Administrators Guide

eXo Platform

Introduction

Welcome to eXo Platform 3.5

eXo Platform is the first and only integrated, cloud-ready user experience platform for building and deploying transactional websites, managing web and social content and creating gadgets and dashboards. eXo Platform lets companies leverage their existing Java infrastructure, while accommodating changing user behavior driven by consumer web technologies, such as social networks, social publishing and forums.

The following illustration gives you the overall architecture of eXo Platform 3.5.

The foundation of eXo Platform 3.5 is an enterprise portal and content management system. This provides a powerful set of REST-based services for rapid website development, content management and gadget-based development and deployment. eXo Platform 3.5 includes the following features.

- An enterprise portal serves as a powerful framework for developing portlets and other web-based user interfaces, and is based on the open source GateIn portal project co-developed by eXo and Red Hat.

- Web Content Management extends portal-based applications, allowing you to build dynamic and content-rich websites.
- Document Management for capturing and organizing documents and unstructured content, with content storage in the built-in Java Content Repository (JCR).
- xCMIS is an implementation of the full stack of Java-based CMIS (Content Management Interoperability Specification) services, so eXo-based applications can integrate with existing content management tools.
- Business Process Management (BPM) from Bonita Open Solution enables you to define workflow processes with automatic actions for web content, documents and more.
- Cloud-ready features allow eXo Platform 3.5 to run in multi-tenant environments, so social intranets and websites can take advantage of the benefits of private and public cloud platforms.

With eXo Platform 3.5, you can customize and extend your portal-based applications with user experience services to build social intranets and extranets.

- Enterprise social network features allow users to connect, collaborate within dedicated spaces, and publish real-time updates in activity streams. Support for OpenSocial provides a framework for building gadgets that can display and mash up activity information for contacts, social networks, applications and services.
- Collaboration & communication tools let you build a more productive and interactive dashboard for social intranet users. Intuitive Mail, Chat, Calendar and Address Book functionalities can seamlessly extend your portal-based web applications.
- Knowledge management features, including Forums, Answers, FAQs, and a complete enterprise wiki, can transform an extranet into an interactive online community and valuable knowledge base.
- Custom development in the web-based IDE is an intuitive web-based development environment where you can build, test and deploy client applications, such as gadgets and mash-ups, and RESTful services online. Offering the ability to extend eXo Platform online, eXo IDE instantly publishes any application that you can create and deploy immediately in your portal-based solutions.
- CRaSH is an open source tool to view and query content within a JCR server at runtime. It enables you to browse JCR trees, and serves as a shell for executing JCR operations easily, such as importing or exporting data securely. You can now extend the shell by writing Groovy commands, without recompiling easily.
- Native mobile applications for iPhone, iPad and Android allow users to easily and securely access their personalized intranet dashboards, activity streams, documents and more.

About this guide

This guide describes how to get started with eXo Platform 3.5, especially for:

- *System Administrators* who want to use, deploy and manage the eXo Platform system in their enterprises.

- *Developers* who want to know how to leverage eXo Platform in their customer projects.

Through the guide, you can do many administrative tasks when implementing eXo Platform 3.5. The administration of eXo Platform 3.5 is categorized into the following main topics:

Introduction	Overview of eXo Platform 3.5, of the administration guide and its intended readers.
Installation	Knowledge of how to install the Tomcat bundle and JBoss EARs and information of eXo Platform profiles.
Configuration	Understanding of configuration related to eXo Platform, database, cache, users and gadget proxy, file system paths, mail server, WebDAV Cache Control, Chat Server, OpenOffice Server, Log-in, JCR.
Management	Introduction to eXo Platform management, knowledge of eXo Platform management views.
Security	Changes related to the JAAS realm.
Backup	Backup of database and file systems for the JCR index and value storage.
Clustering	Changes related to clustering, which are necessary for eXo Platform to work in the cluster mode.
Deployment	How-Tos of removing sample applications, deploying a custom extension, setting up Apache Front-end and configuring the session timeout for the web server.
Integration	Instructions on how to connect eXo Platform to a populated data source, such as LDAP server, MS ActiveDirectory, or Database.

Installation

eXo Platform is packaged as a deployable enterprise archive defined by the Java EE specification, and as a configuration directory. In this chapter, you will see the following topics:

Install the Tomcat bundle

The easiest way to install eXo Platform is to use the Tomcat bundle. This is a ready-made package on top of the Tomcat 6 application server. First, you need to download and extract the package named *eXo-Platform-tomcat-3.5.x.zip* on your server.

Start up the server.

eXo Platform leverages the application server on which it is deployed. This means that you only need to start and stop your application with the default commands.

- On Linux and OS X:

```
$TOMCAT_HOME/start_eXo.sh
```

- On Windows:

```
%TOMCAT_HOME%\start_eXo.bat
```

The server is started successfully when you see the following message in your log/console:

```
INFO: Server startup in 353590 ms
```

Shut down the server.

