iObeya

Platform Configuration and Optimisation

A practical configuration handbook

09/02/2023

Summary

1. Overview

1.1. Introduction

The configuration of an iObeya system can differ depending on how it will be used, so it is crucial to adjust the server size to match the anticipated usage and workload of the iObeya environment. To optimize performances, various parameters must be managed and adjusted. Therefore we share here our own set of server configuration parameters and best practices as a guide.

However, iObeya cannot be held responsible for on-premise infrastructure, organizational limitations, or capabilities. Additionally, we understand that our configurations may not always be directly applicable and should be considered as an estimate.

1.2. Purpose

- Provide a guide on hardware configuration before installing iObeya.
- Provide the best practices in terms of platform configuration.
- Provide the different parameters for optimal performances.
- Introduction to metrics and performance monitoring of the platform.

The configurations presented here go beyond the minimal configuration that is shown in the technical requirements.

1.3. Audience

This document is targeted to IT professionals who wish to deploy the iObeya platform on their own infrastructure and to manage the upgrades and the optimization of their platform.

It should be used as guidelines to set-up the platform in an optimized manner that will depend on the volume of rooms and users on the platform.

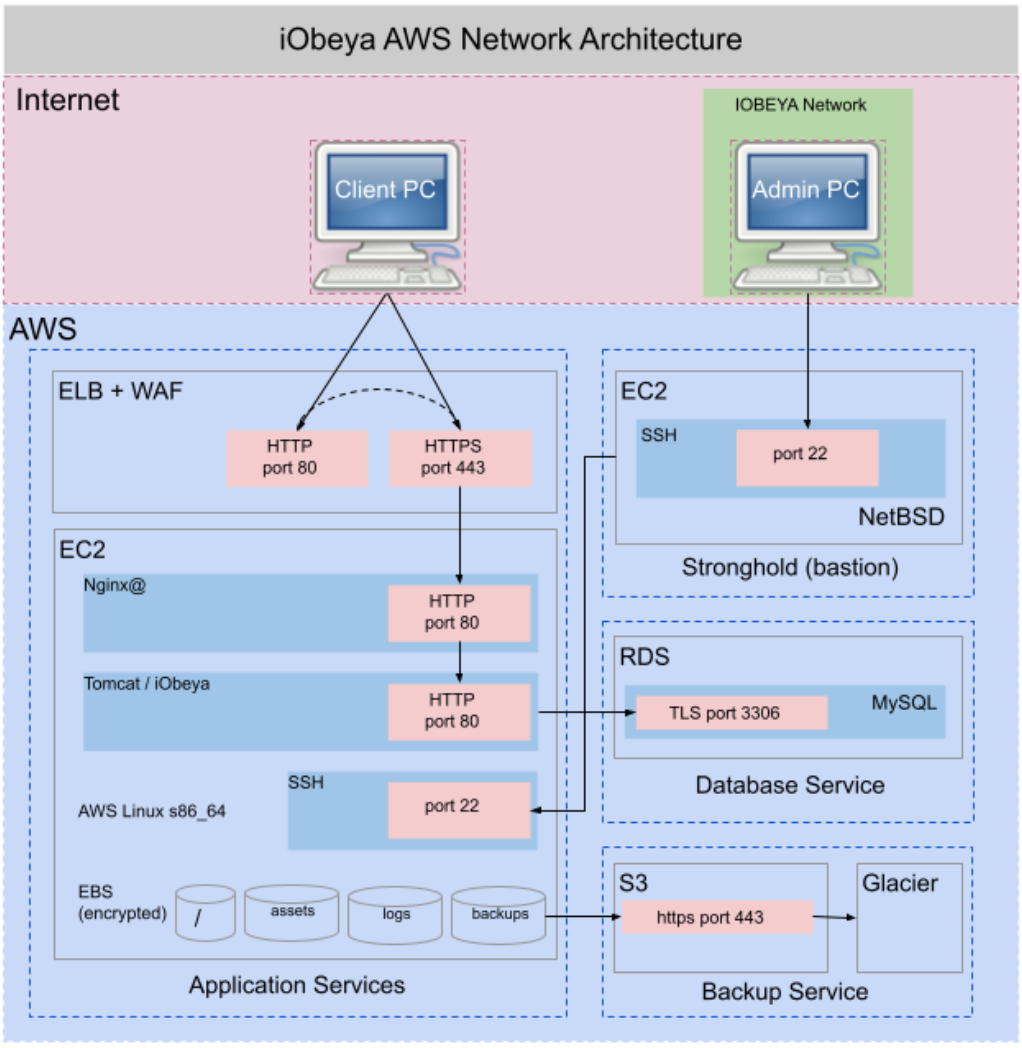
2. Preparation

2.1. Presentation

In this section, we will cover the preparation phase of the platform that will host iObeya, including how to estimate the sizing of the platform and the hardware configuration of the servers.

