name. For an autolock the lock-id is the server-generated id of the object on which that lock was taken out. An example of an autolock id is @1001. That autolock id corresponds to the shared object upon which distributed synchronization was carried out. You can use the object browser to view the state of shared object @1001.

Searching for Specific Locks

You can search for specific locks listed in the **Lock** column. Enter a string in **Find**, then click **Next** or **Previous** to move through matching entries.

Trace Depth

A single lock-expression in the configuration can result in the creation of multiple locks by the use of wildcard patterns. A single lock can be arrived at through any number of different code paths. For example, there could be 3 different call sequences that result in a particular lock being granted, with one of the paths rarely entered and another responsible for the majority of those lock grants. By setting the trace depth appropriately you can gain insight into the behavior of your application and how it can affect the performance of your clustered system.

The trace depth control sets the number of client call-stack frames that are analyzed per lock event to record lock statistics. A depth of 0 gathers lock statistics without regard to how the lock event was arrived at. A lock depth of 1 means that one call-stack frame will be used to disambiguate different code paths when the lock event occurred. A lock depth of 2 will use two frames, and so on.

Trace stack frames can include Java source line numbers if the code is compiled with debugging enabled. This can be done by passing the -g flag to the javac command, or in Ant by defining the javac task with the debug="true" attribute.

With a trace-depth setting of 1 all locks are recorded together, regardless of the call path. This is because the stack depth analyzed will always be just the method that resulted in the lock event (in other words the surrounding method). For example, a lock event that occurs within method Foo() records all lock events occurring within Foo() as one single statistic.

With a lock depth of 2, different call paths can be separated because both the surrounding method and the calling method are used to record different lock statistics. For example, the callers of Foo(), Bar1() and Bar2(), are also considered. A call path of Bar1() -> Foo() is recorded separately from Bar2() -> Foo().

The Statistics Table

Lock Statistic	Description
Times Requested	Number of times this lock was requested by clients in the cluster.
Times Hopped	Times an acquired greedy lock was retracted from a holding client and granted to another client.
Average Contenders	Average number of threads wishing to acquire the lock at the time it was requested.
Average Acquire Time	Average time (in milliseconds) between lock request and grant.
Average Held Time	Average time (in milliseconds) grantee held this lock.
Average Nested	Average number of outstanding locks held by acquiring thread at grant time.

Greedy Locks

Terracotta employs the concept of *greedy locks* to improve performance by limiting unnecessary lock hops. Once a client has been awarded a lock, it is allowed to keep that lock until another client requests it. The assumption is that once a client obtains a lock it is likely to request that same lock again. For example, in a cluster with a single node repeatedly manipulating a single shared object, server lock requests should be 1

until another client enters the cluster and begins manipulating that same object. Server statistics showing "na" (undefined) are likely due to greedy locks.

Lock Element Details

The bottom portion of the client's view displays details on the selected lock element. The currently selected lock trace is shown on the left and the configuration element responsible for the creation of the selected lock is shown on the right.

Real-Time Performance Monitoring

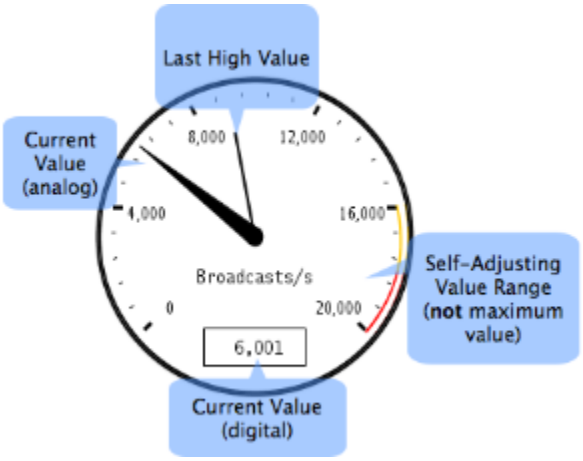
Real-time cluster monitoring allows you to spot issues as they develop in the cluster.

Dashboard

The cluster activity gauges provide real-time readings of crucial cluster metrics.

Each gauge has the following characteristics:

- Yellow and red zones on the dial indicate when the metric value has reached warning or extreme levels.
- A digital readout field displays the metric's current value.
- A tooltip shows the metric's full name, last maximum value, and average value (over all samples).
- By default, values are sampled over one-second intervals (except for **Unacked Txns**). The sample rate can be changed in the Options Dialog.
- A "high-water" mark tracks the last high value, fading after several seconds.
- A self-adjusting value range uses a built-in multiplier to automatically scale with the cluster.