- On Linux and OS X:

```
$TOMCAT_HOME/stop_eXo.sh
```

- On Windows:

```
%TOMCAT_HOME%\stop_eXo.bat
```

If you receive the message when you try to stop Tomcat as below, you must stop Tomcat by pressing Ctrl+C or by killing with the -9 command. To perform the **kill** action automatically, you can run **stop_eXo.sh -force** that is only available on Linux and OS X systems.

```
Tomcat did not stop in time. The PID file was not removed.
```

The server has been stopped successfully when you see the following message in your log/ console:

```
INFO: Stopping Coyote HTTP/1.1 on http-8080
```

Start up eXo Platform by running built-in startup scripts.

You can start up eXo Platform by running one of the following built-in startup scripts:

- Linux & OS X: **start_eXo.sh**.
- Windows: **start_eXo.bat**.
- Linux and OS X in the developer mode: **bin/gatein-dev.sh**.
- Windows in the developer mode: **bin/gatein-dev.bat**.

In the normal mode, the **start_eXo** scripts launch eXo Platform with the following JVM options:

```
-Xms256m
-Xmx1024m
-XX:MaxPermSize=256m
-Djava.security.auth.login.config=../conf/jaas.conf
-Dexo.conf.dir.name=gatein/conf
-Dexo.profiles=default
```

Details:

`-Xms`	Minimal Heap Size (defaults to 256 MB).
`-Xmx`	Maximal Heap Size (defaults to 1 GB).
`-Djava.security.auth.login.config`	Path to the JAAS security file where the security domains and JAAS authentication modules are declared.

`-Dexo.conf.dir.name`	Path where eXo Platform will start looking at *configuration.properties* and *configuration.xml*.
`-Dexo.profiles`	List of comma-separated profiles of eXo Plaform to activate.

In the developer mode, the **gatein-dev** scripts launch eXo Platform in the developer mode with a few JVM options.

```
-Xdebug -Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n
-Dcom.sun.management.jmxremote
-Dorg.exoplatform.container.configuration.debug
-Dexo.product.developing=true
```

Details:

-Dcom.sun.management.jmxremote	Activate the remote JMX monitoring.
-Xdebug -Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n	Enable the remote debugging.
-Dorg.exoplatform.container.configuration.debug	The container will log in to the console which the .xml file loads.
-Dexo.product.developing=true	Deactivate Javascript and CSS merging for debugging more easily. Next, activate a special language called Magic Locale "ma" showing the property keys instead of the translations.

Now, you can start and run the eXo Platform demo, but you will need to adjust these values for a production setup.

Customize environment variables in tomcat

Previously, users got used to changing the *gatein.sh* file, but this file is no longer used since Platform 3.5.0-CR1. Now, to customize the environment variables, you have to modify the *setenv.sh* file (*setenv.bat* file in Windows).

This file is located in:

- Linux & OS X: *$TOMCAT_HOME/bin/setenv.sh*.
- Windows: *$TOMCAT_HOME/bin/setenv.bat*.

The following is the content of this file:

```
LOG_OPTS="-
Dorg.apache.commons.logging.Log=org.apache.commons.logging.impl.SimpleLog"
SECURITY_OPTS="-Djava.security.auth.login.config=../conf/jaas.conf"
EXO_OPTS="-Dexo.product.developing=false -Dexo.conf.dir.name=gatein/conf"
IDE_OPTS="-Djavasrc=$JAVA_HOME/src.zip -Djre.lib=$JAVA_HOME/jre/lib"
if [ "$EXO_PROFILES" = "" -o "$EXO_PROFILES" = "-Dexo.profiles=default"] ; then
EXO_PROFILES="-Dexo.profiles=default"
fi

CATALINA_OPTS="$CATALINA_OPTS $LOG_OPTS $SECURITY_OPTS $EXO_OPTS
 $IDE_OPTS $EXO_PROFILES"
export CATALINA_OPTS
```

Install JBoss EARs

eXo Platform provides EARs packages to deploy in your existing JBoss application server. They are distributed in the package named *eXo-Platform-jboss-3.5.x.zip*.

To install eXo Platform on JBoss, follow these steps:

1. Copy the following files to *jboss-root/server/default/deploy*.

- gatein-ds.xml
- gatein.ear
- gatein-wsrp-extension-$version.ear
- starter-gatein-$version.ear
- exo-wcm-extension-$version.ear
- exo-workflow-extension-$version.ear
- exo-collaboration-extension-$version.ear
- exo-knowledge-extension-$version.ear
- exo-social-extension-$version.ear
- exo-platform-extension-$version.ear
- exo-social-intranet-$version.ear
- exo-acme-website-$version.ear
- exo-default-portal-$version.ear