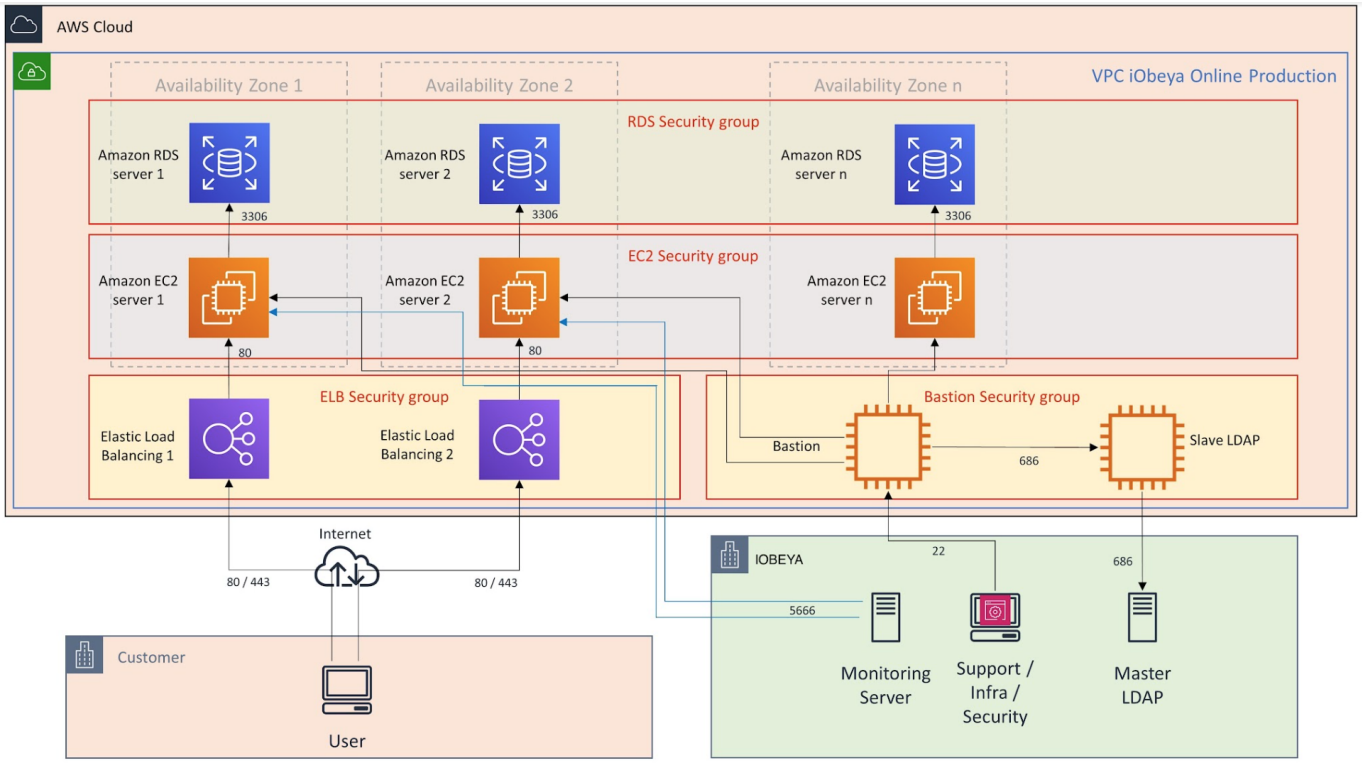
2.2. Servers architecture

For medium and large deployments, the database and the application servers must be hosted on distinct virtual machines or computers, and the global architecture is shown below, as it's currently deployed on our cloud instances:



2.3. Platform Securitization

The diagram below shows how our AWS platforms are set-up from a security perspective, in case you would like to replicate an equivalent architecture on a private AWS cloud.



This diagram show:

- The database layer: RDS is the database server, here running MySQL on a separate instance
- The application layer: EC2 is the main server, here running Apache Tomcat
- The front end layer: AWS 'elastic load balancing' at the http request level, filtering and anti-brute force protection
- The stronghold, or 'bastion', layer that provides a secure access to our platforms for our IT
- The Internet client layer

The main points here are:

- To use separate servers for the MySQL database and for Tomcat.
- To use https for security
- To secure access to the server's OS.

2.4. Platform scaling

2.4.1. Scaling

In the current generation (4.x) iObeya can only be scaled 'vertically', i.e. by raising the raw power, CPU or memory or both, of the servers.
It's not possible to scale by adding more servers, in an 'horizontal' manner.
Therefore, we have organized the scale in the form of different packages for different combinations of users and rooms.

2.4.2. Packages

Each package is characterized by the number of active users, which is the most determinant scaling factor.
The figures that we have selected in the following table come from our own experience with different sets of platforms and sizes ranging from pack 1 to pack 10, that we use for estimating the scale of our client platforms. **The package is chosen based on the estimated value of active users**.