The left-most gauge (the large dial with the red needle) measures the rate of write transactions, which reflects the work being done in the cluster, based on Terracotta transactions. This gauge may have a high value or trend higher in a busy cluster. An indication that the cluster may be overloaded or out of tune is when this gauge is constantly at the top of its range.

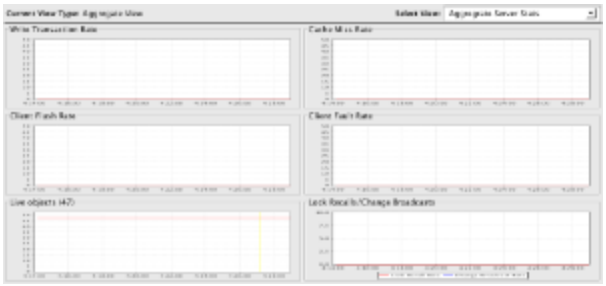
The remaining gauges, which measure "impeding factors" in your cluster, typically fluctuate or remain steady at a low value. If any impeding factors consistently trend higher over time, or remain at a high value, a problem may exist in the cluster. These gauges are listed below:

- **Objects Created/s** – The rate of shared objects being created. A rising trend can have a negative impact on performance by reducing available memory and necessitating more garbage collection.
- **Lock Recalls/s** – The number of locks being recalled by Terracotta servers. Growing lock recalls result from a high contention for shared objects, and have a negative performance impact. Higher locality of reference can usually lower the rate of lock recalls.
- **Broadcasts/s** – The number of object changes being communicated by the server to affected clients. High broadcast rates raise network traffic and can have a negative performance impact. Higher locality of reference can usually lower the need for broadcasts.
- **Faults/s** – Rate of faulting objects from servers to all connected clients. A high or increasing value can indicate one or more clients running low on memory or poor locality of reference.
- **Flushes/s** – Rate of flushing objects from all connected clients to servers. A high or increasing value can indicate one or more clients running low on memory.
- **Transaction Size KB/s** – Average size of total transactions.
- **Unacked Txns** – The current count of unacknowledged client transactions. A high or increasing value can indicate one more troubled clients.

Runtime Statistics

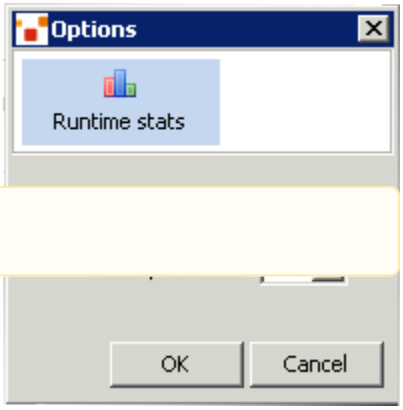
Runtime statistics provide a continuous feed of sampled real-time data on a number of server and client metrics. The data is plotted on a graph with configurable polling and historical periods. Sampling begins automatically when a runtime statistic panel is first viewed, but historical data is not saved. To record and save historical data, see the Cluster Statistics Recorder.

To view runtime statistics for a cluster, expand the cluster's **Diagnositics** node, then click the **Runtime statistics** node.



Use the **Select View** menu to set the runtime statistics view to one of the following:

- Aggregate View – Choose **Aggregate Server Stats** to display cluster-wide statistics.
- Per-Client View – Choose a client to display runtime statistics for that client.
- Per-Server View – Choose a server to display runtime statistics for that server.



To adjust the poll and history periods, choose **Options** from the **Tools** menu. In the **Options** dialog, adjust the values in the polling and history fields. These values apply to all runtime-statistics views.

Specific runtime statistics are defined in the following sections. The cluster components for which the statistic is available are indicated in parentheses.

> If "Cluster" is indicated as a cluster component, it indicates the aggregate for all servers in the clusters.

Heap Usage (Server, Client)

Shows the amount, in megabytes, of maximum available heap and heap being used.

Host CPU Usage (Server, Client)

Shows the CPU load as a percentage. If more than one CPU is being used, each CPU's load is shown as a separate graph line.

Write Transaction Rate (Cluster, Server, Client)

Shows the number of completed Terracotta transactions. Terracotta transactions are sets of one or more clustered object changes, or writes, that must be applied atomically.

> Some statistics available through the Terracotta Developer Console are about Terracotta transactions. Terracotta transactions are not application transactions. One Terracotta transaction is a batch of one or more writes to shared data. See the Concept and Architecture Guide for more information.