2. Create the folder *jboss-root/server/default/conf/gatein* and then copy these files to this folder.

- configuration.properties
- configuration.xml

3. Copy the file *oauthkey.pem* to *jboss-root/server/default/conf/gatein/gadgets*.

4. Configure the JVM parameters.

On **Linux**, add these lines to the end of *jboss-root/bin/run.conf:*

```
# Platform environment variables
EXO_PROFILES="-Dexo.profiles=default"
EXO_OPTS="-Dexo.product.developing=false -Dexo.conf.dir.name=gatein -Dgatein.data.dir=../gatein"
REMOTE_DEBUG="-Xdebug -Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n -Dcom.sun.management.jmxremote -Dorg.exoplatform.container.configuration.debug"
EXO_XML="-Djavax.xml.stream.XMLOutputFactory=com.sun.xml.stream.ZephyrWriterFactory -Djavax.xml.stream.XMLInputFactory=com.sun.xml.stream.ZephyrParserFactory -Djavax.xml.stream.XMLEventFactory=com.sun.xml.stream.events.ZephyrEventFactory"
JAVA_OPTS="$JAVA_OPTS $EXO_OPTS $EXO_PROFILES $EXO_XML"
```

- On **Windows**, add these lines to the end of **jboss-root/bin/run.conf.bat**:

```
rem # Platform environment variables
set "EXO_PROFILES=-Dexo.profiles=default"
set "EXO_OPTS=-Dexo.product.developing=false -Dexo.conf.dir.name=gatein -Dgatein.data.dir=../gatein"
set "REMOTE_DEBUG=-Xdebug -Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n -Dcom.sun.management.jmxremote -Dorg.exoplatform.container.configuration.debug"
set "EXO_XML=-Djavax.xml.stream.XMLOutputFactory=com.sun.xml.stream.ZephyrWriterFactory -Djavax.xml.stream.XMLInputFactory=com.sun.xml.stream.ZephyrParserFactory -Djavax.xml.stream.XMLEventFactory=com.sun.xml.stream.events.ZephyrEventFactory"
set "JAVA_OPTS=%JAVA_OPTS% %EXO_OPTS% %EXO_PROFILES% %EXO_XML%"
```

Adapt to your needs:

- To use another implementation of SAX, change the class names in the *EXO_XML* variable, for example: com.sun.xml.internal.stream.XMLOutputFactoryImpl.

- To debug the application, simply add *$REMOTE_DEBUG* to the *JAVA_OPTS* variable.

5. Add the eXo Platform logging categories to *jboss-root/server/default/conf/jboss-log4j.xml*.

```
<!-- Limit the JSR170 categories -->
<category name="exo.jcr">
  <priority value="INFO"/>
</category>
<!-- Limit the JSR-168 and JSR-286 categories -->
<category name="org.exoplatform.services">
  <priority value="INFO"/>
</category>
```

6. Start up the server.

- On Linux and OS X:

```
$JBOSS_HOME/bin/run.sh
```

- On Windows:

```
%JBOSS_HOME%\bin\run.bat
```

The server is started successfully when you see the following message in your log/console:

```
INFO [org.jboss.bootstrap.microcontainer.ServerImpl] (main) JBoss
 (Microcontainer) [5.1.1 (build: ...)] Started in 5m:29s:259ms
```

7. Shut down the server.

- On Linux and OS X:

```
$JBOSS_HOME/bin/shutdown.sh
```

- On Windows:

```
%JBOSS_HOME%\bin\shutdown.bat
```

The server has been stopped successfully when you see the following message in your log/console:

```
INFO  [org.jboss.bootstrap.microcontainer.ServerImpl] (JBoss Shutdown Hook)
 Shutdown complete
```

Profiles of eXo Platform

eXo Platform comes with different runtime profiles, enabling you to customize which modules you want to enable/disable in each eXo Platform instance.

When using Tomcat, the **start_eXo** command accepts a comma-separated list of profiles. When using JBoss, you need to modify the run.conf file (run.conf.bat in Windows), then edit the line containing *EXO_PROFILES="-Dexo.profiles=default"*.

The following profiles are supported:

Profile	Description
`collaboration`	Enable the eXo Collaboration module.
`knowledge`	Enable the eXo Knowledge module.
`social`	Enable the eXo Social module.
`workflow`	Enable the Workflow add-ons within the eXo Content module.
`webos`	Enable the eXo WebOS module.

Additionally, you can use these composite profiles:

Profile	Description
`minimal`	Contains GateIn + eXo Content.
`default`	Contains all modules except workflow and webos (GateIn, eXo IDE, eXo Collaboration, eXo Social, eXo Knowledge).
`all`	All available modules.

For example:

- *./start_eXo.sh default,workflow*: Start default modules + workflow.
- *./start_eXo.sh collaboration,knowledge*: Start eXo Platform with GateIn, eXo Content, eXo Collaboration and eXo Knowledge enabled.
- *./start_eXo.sh minimal,social*: Start with eXo Social, GateIn and eXo Content.

Configuration

This chapter covers the following topics:

eXo Platform configuration

In eXo Platform, the configuration is performed in a folder whose location is controlled by a system property named *exo.conf.dir*. By default, the *gatein.sh* startup script sets this property as follows:

```
-Dexo.conf.dir.name=gatein/conf
```

So the main entry point for the eXo Platform configuration is */gatein/conf/*. This directory contains the following files:

- *configuration.properties*: the main system configuration.
- *configuration.xml*: contains the default portal container configuration.
- *portal/portal/configuration.xml*: the main external customization entry point for the default portal container.

Portal Containers, Customization and Configurations

This section explains some parts of the eXo Platform internals so that you can understand the roles of these configuration files.