The other values are given as a typical indication on the platform usage in terms of users and rooms, depending on the number of active users.
Depending on the projections of **active users** for your own platform, you will then select the corresponding pack from the list below:

	Pack 1	Pack 2	Pack 3	Pack 4	Pack 5	Pack 6	Pack 7	Pack 8	Pack 9	Pack 10
Active users	<20	>20 et <50	50 et <100	>100 et <150	>150 et <200	>200 <300	> 300	> 700	>1000	>1500
Nb rooms created	<=5	< 20	>20 et <100	> 100 et <500	>500	>500	>500	>500	>1000	>1500

The following pages will then provide the values of parameters that must be adjusted depending on the pack that has been chosen. When the usage of your platform will grow over the numbers of the initial package selected, you'll have to scale the configuration up to the next package level.

2.5. Hardware configuration

2.5.1. Overview

The architecture is based on separate Application and Database servers whose configurations are listed in the next section. We do not specify the type of processor as this is not the most determinant factor, as long as the VM runs on a modern processor with a decent clock speed.

2.5.2. Hardware configuration table

The table below list the CPU & RAM configurations for both the application and database server for the different packs:

	Pack 1	Pack 2	Pack 3	Pack 4	Pack 5	Pack 6	Pack 7	Pack 8	Pack 9	Pack 10
App Server RAM	4GB	8GB	16GB	32GB	32 GB	72 GB	96 GB	96 GB	96 GB	144 GB
App Server CPUs	2	2	4	8	16	36	48	48	48	72
Database server RAM	1	2	4	8	16	16	32	64	128	192
Database server vCPUs	2	2	2	2	4	4	8	16	32	48

3. Installation

3.1. Introduction

The iObeya server runs in Apache Tomcat within a JAVA environment.