Cache Miss Rate (Cluster, Server)

Disk-to-server faults occur when an object is not available in a server's in-memory cache. The Cache Miss Rate statistic is a measure of how many objects (per-second) are being faulted from the disk in response to client requests. Objects being requested for the first time, or objects that have been flushed from the server heap before a request arrives, must be faulted in from disk. A high Cache Miss Rate could indicate inadequate memory allocation at the server.

Unacknowledged Transaction Broadcasts (Client)

Every Terracotta transaction in a Terracotta cluster must be acknowledged by Terracotta clients with in-memory shared objects that are affected by that transaction. For each client, Terracotta server instances keep a count of transactions that have not been acknowledged by that client. The Unacknowledged Transaction Broadcasts statistic is a count of how many transactions the client has yet to acknowledge. An upward trend in this statistic indicates that a client is not keeping up with transaction acknowledgments, which can slow the entire cluster. Such a client may need to be disconnected.

Client Flush Rate (Cluster, Server, Client)

The Client Flush Rate statistic is a measure of how many objects are being flushed out of client memory to the Terracotta server. These objects are available in the Terracotta server if needed at a later point in time. A high flush rate could indicate inadequate memory allocation at the client.

On a server, the Client Flush Rate is a total including all clients. On a client, the Client Flush Rate is a total of the objects that client is flushing.

Client Fault Rate (Cluster, Server, Client)

The Client Fault Rate statistic is a measure of how many objects are being faulted into client memory from the server. A high fault rate could indicate poor locality of reference or inadequate memory allocation at the client.

On a server, the Client Fault Rate is a total including all clients. On a client, the Client Fault Rate is a total of the objects that have been faulted to that client.

> When the Terracotta server faults an object, it also faults metadata for constructing a certain number of the objects referenced by or related to that object. This improves locality of reference. See the definition of the fault-count property in Terracotta Configuration Guide and Reference for more information.

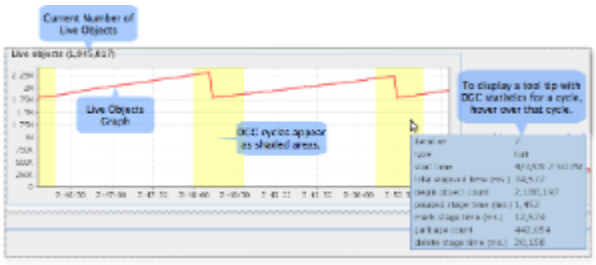
Lock Recalls / Change Broadcasts (Cluster)

Terracotta servers recall a lock from one client as a response to lock requests from other clients. An upward trend in lock recalls could indicated poor locality of reference.

Change broadcasts tracks the number of object-change notifications that Terracotta servers are sending. See l2 changes per broadcast, l2 broadcast count , and l2 broadcast per transaction for more information on broadcasts.

Live Objects (Cluster)

If the trend for the total number of live objects goes up continuously, clients in the cluster will eventually run out of memory and applications may fail. Upward trends indicate a problem with application logic, garbage collection, or a tuning issue on one or more clients. The total number of live objects is given in the graph's title.



Distributed Garbage Collection

Objects in a DSO root object graph can become unreferenced and no longer exist in the Terracotta client's heap. These objects are eventually marked as garbage in a Terracotta server instance's heap and from persistent storage by the Terracotta Distributed Garbage Collector (DGC). The DGC is unrelated to the Java garbage collector.

> For more information on the DGC, see the Terracotta Concept and Architecture Guide.

To view a history table of DGC activity in the current cluster, expand the cluster's **Cluster heap** node, then click the **Garbage collection** node. The history table is automatically refreshed each time a collection occurs. Each row in the history table represents one distributed garbage collection cycle, with the following columns:

Column	Definition	Values
Iteration	The index number of the DGC cycle	Sequential integer
Type	The type of cycle	Full – Running a full collection cycle targeting all eligible objects. Young – Running a collection cycle targeting Young Generation objects.
Status	The collection cycle's current state	START – Monitoring for object reference changes and collecting statistics such as the object begin count. MARK – Determining which objects should be collected and which should not. PAUSE – Determining if any marked objects should not be collected. MARK COMPLETED – Stops checking for reference changes (finalizing marked object list). DELETE – Deleting objects. COMPLETE – Completed cycle.
Start time	The date and time the cycle began	Date and time stamp (local server time)
Begin count	The total number of shared objects held by the server	Integer counter
Paused stage	The total time the DGC paused	Milliseconds
Mark stage	The total time the DGC took to mark objects for collection	Milliseconds
	The number of shared objects marked for collection	Integer counter