The eXo Platform Kernel collects runtime components in the portal containers. A portal container holds all components to run a portal instance. It serves portal pages under the servlet context for its name.

The default portal container in eXo Platform is simply called "portal". This explains why the default URL of the samples is *http://localhost:8080/portal.*

The default portal container can be configured directly inside *exo.conf.dir*.

However, eXo Platform is capable of running several portal instances simultaneously on the same server. Each instance can be configured and customized independently via files located at: */gatein/conf/portal/$PORTAL_NAME*, where *$PORTAL_NAME* is the name of the portal container.

Note

The name of the configuration file can be altered. Please refer to the section dedicated to *PortalContainerDefinition* in the Kernel reference for more details on portal containers and other options to modify the location of the properties.

Services that run inside a portal container are declared via the xml configuration files like *configuration.xml*. Such files exist in jars, wars and below *exo.conf.dir*.

The .xml configuration files also serve as the main way to customize the portal via the multiple plugins offered by the eXo Platform components.

Additionally, the .xml files may contain variables that are populated via properties defined in *configuration.properties*. Hence, the *configuration.properties* file serves as exposing some selected variables that are necessary to configure eXo Platform in a server environment.

configuration.properties The system configuration is mostly done in the *configuration.properties* file. In most cases, this should be the only file for the system administrator to configure.

- In the Tomcat bundle, this file is located at */gatein/conf/configuration.properties*.
- In the JBoss server, this file is located at *server/default/conf/gatein/configuration.properties*.

configuration.xml This file contains the built-in configuration for the "portal" portal container.

- In most cases, you should not change this file.
- In case you do not want to use "portal" as the default portal for your project, this file can be used to import another PortalContainerDefinition into the root container.

Note

To learn more about how to configure a new portal container, please refer to the Kernel reference guide.

- **portal/portal/configuration.xml**

This file is empty by default. This is where further customizations can be placed. Generally, custom configurations are provided by extension wars. However, this file is the last loaded by the kernel. It has a higher priority over any other configuration files, including extensions. So, you can override any internal component configuration.

This may turn handy services or configurations that are not exposed in *configuration.properties*.

For example, you can decide to change the default WebDAV update policy for replace with this piece of xml:

```
<component>
  <key>org.exoplatform.services.jcr.webdav.WebDavServiceImpl</key>
  <type>org.exoplatform.services.jcr.webdav.WebDavServiceImpl</type>
  <init-params>
    <value-param>
      <name>auto-mix-lockable</name>
      <value>false</value>
    </value-param>
    <value-param>
      <name>def-folder-node-type</name>
      <value>nt:folder</value>
    </value-param>
    <value-param>
      <name>def-file-node-type</name>
      <value>nt:file</value>
    </value-param>
    <value-param>
      <name>def-file-mimetype</name>
      <value>text/plain</value>
    </value-param>
    <value-param>
      <name>update-policy</name>
      <value>replace</value>
    </value-param>
  </init-params>
</component>
```

Database configuration

eXo Platform relies on the application server for its database access, so the database must be configured as a data source at the AS level. The data source is obtained by accessing the Enterprise Naming Context (ENC) through the Java Naming and Directory Interface (JNDI) service.

Connect to a production database

If you intend to bring your eXo Platform to production, the embedded hsql database will not be appropriate and you will need to configure your app server to use another one. You need to learn how to configure eXo Platform data sources and your app server. If you need to change the data sources name, read **Change the datasources names** below.

Note

The steps below will show you how to configure eXo Platform to use a MySQL database. You need to adapt them to your actual production environment.

Refer to the Database FAQ below to find out our sample configuration files for MySQL and other DB systems.

Step 1. Prepare your database server.

You need to prepare two database schema, then do as follows:

1. Connect to your database server using the **ssh** command:

```
ssh root@db.example.org
```

2. Verify that MySQL is running:

```
sudo /etc/init.d/mysqld status
```

3. Connect to MySQL:

```
mysql -u root -p
```

In this step, you will be prompted for entering your password.

4. Create 2 databases: one for idm (*$dbname-idm*) and the other for jcr *$dbname-jcr*). For example:

i. Create the first database:

```
create database _$dbname_;
```

ii. Configure the user who have the remote access right (not only from the host server):

```
grant all on _$dbname_.* to '_$username_'@'_$IP_' identified by '_$password_';
```

In which, $IP = AS host name, $IP = IP with wildcard (eg 192.168.1.% = all IPs on 192.168.1.x network) and $username = username that eXo Platform will connect with (i.e. 'dbname*user').*

5. Verify that both databases were created successfully:

```
show databases;
```

6. Quit the server with the **exit** command.

Note

eXo Platform does not require tables to be created before it starts because these tables are created automatically on the first startup. If you want to run the DDL script to create the database objects, please contact eXo Support to obtain the script for your database.

Step 2. Configure eXo Platform.

After the database is ready, you need to configure eXo Platform to connect to it. The configuration steps may be different, depending on the application server. Here, instructions are for Tomcat and JBoss.

