Firebase JMX

Firebase Version: 1.5

Introduction

This document details the public management (JMX) interfaces for a Firebase server. The JMX standard defines remote interfaces (MBeans) for enterprise Java applications and can be used for application monitoring and control.

Each Java distribution comes with JConsole, an application for managing a remote Java VM, including its MBeans.

Firebase uses JMX extensively for monitoring and diagnostics. This document contains the MBeans available in the “firebase-jmx” module. All MBeans are also documented using JavaDoc.

Client Registry

Invocation Target

Object Name: com.cubeia.firebase.clients:type=ClientRegistry
Interface: com.cubeia.firebase.service.clientreg.state.StateClientRegistryMBean

Methods

Get Number Of Clients

Signature:

```
public int getNumberOfClients();
```

Returns the number of clients on the local server.

Kick Player

Signature:

```
public void kickPlayer(int playerId);
```

Kick the client with the given player id from the system. The client will be removed from all tables and the socket will be closed. The client will receive a Forced Logout Packet.

Note: This method only applies to the local client registry. This means that you need to invoke this method on the server where the client is logged in.

Kick Player From Table

Signature:

```
public void kickPlayerFromTable(int playerId, int tableId);
```

Kick a player from a table. The player will be kicked from the table only. The client will not be logged out or disconnected from the system.

Note: The player will be removed from the table regardless of where the session is located. However, the Kick Player Packet, which is a notification of the kick, will only be sent to client if the session in the same server as the invocation target.

Get Remote Address

Signature:

```
public String getRemoteAddressText(int clientId);
```

Returns the client's remote address in String format (SocketAddress.toString).

The method will return null if the client is not logged in (I.e. Not found in the client registry).

Note: This method only applies to the local client registry. This means that you need to invoke this method on the server where the client is logged in. If the client is not logged in locally a null pointer exception will be thrown.

Is Logged In

Signature:

```
public boolean isLoggedIn(int clientId);
```

Returns true if the client with the given id has a session in on the system, this includes clients that are connected and have lost connections but are not yet reaped.

This lookup is system wide.

Get Seated Tables

Signature:

```
public Map<Integer, Integer> getSeatedTables(int playerId);
```

Returns the tables and seats the player is seated at.
The map will consist of `<tableid:seat>`

Works system wide.

Get Watching Tables

Signature:

```
public List<Integer> getWatchingTables(int playerId);
```

Returns a list of tables that the player is watching. The list will not contain tables the player is seated at, but only tables where the player is registered as a watcher.

Works system wide.

Get Screenname

Signature:

```
public String getScreenname(int clientId);
```

Returns the screen name of the client with the given client id or null if not found. The client must be logged in, if the client is not logged in then null will be returned.

Works system wide.

Is Local

Signature:

```
public boolean isLocal(int clientId);
```

Returns true if the client is managed by this local node. Returns false otherwise. The check includes client sessions currently connected and client sessions waiting for reconnect. (See is logged in).

The method only checks the local node so it is not a system wide call.

Send System Message (broadcast)

Signature:

```
public void sendSystemMessage(int type, int level, String message);
```

Send a message to all logged in players. The type and level integers are mapped directly against the actual protocol packet variables with the same name, ie. are implementation specific.

Send System Message

Signature:

```
public void sendSystemMessage(int type, int level, int[] playerIds,
    String message);
```

Send a message to a subset of the logged in players. If a player identified in the player id array is not logged in the message will be dropped. The type and level integers are mapped directly against the actual protocol packet variables with the same name, ie. are implementation specific.

Get All Logged In

Signature:

```
public int[] getAllLoggedIn();
```

Get a list of all clients logged in to the system (client ids). This returns all players from all nodes, and not just locally logged in players.

Get Client Status Ordinal

Signature:

```
public int getClientStatusOrdinal(int clientId);
```

Get the state of a client. The returned int will be the ordinal of the ClientSessionState enumeration.

Get Client Status String

Signature:

```
public String getClientStatusString(int clientId);
```

Get the state of a client as a readable string representation.

Server Instance

Invocation Target

Object Name: com.cubeia.firebase:type=ServerInstance
Interface: com.cubeia.firebase.server.jmx.LocalServerMBean

Methods

Add Node

Signature:

```
public void addNode(String nodeType, String id);
```

Adds a node to the current server. The node type must be one of “mtt”, “game”, or “client” and the node id must be unique across the cluster. You cannot have multiple nodes of the same type on a single server.