Garbage count		
Delete stage	The total time the DGC took to collect marked objects	Milliseconds
Total elapsed time	The total time the DGC took to pause, mark objects, and collect marked objects	Milliseconds

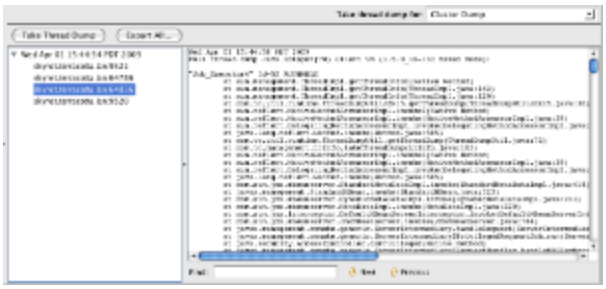
The DGC graph combines a real-time line graph (with history) displaying the DGC total elapsed time with a bar graph showing the total number of freed objects.

Triggering a DGC Cycle

The DGC panel displays a message stating the configured frequency of DGC cycles. To manually trigger a DGC cycle, click **Run DGC**.

Thread Dumps

You can get a snapshot of the state of each server and client in the Terracotta cluster using thread dumps. To display the console's thread-dumps feature, expand the **Diagnostics** node, then click the **Cluster thread dumps** node.



The thread-dump navigation pane lists completed thread dumps by date-time stamp. The contents of selected thread dumps are displayed in the right-side pane. A **Find** tool is available for searching through the currently displayed thread dump.

To delete a thread dump (or all thread dumps) from the thread-dump navigation pane, right-click the thread dump in the thread-dumps navigation pane, then choose **Delete** (or **Delete All**) from the context menu. You can also delete a selected thread dump with your keyboard's Delete key.

Servers that appear in the console but are not running produce empty thread dumps.

Taking a Thread Dump

To generate a thread dump, follow these steps:

1. Choose the target of the thread dump from the **Take thread dump for** menu.
   To take thread dumps for all of the cluster's components, choose **Cluster Dump**.
2. Click **Take Thread Dump**.
   When complete, the thread dump appears in the thread-dumps navigation pane .
3. Expand the entry created in the thread-dumps navigation pane to see the thread-dump entry or entries.
   The entries correspond to servers and clients included in the thread dump.
4. Click a server or client entry to display the thread dump for that server or client.

Saving Thread Dumps

Thread dumps are deleted each time the console is restarted.

To save all existing thread dumps to disk as a ZIP file, click **Export All...**.

To save any thread dump as a text file, right-click the thread dump in the thread-dumps navigation pane, then choose **Export As Text...** from the context menu.

Logs and Status Messages

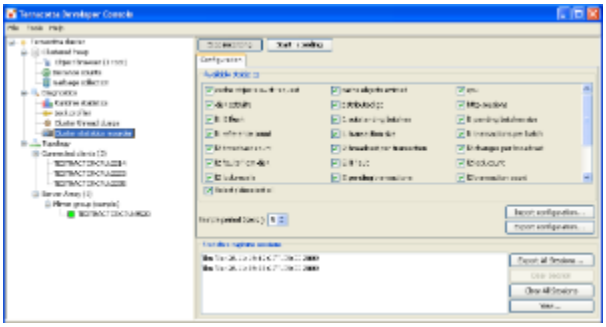
Click the **Logs** tab in the Status Panel to display log messages for any of the servers in the cluster. From the **View log for** menu, choose the server whose logs you want to view.

The status bar at the bottom of the console window displays messages on the latest changes in the cluster, such as nodes joining or leaving.

Recording and Viewing Statistics

Cluster Statistics Recorder

The **Cluster Statistics Recorder** panel can generate recordings of selected cluster-wide statistics. This panel has controls to start, stop, view, and export recording sessions. You can use the Snapshot Visualization Tool to view the recorded information.