The versions compatibles with iObeya are listed in the 'technical requirements guide' here:

- https://center.iobeya.com/doc/technical-requirements/

The steps to configure the JVM parameters are described in the installation guide, here, at the chapter 'Final Steps / Increase server memory allocation.

3.2. Follow the guides

The purpose of this document is not to explain how to install iObeya, as this is already detailed in the standard documentation that can be found on the iObeya resource center, for the installation of the components:

- https://center.iobeya.com/doc/setup-and-maintenance-guide/#installation-guide.html
- https://center.iobeya.com/doc/setup-and-maintenance-guide/#upgrade-guide.html

The extra installation steps described in this guide will also explain how to set-up the essential memory parameters for the JVM, so that iObeya starts.

3.3. Adjust the servers configuration

The final values for these parameters will depend on the package that you have selected, and the next sections of this guide will show you what parameters should be used.

3.4. Pre-production Server

Having a pre-production server to test installations and upgrades before deploying them in production is highly recommended, especially if iObeya usage is critical. It is imperative to validate any technical operations in a separate environment.

3.5. Backups

It's highly recommended to do backups before upgrades. However to backup iObeya is out of the scope of this document, but can be found in the installation guide.

4. Configuration

4.1. JVM configuration

4.1.1. Memory parameters

These parameters are probably the most important ones for Tomcat, and they must be set at the JVM level through the tomcat configuration tool, or directly at console level.
The following table give the recommended values for these parameters, depending on the package selected:

JVM Parameters	Pack 1	Pack 2	Pack 3	Pack 4	Pack 5	Pack 6	Pack 7	Pack 8	Pack 9	Pack 10
-XX:XMX	2G	4G	10G	21G	21G	47G	63G	63G	63G	94G
-XX:XMS	2G	4G	10G	21G	21G	47G	63G	63G	63G	94G
-XX:XMN	1G	1G	3G	7G	7G	16G	21G	21G	21G	31G
-XX:MaxMetaspaceSize	1G	1G	3G	7G	7G	16G	21G	21G	21G	31G
-XX:CompressedClassSpaceSize	1G	1G	2G	2G	2G	3G	3G	3G	3G	3G

4.1.2. G1 Garbage collector

With Java 8 and more, the garbage collector **G1** has overall better performance than the default (concurrent) GC, so it's recommended to activate it with the following option:

-XX:+UseG1GC

More explanations for fine-tuning of the GC can be found here:

- https://www.oracle.com/technical-resources/articles/java/g1gc.html

4.1.3. Other parameters

An other option that might help to save some resources is:

-XX:+UseStringDeduplication: its actual impact might be really neglectable in most situations, it doesn't hurt to activate this option.

4.2. Tomcat configuration

4.2.1. Presentation

There are many options for the configuration of Tomcat, that will be found in the **server.xml** file in the Tomcat configuration folder.
This file is preconfigured with minimal options upon installation and it is very important to adjust the values of the parameters listed below to get the best results according to the configuration shown below.
Tomcat configuration is based on a 'Connector' that can itself be related to an 'Executor' for managing a separate thread pool, as explained in the official documentation here.

4.2.2. Example of configuration for https

```
<Connector SSLEnabled="true"
port="443"
keystoreFile="C:/var/iobeya/settings/.keystore" keystorePass="iobeya"
protocol="org.apache.coyote.http11.Http11NioProtocol"
scheme="https"
secure="true"
sslProtocol="TLS"
clientAuth="false"
maxThreads="480"
minSpareThreads="24"
processorCache="720"
compression="on"
compressionMinSize="1024"
compressibleMimeType="text/html,text/xml,text/plain,text/css,text/javascript
,application/javascript,application/json,application/vnd.ms-fontobject,appl
ication/xml,image/svg+xml,font/otf,font/ttf"/>
```

To go beyond the simple example above, please refer to the Tomcat reference documentation, such as:

- SSL configuration: https://tomcat.apache.org/tomcat-8.5-doc/ssl-howto.html
- Mod proxy: http://httpd.apache.org/docs/current/mod/mod_proxy.html

4.2.3. Important parameters

The most important parameters for optimization of Tomcat performances are the ones highlighted in bold in the previous example:

- **maxThreads**: indicates how many threads can be run simultaneously, this value will change with platform sizing (see table below)
- **compression**: in any configuration, you would activate http compression that will lower the amount of data sent over https.
- **compressionMinSize**: It's inefficient to compress data < 1024 bytes.
- **compressibleMimeType**: only compress the text data, not images or binary.