Remove Node

Signature:

```
public void removeNode(String id);
```

Stop and destroy a node on the local server.

Get Live Nodes

Signature:

```
public String[] getLiveNodes();
```

This method returns the currently running nodes of the server in a '<type>:<id>' string format.

Game Node

Invocation Target

Object Name: com.cubeia.firebase:type=GameNode
Interface: com.cubeia.firebase.server.game.GameNodeMBean

Methods

Is Halted

Signature:

```
public boolean isHalted();
```

Check if the game node is halted or not.

Resume

Signature:

```
public void resume();
```

Forcibly resume the game node. This should only be used as a last resort if the system is in an inconsistent state.

Halt

Signature:

```
public void halt();
```

Forcibly halt the game node. This should only be used as a last resort if the system is in an inconsistent state.

Is Coordinator

Signature:

```
public boolean isCoordinator();
```

This method returns true if the node is the Firebase coordinator for the type. There is only ever one coordinator per type within a cluster.

Get Node Id

Signature:

```
public String getNodeId();
```

Get the node id. The id is used within the cluster for communication and identification. If not set at startup it will be auto-generated.

Client Node

Invocation Target

Object Name: com.cubeia.firebase:type=ClientNode
Interface: com.cubeia.firebase.server.gateway.GatewayNodeMBean

Methods

Is Coordinator

Signature:

```
public boolean isCoordinator();
```

This method returns true if the node is the Firebase coordinator for the type. There is only ever one coordinator per type within a cluster.

Get Node Id

Signature:

```
public String getNodeId();
```

Get the node id. The id is used within the cluster for communication and identification. If not set at startup it will be auto-generated.

Master Node

Invocation Target

Object Name: com.cubeia.firebase:type=MasterNode
Interface: com.cubeia.firebase.server.master.MasterNodeMBean

Methods

Is Primary

Signature:

```
public boolean isPrimary();
```

This method returns true if the node is the Firebase primary master. There is only ever primary master per type within a cluster.

Get Node Id

Signature:

```
public String getNodeId();
```

Get the node id. The id is used within the cluster for communication and identification. If not set at startup it will be auto-generated.

Get Local Address

Signature:

```
public String getLocalAddress();
```

Returns the local socket address which is used for internal communications.

Tournament Node

Invocation Target

Object Name: com.cubeia.firebase:type=MttNode
Interface: com.cubeia.firebase.server.mtt.MttNodeMBean

Methods

Is Coordinator

Signature:

```
public boolean isCoordinator();
```

This method returns true if the node is the Firebase coordinator for the type. There is only ever one coordinator per type within a cluster.

Get Node Id

Signature:

```
public String getNodeId();
```

Get the node id. The id is used within the cluster for communication and identification. If not set at startup it will be auto-generated.

Cluster Node Registry

Invocation Target

Object Name: com.cubeia.firebase:type=ClusterNodeRegistry
Interface: com.cubeia.firebase.server.master.ClusterNodeRegistryMBean

Methods

Get Registry Size

Signature:

```
public int getRegistrySize();
```

Count the number of active participants in the cluster.

Print All Participants

Signature:

```
public String[] printAllParticipants();
```

Get all participants in a string form. Each string in the array represents a single node. The string format is:

{id: <node-id>[<server-id>[<comm-ip>:<comm-port>]]; role: <role>; isCoordinator: <true | false>}

Service Registry

Invocation Target

Object Name: com.cubeia.firebase:type=ServiceRegistry
Interface: com.cubeia.firebase.server.service.jmx.ServiceRegistryMBean

Methods

Get Service Count

Signature:

```
public int getServiceCount();
```

Count the number of services within the server.

Get Services

Signature:

```
public TabularData getServices();
```

This method returns all services in the server as composite types. The composite type is described in the ServiceBeanType class.

System Cache Info

Invocation Target

Object Name: com.cubeia.firebase.cache:type=SystemStateCacheInfo
Interface: com.cubeia.firebase.util.TreeCacheInfoMBean

Methods

Get Local Address

Signature:

```
public String getLocalAddress();
```

Get the local socket address of the cache. This is the end point to which the cache binds.

Print Cache Content Details

Signature:

```
public String printCacheContentDetails();
```

This method attempts to print the entire cache content as a string. **NB:** Use with care as this method is very resource demanding on large caches.