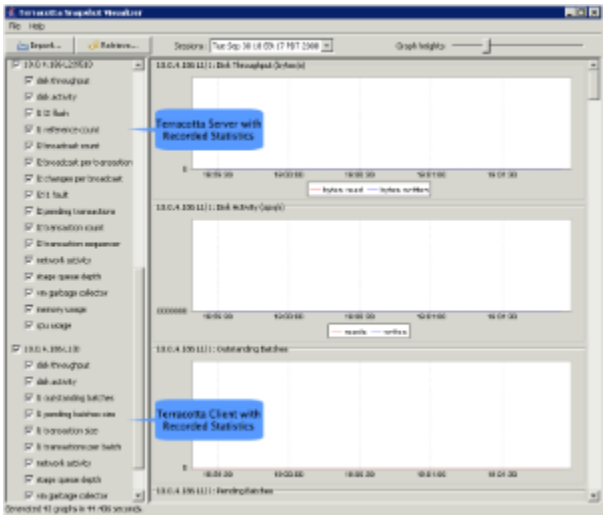


For definitions of available statistics, see Cluster Statistics Definitions. To learn about configuring the Terracotta Cluster Statistics Recorder, using its command-line interface, and more, see the Platform Statistics Recorder Guide.

> Gathering and recording statistics can impact a cluster's performance. If statistics are being gathered, you are alerted in the cluster list by a flashing icon next to the affected cluster.

">Snapshot Visualization Tool

The **Snapshot Visualization Tool** (SVT) provides a graphical view of cluster information and statistics. The view is created using data recorded with the Statistics Recorder.



The SVT is provided as a TIM, called tim-svt, which you can install from the TIM Update Center. Once the SVT is installed, start (or restart) the Terracotta Developer Console and confirm that the **View** button on the Cluster Statistics Recorder is enabled. No changes to the Terracotta configuration file are necessary when you install the SVT.

> In Eclipse with the Terracotta DSO Eclipse plug-in, use the **Terracotta|Update modules...** menu to install SVT.

SVT controls are defined below.

Import...

Load a saved cluster statistics recording that was saved to file. Clicking **Import...** opens a standard file-selection window for locating the file.

Retrieve...

Find recorded sessions on active Terracotta servers. Clicking **Retrieve...** opens a dialog to enter the server address in the format <server_ip_address: JMX_port> or <server_name:JMX_port> as defined in `tc-config.xml`. Once connected to the server, available recorded sessions appear in the SVT **Sessions** menu.

Sessions

Select a recorded session from the **Sessions** drop-down menu to load it. **Sessions** lists the recorded sessions last retrieved from a Terracotta server.

Graph heights

Scale the y-axis on the graphs displayed in the SVT. Moving the slider to the left shrinks the graph height, while moving it to the right grows the graph height.

Server, Client, and Statistics checkboxes

Choose which servers and clients have their recorded statistics displayed in the graphs. For each server and client, choose which statistics are displayed in the graphs.

Generate graphs

Click **Generate graphs** to create the graphs from the selected statistics.

Terracotta for Hibernate

If you are using the Terracotta for Hibernate with your Hibernate-based application, the Hibernate and second-level cache views offer you the following:

- Deep visibility into Hibernate and cached data
- Live statistics
- Graphs, including puts and misses
- Historical data for trend analysis
- Parameters for control over the size and efficiency of cache regions
- A configuration generator

To access the Hibernate and second-Level cache views, expand the **My application** node in the cluster navigation pane, then click the **Hibernate** node.

Each time you connect to the Terracotta cluster with the Developer Console, Hibernate and cache statistics gathering is automatically started. Since this may have a negative impact on performance, consider disabling statistics gathering during performance tests and in production. To disable statistics gathering, navigate to the **Overview** panel in the **Second-Level Cache** view, then click **Disable Statistics**.

Use the view buttons to choose **Hibernate** (Hibernate statistics) or **Second-Level Cache** (second-level cache statistics and controls).

Tips for Working with the User Interface

- The fully qualified name of a region, entity, or collection shown in a table may be abbreviated. You can view the unabbreviated name in a tooltip by placing the mouse pointer over the abbreviated name. Note that expanding the width of a column does not undo abbreviations.
- Queries are not abbreviated, but can still appear to be cut off by columns that are too narrow. To view the full query string, you can expand the column or view the full query string in a tooltip by placing the mouse pointer over the cut-off query string.
- You can view the total sum of a column of numbers (such as Hibernate Inserts) in a tooltip by placing the mouse pointer anywhere in the column.
- To order a table along the values of any column, double-click its heading. An arrow appears in the column heading to indicate the direction of the order. You can reverse the order by double-clicking the column head again.
- Some panels have a **Clear All Statistics** button. Clicking this button clears statistics from the current panel and all other Hibernate and cache panels that display statistics.
- If your cluster has more than one second-level cache, use the **Persistence Unit** drop-down menu (available in all panels) to choose the cache you want to view.