Tomcat bundle:

In Tomcat, the data sources configuration requires you to edit two files:

- server.xml
- starter.xml

Please refer to Tomcat's *JNDI Resources How To* [http://tomcat.apache.org/tomcat-6.0-doc/jndi-resources-howto.html] for more details on the JNDI resources binding in Tomcat.

To edit the server.xml file:

1. Open the file following the *$TOMCAT_HOME/conf/server.xml* path.

2. Declare the binding of the data sources in the GlobalNaming context:

- Change the driver from *org.hsqldb.jdbcDriver* to *com.mysql.jdbc.Driver*.
- Change the username and password to the values set above.
- Change the URL to access your DataBase from *jdbc:hsqldb:file:../gatein/data/hsql/exo-jcr_portal* to *jdbc:mysql://_$host_:3306/_$dbname_*.

The code now should look like:

```
<!-- eXo JCR Datasource for portal -->
<Resource auth="Container" driverClassName="com.mysql.jdbc.Driver" logAbandoned="true"
    maxActive="20" maxIdle="10" maxWait="10000" minEvictableIdleTimeMillis="60000"
            name="exo-jcr_portal" password="_$password_" removeAbandoned="true"
 removeAbandonedTimeout="10" type="javax.sql.DataSource" url="jdbc:mysql://_$host_:3306/
_$dbname-jcr_" username="_$username_"/>

<!-- eXo IDM Datasource for portal -->
<Resource auth="Container" driverClassName="com.mysql.jdbc.Driver" logAbandoned="true"
    maxActive="20" maxIdle="10" maxWait="10000" minEvictableIdleTimeMillis="60000"
           name="exo-idm_portal" password="_$password_" removeAbandoned="true"
 removeAbandonedTimeout="10" type="javax.sql.DataSource" url="jdbc:mysql://_$host_:3306/
_$dbname-idm_" username="_$username_"/>
```

3. Add the JDBC driver. In this step, you need to add the MySQL connector to Tomcat by adding *mysql-connector-java-5.1.x.jar* to *$TOMCAT_HOME/lib/*.

To edit the starter.xml file:

1. Open the *starter.xml* file to the *$TOMCAT_HOME/conf/Catalina/localhost/starter.xml* path.

2. Declare the resource links that make your datasources accessible to the starter webapp which is used when starting eXo Platform.

```
<ResourceLink global="exo-jcr_portal" name="exo-jcr_portal" type="javax.sql.DataSource"/>

<ResourceLink global="exo-idm_portal" name="exo-idm_portal" type="javax.sql.DataSource"/>
```

JBoss:

To configure the data source for eXo Platform under JBoss, do as follows:

- Edit gatein-ds.xml

- Add the JDBC driver

1. Edit the *gatein-ds.xml* file.

i. Open the file following the *$JBOSS_HOME/server/default/deploy/gatein-ds.xml* path.

ii. Declare the binding of the data sources in the GlobalNaming context:

- Change the driver: *org.hsqldb.jdbcDriver* to *com.mysql.jdbc.Driver*.

- Change the username and password to the values set earlier.

- Change the URL to access your database: *<connection-url>jdbc:hsqldb:${jboss.server.data.dir}${/}exo${/}hypersonic${/}exo-xxx_portal-localDB</ connection-url>* to *<connection-url>jdbc:mysql://_$host_:3306/_$dbname_</connection-url>*.

The configuration should now look like:

```
<datasources>
   <no-tx-datasource>
     <jndi-name>exo-idm_portal</jndi-name>
     <connection-url>jdbc:mysql://_$host_:3306/_$dbname-idm_</connection-url>
     <driver-class>com.mysql.jdbc.Driver</driver-class>
     <user-name>_$username_</user-name>
     <password>_$password_</password>

     <min-pool-size>5</min-pool-size>
     <max-pool-size>20</max-pool-size>
     <idle-timeout-minutes>0</idle-timeout-minutes>
     <prepared-statement-cache-size>32</prepared-statement-cache-size>
  </no-tx-datasource>
<!-- ...  -->
  <no-tx-datasource>
     <jndi-name>exo-jcr_portal</jndi-name>
     <connection-url>jdbc:mysql://_$host_:3306/_$dbname-jcr_</connection-url>
     <driver-class>com.mysql.jdbc.Driver</driver-class>
     <user-name>_$username_</user-name>
```

```
    <password>_$password_</password>

    <min-pool-size>5</min-pool-size>
    <max-pool-size>20</max-pool-size>
    <idle-timeout-minutes>0</idle-timeout-minutes>
    <prepared-statement-cache-size>32</prepared-statement-cache-size>
  </no-tx-datasource>
</datasources>
```

2. Add the JDBC driver. In this step, you need to add the MySQL connector library in JBoss by adding *mysql-connector-java-5.1.x.jar* to *$JBOSS_HOME/server/default/lib/*.

You can get the latest MySQL connector *here* [http://dev.mysql.com/downloads/connector/j/].

3. Rename the data source.

By default, eXo Platform defines two data sources:

- exo-jcr_portal - for the Java Content Repository (JCR).
- exo-idm_portal - for the organizational model.

You may want to rename the data source as follows:

i. Open and edit the configuration.properties path.