4.2.4. Parameters table

The parameters that must be adjusted depending on the package are listed below:

Tomcat Parameters	Pack 1	Pack 2	Pack 3	Pack 4	Pack 5	Pack 6	Pack 7	Pack 8	Pack 9	Pack 10
maxThreads	200	200	200	200	200	200	480	480	480	720
minSpareThreads	10	10	10	10	10	24	24	24	24	36
processorCache	200	200	400	800	-1	-1	-1	-1	-1	-1

4.3. iObeya Configuration

In the iObeya context file for Tomcat (iobeya.xml or ROOT.xml), the important parameter to modify is **maxActive** and its should follow the values in the table below:

iObeya Parameters	Pack 1	Pack 2	Pack 3	Pack 4	Pack 5	Pack 6	Pack 7	Pack 8	Pack 9	Pack 10
maxActive	20	50	100	150	200	300	500	700	1000	1500

4.4. MySQL Configuration

The configuration file of MySQL (my.ini/mysql.ini or my.cnf) will be edited and the following values updated:

MySQL Parameters	Pack 1	Pack 2	Pack 3	Pack 4	Pack 5	Pack 6	Pack 7	Pack 8	Pack 9	Pack 10
innodb_buffer_pool_size	256MB	1GB	2GB	5GB	11GB	11GB	23GB	46GB	92GB	138GB
innodb_buffer_pool_instances	1	8	8	8	8	8	8	8	8	8
Innodb_thread_concurrency	0	0	0	0	0	0	0	0	0	0
max_connections	85	170	341	682	1365	1365	2730	5461	10922	16384

4.5. Global view of all parameters

	Pack 1	Pack 2	Pack 3	Pack 4	Pack 5	Pack 6	Pack 7	Pack 8	Pack 9	Pack 10
Active users	<20	>20 et <50	50 et <100	>100 et <150	>150 et <200	>200 <300	> 300	> 700	>1000	>1500
Nb users	<200	<500	<1000	>1000	>1000	>1000	>1000	>1000	>1500	>2000
Rooms	<=5	< 20	>20 et <100	> 100 et <500	>500	>500	>500	>500	>1000	>1500
Host RAM	4GB	8GB	16GB	32GB	32 GB	72 GB	96 GB	96 GB	96 GB	144 GB
Host CPUs	2	2	4	8	16	36	48	48	48	72
DB vCPUs	2	2	2	2	4	4	8	16	32	48
DB RAM	1	2	4	8	16	16	32	64	128	192
-XX:XMX	2G	4G	10G	21G	21G	47G	63G	63G	63G	94G
-XX:XMS	2G	4G	10G	21G	21G	47G	63G	63G	63G	94G
-XX:XMN	1G	1G	3G	7G	7G	16G	21G	21G	21G	31G
-XX:MaxMetaspaceSize	1G	1G	3G	7G	7G	16G	21G	21G	21G	31G
-XX:CompressedClassSpaceSize	1G	1G	2G	2G	2G	3G	3G	3G	3G	3G
maxThreads	200	200	200	200	200	200	480	480	480	720
minSpareThreads	10	10	10	10	10	24	24	24	24	36
processorCache	200	200	400	800	-1	-1	-1	-1	-1	-1
maxActive	20	50	100	150	200	300	500	700	1000	1500
innodb_buffer_pool_size	256MB	1GB	2GB	5GB	11GB	11GB	23GB	46GB	92GB	138GB
innodb_buffer_pool_instances	1	8	8	8	8	8	8	8	8	8
Innodb_thread_concurrency	0	0	0	0	0	0	0	0	0	0
max_connections	85	170	341	682	1365	1365	2730	5461	10922	16384

5. Monitoring

5.1. Presentation

There are a few key things to keep in mind in order to have a performant Java server:

1. It is important to ensure that the server has enough resources (memory, CPU, etc.) to run the application.
2. The application must be designed to be efficient and use resources wisely.
3. Monitoring of the server's performance will help identify any issues early on.