Hibernate View

Click **Hibernate** view to display a table of Hibernate statistics. Some of the main tasks you can perform in this view are:

- View statistics for the entire cluster on Hibernate entities, collections, and queries.
- View statistics for each Terracotta client (application server) on Hibernate entities, collections, and queries.

Hibernate statistics are sampled at the rate determined by the rate sent in the Options dialog. Use the controls that appear below the statistics table to update the statistics:

- Refresh – Click **Refresh** to force the console to immediately poll for statistics.
- Clear – Click **Clear All Statistics** to wipe the current display of statistics.

You can set the scope of the statistics display using the **Select View** menu. To view statistics for the entire cluster, select **Cluster stats**. To view statistics for an individual Terracotta client (application server), choose that client from the **Per Client View** submenu.

Entities

Click **Entities** to view the following standard Hibernate statistics on Hibernate entities in your application:

- Name
- Loads
- Updates
- Inserts
- Deletes
- Fetches
- Optimistic Failures

Collections

Click **Collections** to view the following standard Hibernate statistics on Hibernate collections in your application:

- Role
- Loads
- Fetches
- Updates
- Removes
- Recreates

Queries

Click **Queries** to view the following standard Hibernate statistics on Hibernate queries in your application:

- Query
- Executions
- Rows
- Avg Time (Average Time)
- Max Time (Maximum Time)
- Min Time (Minimum Time)

Second-Level Cache View

The **Second-Level Cache** view provides performance statistics and includes per-region cache configuration. Some of the main tasks you can perform in this view are:

- View both live and historical performance metrics in graphs.
- Enable and disable cache regions.
- Flush cache regions.
- Set eviction parameters per region.
- Generate a configuration file based on settings in the Second-Level Cache view.

Overview

The **Overview** panel displays the following real-time performance statistics for both Global Cache Performance (covering the entire cache) and Per-Region Cache Performance:

- Hits – (Green) Counts queries that return data from the second-level cache.
- Puts – (Blue) Counts each new (or updated) element added to the cache.
- Misses – (Red) Counts each cache miss; each miss causes data fault-in from the database.

The number of puts can be greater than the number of misses because updates are counted as puts.

These statistics are displayed as color-coded bar graphs with current values shown on the left end and "high-water" (current maximum) values shown on the right end.

The **Overview** panel provides the following controls:

- Enable All Cache Regions – Turns on all configured cache regions.
- Disable All Cache Regions – Turns off all configured cache regions shown.
- Evict All Entries – Removes all entries from the cache (clears the cache).
- Disable Statistics – Turns off the gathering of statistics. Only **DB SQL Execution Rate** (queries per second), **In-Memory Count**, and **Total Count** of cache elements continue to function. When statistics gathering is off, this button is called **Enable Statistics**. *Gathering statistics may have a negative impact on performance.*
- Clear All Statistics – Restart the recording of statistics (zero out all values).

Statistics

The **Statistics** panel displays second-level cache statistics in history graphs and a table. The graphs are useful for recognizing trends, while the table presents a snapshot of statistics.

Some of the main tasks you can perform in this panel are:

- View cache statistics for the entire cluster.
- View cache statistics for each Terracotta client (application server).

Cache statistics are sampled at the rate determined by the rate sent in the Options dialog. Use the controls that appear below the statistics table to update the statistics:

- Refresh – Click **Refresh** to force the console to immediately poll for statistics.
- Clear – Click **Clear All Statistics** to wipe the current display of statistics.

You can set the scope of the statistics display using the **Select View** menu. To view statistics for the entire cluster, select **Cluster stats**. To view statistics for an individual Terracotta client (application server), choose that client from the **Per Client View** submenu.

Cache Statistics Graphs

The line graphs available display the following statistics over time:

Cache Hit Ratio – The ratio of cache hits to queries. A ratio of 1.00 means that all queried data was obtained from the cache. A low ratio (closer to 0.00) implies a higher number of misses that result in more faulting of data from the database.
Cache Hit/Miss Rate – The number of cache hits per second (green) and the number of cache misses per second (red). Current values are overlaid on the graph.
DB SQL Execution Rate – The number of queries executed per second. The current value is overlaid on the graph.
Cache Put Rate – The number of cache puts executed per second. The number of puts always equals or exceeds the number of misses, since every miss leads to a put. The current value is overlaid on the graph.