In this step, indicate to eXo name of the data sources.

```
# JNDI name of the datasource that will be used by eXo JCR
gatein.jcr.datasource.name=java:/comp/env/exojcr
...
# JNDI Name of the IDM datasource
gatein.idm.datasource.name=java:/comp/env/exo-idm
```

Note

eXo Platform automatically appends the portal container name ("*portal" by default) to these values before performing a JNDI lookup.*

ii. Change the data source name in the application server.

In this step, you need to change the name under which the data sources are bound in the JNDI tree by the app server. This is a dependent application sever.

FAQs of database configuration

Q1. How to configure eXo Platform to connect to other database systems?

Configuring eXo Platform to connect to other database can be done easily. eXo Platform provides sample configuration files in the folders:

- eXo Platform-3.5.x.zip/conf/db/
  - mysql
  - oracle
  - postgres

In each folder, you will find two sample xml files: *gatein-ds.xml* for JBoss and *server.xml* for Tomcat.

Each file contains 2 preconfigured datasources. For example (JCR datasource in *mysql/server.xml*):

```
<Resource auth="Container" driverClassName="com.mysql.jdbc.Driver" maxActive="128"
      maxIdle="32" maxWait="10000" name="exo-jcr_portal" password="${db.password}"
    testWhileIdle="true" timeBetweenEvictionRunsMillis="30000" type="javax.sql.DataSource"
                  url="jdbc:mysql://${db.host}:${db.port}/${db.jcr.name}"                  username="${db.username}"
 validationQuery="SELECT 1"/>
```

You simply replace the variables with the expected values:

Variables	Expected values
db.username	The username that connects to the database.
db.password	The password for the above user.
db.host	The hostname or IP address of the DB server.
db.port	The port to connect to the DB.
db.jcr.name	The DB name for the JCR datasource.
db.idm.name	The DB name for the IDM datasource.

Please remember to add the JDBC connector JAR in the classpath of your application server.

You can download the official JDBC connector JARs from the following websites:

- *http://www.mysql.com/downloads/connector/j/*
- *http://jdbc.postgresql.org/download.html*
- *http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html*

Q2. How to remove the idle MySQL connections?

Some RDBMSs, like MySQL, close the idle connections after a period (8 hours on MySQL by default). Thus, a connection from the pool will be invalid and any application SQL command will fail, resulting in errors as below:

```
org.hibernate.SessionException: Session is closed!
at org.hibernate.impl.AbstractSessionImpl.errorIfClosed(AbstractSessionImpl.java:72)
at org.hibernate.impl.SessionImpl.getTransaction(SessionImpl.java:1342)
```

To avoid this, you can use DBCP to monitor the idle connections and drop them when they are invalid, with the parameters **testWhileIdle**, **timeBetweenEvictionRunsMillis** and **validationQuery**.

The validation query is specific to your RDBMS. For example, on MySQL, you would use:

```
testWhileIdle="true" timeBetweenEvictionRunsMillis="30000" validationQuery="SELECT 1"
```

In which:

- **testWhileIdle** activates idle connections monitoring.
- **timeBetweenEvictionRunsMillis** defines the time interval between two checks in milliseconds (5 minutes in the example).
- **validationQuery** provides a simple SQL command to validate the connection to the RDBMS.

You can add these parameters to the data source configuration file of your application server, for example: conf/server.xml on Tomcat.

For more details on the configuration, or some examples on other RDBMS and applications servers, please refer to:

- *http://markmail.org/message/a3bszoyqbvi5qer4*

- *http://stackoverflow.com/questions/15949/javatomcat-dying-databse-connection*
- *http://confluence.atlassian.com/display/JIRA/Surviving+Connection+Closures*

Q3. How to enable managed DataSource

When want to use a managed datasource (which is the case under JBoss), configure property `gatein.jcr.datasource.managed` to **true** in *configuration.properties* file:

```
# indicates if the jcr datasource is using managed transactions.
# false by default.
gatein.jcr.datasource.managed=true
```

File system paths

eXo Platform requires the read/write access to several paths in the local file system.

```
# Arjuna configuration
com.arjuna.ats.arjuna.objectstore.objectStoreDir=${gatein.data.dir}/jta
.....

# Directory for common data (in cluster, mount it as a network shared directory between nodes)
# On standalone JBoss app server it is assignet to ${jboss.server.data.dir} by default.
exo.shared.dir=../gatein

# Configuration directory (in cluster, this directory is per node)
# Usual locations of configuration directory:
# On JBoss app server it is assigned to ${jboss.server.home.dir}/conf/gatein by default.
gatein.conf.dir=${catalina.home}/${exo.conf.dir.name}

# Data directory
gatein.data.dir=${exo.shared.dir}/data

# path for any JCR data
gatein.jcr.data.dir=${gatein.data.dir}/jcr

# path for file data inserted in JCR
gatein.jcr.storage.data.dir=${gatein.jcr.data.dir}/values

# path for the jcr index
gatein.jcr.index.data.dir=${gatein.jcr.data.dir}/index
```

The following table explains what goes in which path. The **Temporary** column indicates if the data are temporary or persistent.