Java platform monitoring involves using various tools to track the performance of Java applications. This can include monitoring:

- the memory usage,
- CPU usage
- and overall responsiveness of the application.

5.2. Activation of JMX for Tomcat

Before being able to connect a JMX console to Tomcat, it must be enabled by specifying the following command line options for Catalina (environment variable is `CATALINA_OPTS)`, that can also be directly added in the JVM parameters definition for Tomcat:

- -Dcom.sun.management.jmxremote.port=**%my.jmx.port%**
  -> Specify here a free port to use for connection the console
- -Dcom.sun.management.jmxremote.ssl=**false**
  -> The default is no SSL for JMX
- -Dcom.sun.management.jmxremote.authenticate=**false**
  -> The default is no authentication for JMX

These last 2 options could be set to 'true' if required but it's not covered here.

The detailed documentation for JMX and Tomcat can be found here:

- https://tomcat.apache.org/tomcat-9.0-doc/monitoring.html

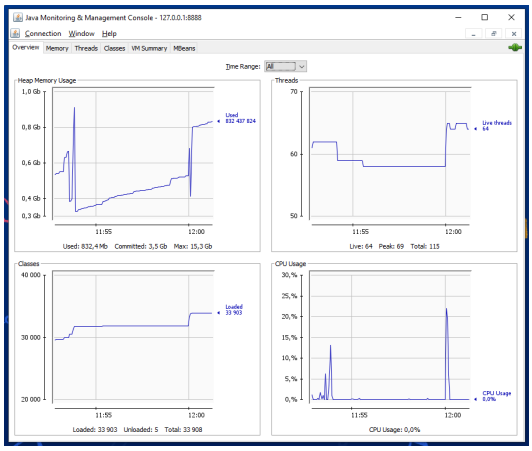
5.3. Connecting a JMX Console

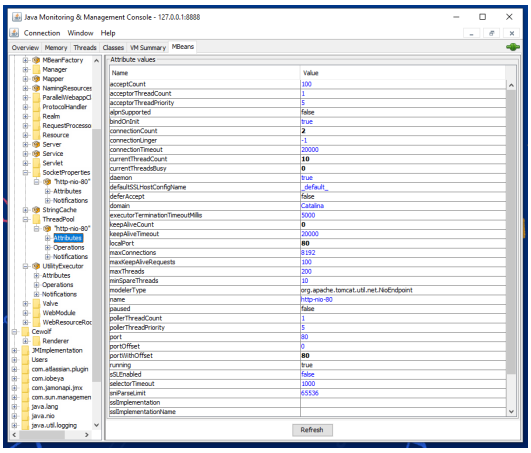
5.3.1. Presentation

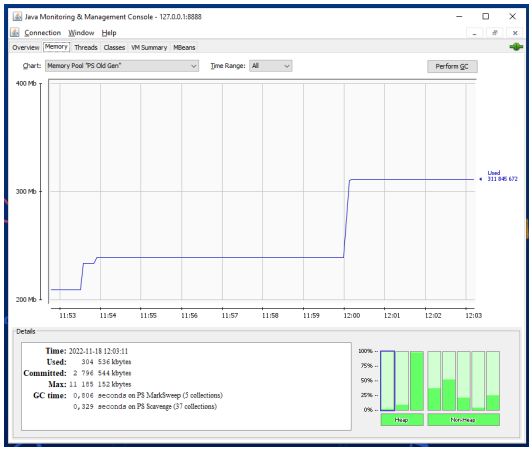
A JMX console is a tool that allows for the monitoring of a Java application through JMX and provides critical information about the good health and performance level of the Java Tomcat server.
Professional JMX management tools discussion is outside the scope of this document but at the time of writing some useful references might be found here: https://www.itprc.com/best-jmx-monitoring-tools/
You will be responsible for having a good monitoring platform that you'll connect to iObeya through JMX.