Print JGroups Config

Signature:

```
public String printJGroupsConfig();
```

The cache is backed up by a JGroups channel for replication. This method returns the JGroups configuration as a string.

Get Members

Signature:

```
public String[] getMembers();
```

This method returns all members in a cache cluster, represented by their local addresses.

Get Object Count

Signature:

```
public int getObjectCount();
```

Get the number of nodes in the cache.

Print Cache Locking Info

Signature:

```
public String printCacheLockingInfo();
```

Print debugging information about the cache locks. This method may be resource heavy, so use with care.

Table Handler

Invocation Target

Object Name: com.cubeia.firebase.cache:type=TableHandler
Interface: com.cubeia.space.handler.table.TableHandlerMBean
Prerequisites: Statistics level PROFILING

Methods

Get Average Commit Time Micros

Signature:

```
public double getAverageCommitTimeMicros();
```

After events are processed in Firebase their state is replicated across the cluster. This is may be done in a transaction, this method returns the average commit time in micro seconds. If the cache does not use transaction this time represents the time it takes to write data to the socket, ie. a "semi-transaction commit".

Get Average Game State Object Size

Signature:

```
public double getAverageGameStateObjectSize();
```

This method returns the average game state size in bytes. The size is calculated from the serialized state.

Game Daemons

The game daemons in a Firebase server are modules which handles distributed messages as received from the message bus and forwards them for local processing. There are four different daemons in the system, each backed by a separate thread pool. These share a common interface, which is described below.

Methods

Get Number of Executing Threads

Signature:

```
public int getNumberOfExecutingThreads();
```

Each daemon is backed by a thread pool, this method returns the current number of executing threads.

Get Dispatched Events per Second

Signature:

```
public int getDispatchedEventsPerSecond();
```

This method returns the number of events dispatched per second. This includes both message bus events and scheduled events.

Get Execution Actions per Second

Signature:

```
public int getExecutionActionsPerSecond();
```

This method returns the number of executions per second. This is in effect the the number of times the underlying message bus have delivered a message and this message has been executed, but excludes scheduled events.

Get Average Raw Execution Time

Signature:

```
public double getAverageRawExecutionTime();
```

This method returns the time in millis of the raw execution. This number is approximate and is based on a bounded queue of 10k events.

Daemon Instances

Chat Event Daemon

Object Name:
com.cubeia.firebase.daemon:type=ReceivingChatEventDaemon
Interface: com.cubeia.firebase.server.event.processing.EventFetcherStatsMBean
Prerequisites: Statistics level PROFILING

Game Event Daemon

Object Name:
com.cubeia.firebase.daemon:type=ReceivingGameEventDaemon
Interface: com.cubeia.firebase.server.event.processing.EventFetcherStatsMBean
Prerequisites: Statistics level PROFILING

Mtt Event Daemon

Object Name:
com.cubeia.firebase.daemon:type=ReceivingMttEventDaemon
Interface: com.cubeia.firebase.server.event.processing.EventFetcherStatsMBean
Prerequisites: Statistics level PROFILING

Client Event Daemon

Object Name:
com.cubeia.firebase.daemon:type=ReceivingClientEventDaemon
Interface: com.cubeia.firebase.server.event.processing.EventFetcherStatsMBean
Prerequisites: Statistics level PROFILING

Gateway Monitor

Invocation Target

Object Name: com.cubeia.firebase.gateway:type=Monitor
Interface: com.cubeia.firebase.server.gateway.jmx.CGWMonitorMBean

Methods

Get Local Seated Clients

Signature:

```
public long getLocalSeatedClients();
```

This method returns the number of clients connected to the gateway which are also seated at one or more tables.

Get Average Game Packets per Second

Prerequisites: Statistics level PROFILING
Signature:

```
public int getAverageGamePacketsPerSecond();
```

This method returns the average number of game packets transmitted through the gateway per second. In non-profiling mode this method return -1.

Get Local Clients

Signature:

```
public int getLocalClients();
```

This method return the number of locally logged in clients.

Get Global Clients

Signature:

```
public int getGlobalClients();
```

This method returns the number of globally logged in clients.

State Lobby

Invocation Target

Object Name: com.cubeia.firebase.lobby:type=SysLobby
Interface: com.game.server.lobby.systemstate.StateLobbyMBean

Methods

Dump Subscription Info to Log

Signature:

```
public void dumpSubscriptionInfoToLog();
```

This method prints information of the lobby subscription to the server log. This subscription contains paths and object information. Use with care as this method is fairly resource demanding.