Cache Statistics Table

The cache statistics table displays a snapshot of the following statistics for each region:

- Region – The fully qualified name of the region (abbreviated).
- Hit Ratio – The aggregate ration of hits to queries.
- Hits – The total number of successful queries on the cache.
- Misses – The total number of unsuccessful queries on the cache.
- Puts – The total number of new (or updated) entities added to the cache.
- In-Memory Count – The total number of cache elements in the region on the client selected in **Select View**. This statistic is not available in the cluster-wide view.
- Total Count – The total number of cache elements in the region. Even when a client is selected in **Select View**, this statistic always displays the cluster-wide total.
- Hit Latency – The time (in milliseconds) it takes to find an entity in the cache. Long latency times may indicate that the entity is not available locally and is being faulted from the Terracotta server.
- Load Latency – The time (in milliseconds) it takes to load an entity from the database after a cache miss.

The snapshot is refreshed each time you display the **Statistics** panel. To manually refresh the table, click **Refresh**.

Configuration

The **Configuration** panel displays a table with the following configuration properties for each cache region:

- Region – The fully qualified name of the region (abbreviated).
- Cached – Whether the region is currently being cached ("On") or not ("Off").
- TTI (Time to idle) – The maximum number of seconds an element can exist in the cache without being accessed. The element expires at this limit and will no longer be returned from the cache. 0 means no TTI eviction takes place (infinite lifetime).
- TTL (Time to live) – The maximum number of seconds an element can exist in the cache regardless of use. The element expires at this limit and will no longer be returned from the cache. 0 means no TTL eviction takes place (infinite lifetime).
- Target Max In-Memory Count – The maximum number of elements allowed in a region in any one client (any one application server). If this target is exceeded, eviction occurs to bring the count within the allowed target. 0 means no eviction takes place (infinite size is allowed).
- Target Max Total Count – The maximum total number of elements allowed for a region in all clients (all application servers). If this target is exceeded, eviction occurs to bring the count within the allowed target. 0 means no eviction takes place (infinite size is allowed).

Having the values of TTI, TTL, Target Max In-Memory Count, and Target Max Total Count all set to 0 for a region in effect *turns off* all eviction of cache elements for that region. Unless you want cache elements to *never* be evicted from a region, you should set these properties to non-zero values that are optimal for your use case.

Configuration is initially loaded from `tc-hibernate-cache.xml` or a built-in default configuration if this file is not available.

`tc-hibernate-cache.xml` must be in your application's class path to be found by the second-level cache. For web applications, it is suggested that tc-hibernate-cache.xml be saved to WEB-INF/classes.

Region Operations

To stop a region from being cached, select that region in the configuration table, then click **Disable Region**. Disabled regions display "Off" in the configuration table's **Cached** column. Queries for elements that would be cached in the region must go to the database to return the desired data.

To return a region to being cached, select that region in the configuration table, then click **Enable Region**. Disabled regions display "On" in the configuration table's **Cached** column.

To clear a region, select that region in the configuration table, then click **Evict All Entries in Cache**. This operation removes from all clients all of the entries that were cached in that region.

If you are troubleshooting or otherwise require more detailed visibility into the workings of the second-level cache, enable **Logging enabled**.

Region Settings

To change the configuration for a region, select that region in the configuration table, then change the values in the fields provided:

- Time to idle
- Time to live
- Target max total count
- Target max in-memory count

You can also turn logging on for the region by selecting **Logging enabled**.

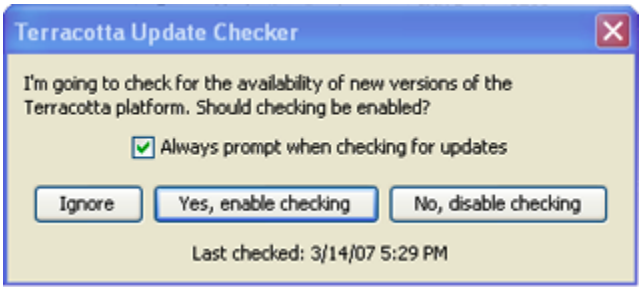
Note that the settings you change in the Second-Level Cache view are not saved to the cache configuration file and are not persisted beyond the lifetime of the cache. To create a configuration file based on the configuration shown in the Second-Level Cache view, click **Generate Cache Configuration...** to open a window containing a complete configuration file. Copy this configuration and paste it into the `tc-hibernate-cache.xml` to be used for configuring the second-level cache. For more information on the Terracotta for Hibernate configuration file, see the Terracotta for Hibernate Reference document in .

Backing Up Shared Data

The Terracotta Operations Center provides console-based backup configuration. Enterprise versions of Terracotta also include a [backup script].