Variable	Content	Temporary
`com.arjuna.ats.arjuna.objectstore.objectStoreDir`	JTA transactional data	V
`gatein.jcr.data.dir`	Directory for JCR data.	X
`gatein.jcr.data.dir`/swap	Directory for swapped data of JCR	V
`gatein.jcr.storage.data.dir`	Binary value storage for JCR.	X
`gatein.jcr.index.data.dir`	Lucene index for JCR.	X

Each variable can be defined as an absolute or a relative path. The default configuration combines them to obtain a compact tree:

```
/gatein       # gatein.data.dir
 /data
   /hsql
   /jcr      # gatein.jcr.data.dir
     /index  # gatein.jcr.index.data.dir
     /swap
     /values # gatein.jcr.storage.data.dir
   /jta
```

JCR system and default Workspaces

```
# JCR system and default workspaces
gatein.jcr.workspace.default=collaboration
gatein.jcr.workspace.system=system
```

Transaction Service

JCR transaction default timeout configured for 7 mins (420 sec) by default. If your application runs longer transactions you might need a bigger timeout.

```
# JCR Transaction Service
# TransactionService default timeout (in seconds), set it to one hour here:
```

```
gatein.jcr.transaction.timeout=3600
```

Mail server

eXo Platform requires the SMTP server to send emails, such as notifications or password reminders.

The Email service can use any SMTP account that needs to be configured in */server/default/conf/gatein/configuration.properties* (Or *$TOMCAT_HOME/ gatein/conf/configuration.properties* if you are using Tomcat).

The relevant section looks like:

```
#EMail
mail.from=
gatein.email.smtp.from=
gatein.email.smtp.username=
gatein.email.smtp.password=
gatein.email.smtp.host=smtp.gmail.com
gatein.email.smtp.port=465
gatein.email.smtp.starttls.enable=true
gatein.email.smtp.auth=true
gatein.email.smtp.socketFactory.port=465
gatein.email.smtp.socketFactory.class=javax.net.ssl.SSLSocketFactory
```

Some authenticated SMTP systems, such as GMail, GMX, MS Exchange, require the value for 'sender' or 'from' in the email message that must be identical to the SMTP user and host.

The Forum application uses it in the notification phase as the default sender. If it is not set, the sender will be empty.

mail.from	Sender's email address.
gatein.email.smtp.from	Sender's email address.
gatein.email.smtp.host	SMTP hostname.
gatein.email.smtp.port	SMTP port.
gatein.email.smtp.starttls.enable	True to enable the secure (TLS) SMTP. See RFC 3207.
gatein.email.smtp.auth	True to enable the SMTP authentication.
gatein.email.smtp.username	Username to send for authentication.
gatein.email.smtp.password	Password to send for authentication.

gatein.email.smtp.socketFactory.port	Specify the port to connect to when using the specified socket factory.
gatein.email.smtp.socketFactory.class	This class will be used to create SMTP sockets.

To see more details, refer to *JavaMail API documentation* [http://java.sun.com/products/javamail/javadocs/com/sun/mail/smtp/package-summary.html].

For eXo Knowledge, you have to add one of the following properties to the configuration file */gatein/conf/configuration.properties* to make sure that this mail service works with the authenticated SMTP systems:

```
mail.from=
gatein.email.smtp.from=
```

The value must be the exact email address of the account configured above.

WebDAV cache control

The embedded WebDAV server lets you control the cache-control http header that transmits to clients by the MIME type. This is useful for fine-tuning your website.

The configuration property is *exo.webdav.cache-control*.

```
exo.webdav.cache-control=text/*:max-age=3600;image/*:max-age=1800;*/*:no-cache;
```

The property expects a comma-separated list of key=pair values, where keys are a list of MIME types followed by the cache-control value to set.

Chat server

XMPPMessenger

If you change the host name and port for the Chat server, you will need to edit two properties:

```
# IP or hostname for the eXo Chat XMPP server
exo.chat.server=127.0.0.1

# TCP port for where the eXo Chat server listens for XMPP calls
exo.chat.port=5222
```

Chat server configuration

The standalone Chat server is configured in the *$CHATSERVER/conf/openfire.xml* file.

Configuration is based on properties expressed in the XML syntax. For example, to set the *prop.name.is.blah=value* property, you would write this xml snippet:

```
<prop>
  <name>
    <is>
      <blah>value</blah>
    </is>
  </name>
</prop>
```

Openfire has an extensive list of configuration properties. Please read the list of all properties in *Openfire documentation* [http://community.igniterealtime.org/docs/DOC-1061] for more details.

The Chat server is an Openfire server bundled with plugins and configurations that allow connectivity to eXo Platform. The following properties are used to configure it.