Get Broadcast Period

Signature:

```
public long getBroadcastPeriod();
```

This method return the broadcast period in milliseconds. The broadcast period is the interval in which Firebase sends lobby information to the clients.

Set Broadcast Period

Signature:

```
public void setBroadcastPeriod(long millis);
```

This method changes the broadcast period, the interval between lobby broadcasts.

Get Table Updates Per Second

Signature:

```
public int getTableUpdatesPerSecond();
```

This method returns the approximate number of change that have been recorded for tables in the lobby per second.

Pause

Signature:

```
public void pause();
```

This method temporarily halts the broadcast of lobby changes to all clients. In effect the lobby will stop updating for all connected clients.

Unpause

Signature:

```
public void unpause();
```

If the lobby is paused, this method unpauses it.

Is Paused

Signature:

```
public boolean isPaused();
```

Checks if the lobby is paused or not.

Partition Map

Invocation Target

Object Name: com.cubeia.firebase.mbus:type=PartitionMap
Interface: com.cubeia.firebase.service.messagebus.util.PartitionMapInfoMBean

Methods

Get Channels for Partition

Signature:

```
public int[] getChannelsForPartition(String partId);
```

Get all channel ids for a partition.

Get Partition for Channel

Signature:

```
public String getPartitionForChannel(int typeOrdinal, int channelId);
```

This method returns the partition id for a given channel. The channel must also be prefixed with its type, which is determined by the ordinal of the EventType enumeration.

Get All Partitions

Signature:

```
public TabularData getAllPartitions();
```

Get all partitions as tabular data.

Executors and Schedulers

All executors and schedulers in a Firebase system share a common interface.

Methods

Get Queue Size

Signature:

```
public long getQueueSize();
```

Get the number of items in the underlying queue for the thread pool.

Get Active Thread Count

Signature:

```
public long getActiveThreadCount();
```

This method returns the approximate number of currently executing threads from the pool.

Get Thread Count

Signature:

```
public long getThreadCount();
```

Get the number of threads that currently exists in the pool.

Get Task Count

Signature:

```
public long getTaskCount();
```

Returns the approximate total number of tasks that have been scheduled for execution.

Get Completed Task Count

Signature:

```
public long getCompletedTaskCount();
```

Returns the approximate total number of tasks that have completed execution.

Selected Executor Instances

Lobby Cache Listener

Object Name: com.cubeia.firebase.threads:type=executor,name=LobbyCacheListener
Interface: com.cubeia.firebase.util.executor.JmxExecutorMBean
Description: Listens for system state changes for the lobby.

Lobby Update

Object Name: com.cubeia.firebase.threads:type=executor,name=LobbyUpdate
Interface: com.cubeia.firebase.util.executor.JmxExecutorMBean
Description: Computes delta changes for the lobby.

Local Handler Service

Object Name: com.cubeia.firebase.threads:type=executor,name=LocalHandlerService
Interface: com.cubeia.firebase.util.executor.JmxExecutorMBean
Description: Hand-off pool for local services.

MTT Init Executor

Object Name: com.cubeia.firebase.threads:type=executor,name=MTT-InitExecutor
Interface: com.cubeia.firebase.util.executor.JmxExecutorMBean
Description: Hand-off for MTT initialization.

Selected Scheduler Instances

Chat Event Daemon

Object Name:
com.cubeia.firebase.threads:type=scheduler,name=ReceivingChatEventDaemon
Interface: com.cubeia.firebase.util.executor.JmxSchedulerMBean
Description: Scheduler thread pool for all chat events within the chat daemon.

Client Event Daemon

Object Name:
com.cubeia.firebase.threads:type=scheduler,name=ReceivingClientEventDaemon
Interface: com.cubeia.firebase.util.executor.JmxSchedulerMBean
Description: Scheduler thread pool for all client events within the client daemon.

Mtt Event Daemon

Object Name:
com.cubeia.firebase.threads:type=scheduler,name=ReceivingMttEventDaemon
Interface: com.cubeia.firebase.util.executor.JmxSchedulerMBean
Description: Scheduler thread pool for all tournament events within the MTT daemon.

Game Event Daemon

Object Name:
com.cubeia.firebase.threads:type=scheduler,name=ReceivingGameEventDaemon
Interface: com.cubeia.firebase.util.executor.JmxSchedulerMBean
Description: Scheduler thread pool for all game events within the game daemon.