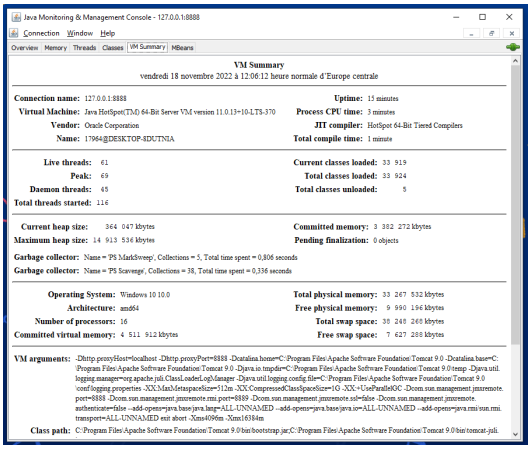
5.3.2. JConsole

JConsole is the basic monitoring tool that is provided as part of the JVM environment (to be found in the bin directory of Java) and it easily provides a lot of information, including performance metrics and memory usage.
The pictures below display examples of monitoring charts and information that can be given by JConsole.









5.4. JMX Attributes

5.4.1. General Attributes

The table below lists what beans and attributes are important to include into the monitoring dashboard, the most important attributes are displayed in bold.

Category	Bean	Attributes	Description
ThreadPool	*Catalina:type=**ThreadPool**,* *name="**http-nio-8080**",* *subType=**SocketProperties***	**maxThreads**, **currentThreadCount**, **currentThreadsBusy**	Used to diagnose a saturation of threads allocated to tomcat.
Manager	*Catalina:type=**Manager**,* *host=**localhost**,* *context=**/***	**activeSessions**	Used to evaluate the number of active users
ConnectionPool	*tomcat.jdbc:name="jdbc/iobeya",* *context=/,engine=Catalina,* *type=ConnectionPool,host=localhost,* *class=org.apache.tomcat.jdbc.pool.DataSource*	**MaxActive**, **Active**, WaitCount, Size	Used to diagnose the usage of connection pools to the database.

5.4.2. iObeya Attributes

They are in the bean named '*com.jamonapi.jmx":type=current*' and with the names listed in the table below:

Names		Attributes	Description
com.iobeya.request.dao, *com.iobeya.request.logic* *com.iobeya.request.network* *com.iobeya.request.service* *com.iobeya.request.sql* *com.iobeya.request.view*	*com.iobeya.rooms.active1* *com.iobeya.rooms.active30,* *com.iobeya.rooms.active90* *com.iobeya.users.active1* *com.iobeya.users.active30* *com.iobeya.users.active90* ***com.iobeya.users.connected***	LastValue	Used to evaluate activity and load on iObeya

5.4.3. iObeya screenshots

For boards screenshots, that are used in different places in iObeya, there is also a specific bean named '***com.iobeya:name=screenshots***' with the following available attributes:

Attributes	Description
SystemPendingScreenshot	indicates number of system based generation requests
UserPendingScreenshot	indicates number of user generation requests for screenshot
TotalPendingScreenshot	indicates global number of generation requests for screenshot
LastScreenshotGeneratedTime	indicates the time taken in generating the last screenshot (in ms)
ScreenshotGenerated	indicates the number of screenshot generated from the last bootstrap of iobeya
SystemScreenshotPeriod	indicates the period to generate a system screenshot. This variable can be set directly with JMX.