Update Checker

On a bi-weekly basis the Terracotta Developer Console will check, when first started, on updates to the Terracotta platform. By default, a notice informing you that an update check is about to be performed is displayed, allowing to ignore the immediate check, acknowledge and allow the check, or to disable further checking.



Should the update check be allowed, the Terracotta Developer Console will query the OpenTerracotta website (www.terracotta.org) and report on any new updates. Should the update checker feature be disabled, it can always be re-enabled via the **Help|Update Checker...** menu.

More Information

More information on the following topics is available in other Terracotta documentation:

- JMX Port
- Roots
- Distributed Garbage Collection
- Object Portability
- Testing and Tuning Terracotta for Hibernate in

Appendix: Definitions of Cluster Statistics

The following categories of cluster information and statistics are available for viewing and recording in the Terracotta Cluster Statistics Recorder.

> l1 = Terracotta client
> l2 = Terracotta server instance
> See the Glossary for definitions of commonly used terms.

cache objects evict request

The total number of objects marked for eviction from the l1, or from the l2 to disk. Evicted objects are still referenced, and can be faulted back to the l1 from the l2 or from disk to l2. The SVT graphs this metric for each l1 and l2 separately. High counts imply that free memory could be low.

cache objects evicted

The number of objects actually evicted. If this metric is not close in value to *cache objects evict request*, then memory may not be getting freed quickly enough.

l1 l2 flush

The object flush rate when the l1 flushes objects to the l2 to free up memory or as a result of GC activity.

l2 faults from disk

The number of times the l2 has to load objects from disk to serve l1 object demand. A high faulting rate could indicate an overburdened l2.

l2 l1 fault

The number of times an l2 has to send objects to an l1 because the objects do not exist in the l1 local heap due to memory constraints. Better scalability is achieved when this number is lowered through improved locality and usage of an optimal number of JVMs.

memory (usage)

The amount of memory (heap) usage over time.

vm garbage collector

The standard Java garbage collector's behavior, tracked on all JVMs in the cluster.

distributed gc (distributed garbage collection, or DGC)

The behavior of the Terracotta tool that collects distributed garbage on an l2. The DGC can pause all other work on the l2 to ensure that no referenced objects are flagged for garbage collection.

l2 pending transactions

The number of Terracotta transactions held in memory by a Terracotta server instance for the purpose of minimizing disk writes. Before writing the pending transactions, the Terracotta server instance optimizes them by folding in redundant changes, thus reducing its disk access time. Any object that is part of a pending transaction cannot be changed until the transaction is complete.

stage queue depth

The depth to which visibility into Terracotta server-instance work-queues is available. A larger depth value allows more detail to emerge on pending task-completion processes, bottlenecks due to application requests or behavior, types of work being done, and load conditions. Rising counts (of items in these processing queues) indicate backlogs and could indicate performance degradation.

server transaction sequencer stats

Statistics on the Terracotta server-instance transaction sequencer, which sequences transactions as resources become available while maintaining transaction order.

network activity

The amount of data transmitted and received by a Terracotta server instance in bytes per second.

l2 changes per broadcast

The number of updates to objects on disk per broadcast message (see l2 broadcast count).

message monitor

The network message count flowing over TCP from Terracotta clients to the Terracotta server.

l2 broadcast count

The number of times that a Terracotta server instance has transmitted changes to objects. This "broadcast" occurs any time the changed object is resident in more than one Terracotta client JVM. This is not a true broadcast since messages are sent only to clients where the changed objects are resident.

l2 transaction count

The number of Terracotta transactions being processed (per second) by a Terracotta server instance.

l2 broadcast per transaction

The ratio of broadcasts to Terracotta transactions. A high ratio (close to 1) means that each broadcast is reporting few transactions, and implies a high co-residency of objects and inefficient distribution of application data. A low ratio (close to 0) reflects high locality of reference and better options for linear scalability.

system properties

Snapshot of all Java properties passed in and set at startup for each JVM. Used to determine configuration states at the time of data capture, and for comparison of configuration across JVMs.

thread dump

Displays a marker on all statistics graphs in the SVT at the point when a thread dump was taken using the *Cluster statistics recorder*. Clicking the marker displays the thread dump.

disk activity

The number of operations (reads and writes) per second, and the number of bytes per second (throughput). The number of reads corresponds to l2 faulting objects from disk, while writes corresponds to l2 flushing objects to disk. The SVT graphs the two aspects separately.

cpu (usage)

The percent of CPU resources being used. Dual-core processors are broken out into CPU0 and CPU1.