Property	Description	Default value
env		
serverbaseURL	Base URL for all URLs below.	*http://localhost:8080/*
restContextName	Name of the rest context.	rest
provider		
authorizedUser.name	Username to authenticate against the HTTP REST service.	root
authorizedUser.password	Password matching with provided authorizeduser.name.	password
eXoAuthProvider		
authenticationURL	URL to authenticate users.	/organization/authenticate/
authenticationMethod	HTTP method used to pass parameters.	POST
eXoUserProvider		
findUsersURL	URL to find all users.	/organization/xml/user/find-all/
findUsersMethod	HTTP method for user/find-all.	GET
getUsersURL	URL to retrieve a range of users.	/organization/xml/user/view-range/
getUsersMethod	HTTP method for user/view-range.	GET
usersCountURL	URL to count users.	/organization/xml/user/count/

Property	Description	Default value
usersCountMethod	HTTP method for user/count.	GET
userInfoURL	URL to get user information.	/organization/xml/user/info/
userInfoMethod	HTTP method for user/ information.	GET
eXoGroupProvider		
groupInfoURL	URL to get group information.	/organization/xml/group/info/
groupInfoMethod	HTTP method for information.	GET
getGroupsAllURL	URL to view all groups.	/organization/xml/group/view-all/
getGroupsAllMethod	HTTP method for group/view-all.	GET
getGroupsRangeURL	URL to view a group range.	/organization/xml/group/view-from-to/
getGroupsRangeMethod	HTTP method for group/view-from-to.	GET
getGroupsForUserURL	URL to get groups for a user.	/organization/xml/group/ groups-for-user/
getGroupsForUserMethod	HTTP method for groups-for-user.	GET
groupsCountURL	URL to count groups.	organization/xml/group/count
groupsCountMethod	HTTP method for group/count.	GET

Ports

To run the chat server properly, several ports must be opened in the firewall.

Port	Type	Description
5222 (1)	Client to server (xmpp)	The standard port for clients is to connect to the server. Connections may or may not be encrypted. You can update the security settings for this port with the *exo.chat.port* property.
9090 && 9091	Admin Console (http)	The ports used for accessing the unsecured and secured Openfire Admin Console respectively.
3478 & 3479	STUN service	

Port	Type	Description
		The port used for the service that ensures connectivity between entities behind a NAT.

OpenOffice server

eXo Platform allows users to view documents of various types directly on the content explorer through the OpenOffice server. To do so, the OpenOffice application must be available in your local device first. Next, start the OpenOffice server by running the command below:

```
soffice -headless -accept="socket,host=127.0.0.1,port=8100;urp;"
 -nofirststartwizard
```

Log-in

The login to eXo Platform is controlled by the *Java Logging API* [http://download-llnw.oracle.com/javase/1.5.0/docs/guide/logging/index.html].

By default, the login is configured to:

- log errors and warnings on the console.
- logs */gatein/logs/gatein-YYYY-MM-DD.log*.

In Tomcat, the login is configured via the *conf/logging.properties* file. Please refer to *Tomcat's Logging Documentation* [http://tomcat.apache.org/tomcat-6.0-doc/logging.html] for more information on how to adjust this file to your needs.

JCR

The set of properties controls the JCR behaviour.

```
# Type of JCR configuration to use. Possible values are:
# local : local JBC configuration
# cluster : cluster JBC configuration
gatein.jcr.config.type=local

....

# JCR dialect.
# auto : enabled auto detection
gatein.jcr.datasource.dialect=auto
```

```
# JCR Session Registry configuration
# Define here the Max Age of the JCR Session in the session registry (in seconds)
gatein.jcr.sessionregistry.sessionmaxage=300

# JCR cache configuration
gatein.jcr.cache.config=file:${gatein.conf.dir}/jcr/jbosscache/${gatein.jcr.config.type}/cache-config.xml
gatein.jcr.cache.expiration.time=15m

# JCR Locks configuration
gatein.jcr.lock.cache.config=file:${gatein.conf.dir}/jcr/jbosscache/${gatein.jcr.config.type}/lock-config.xml

# JCR Index configuration
gatein.jcr.index.cache.config=file:${gatein.conf.dir}/jcr/jbosscache/${gatein.jcr.config.type}/indexer-config.xml

# JGroups configuration
# for eXo Cache and IDM org-service (in cluster cache-config.xml files)
gatein.jgroups.config=${gatein.conf.dir}/jgroups/jgroups-udp.xml
# for JCR
gatein.jcr.jgroups.config=file:${gatein.jgroups.config}
```

Details:

`gatein.jcr.config.type`	Set to **cluster** if you want to use eXo Platform in the cluster mode. Otherwise, leave **local**.
`gatein.jcr.cache.config`	Path to the JBoss Cache configuration for the JCR cache.
`gatein.jcr.cache.expiration.time`	JCR cache expiration time.
`gatein.jcr.lock.cache.config`	Path to the JBoss Cache configuration for the JCR lock.
`gatein.jcr.index.cache.config`	Path to the JBoss Cache configuration for the JCR index.
`gatein.jgroups.config`	Path to the JGroups configuration to use for the cluster mode of eXo Cache and PicketLink IDM organization service (see below in *IDM caches*).
`gatein.jcr.jgroups.config`	Path to the JGroups configuration to use for the cluster mode.

For more details on configuring these files, please refer to the eXo JCR reference guide.

Cache configuration

Please refer to the Chapter 4 of this guide (*Cache management view* [/docs/admin/html/ch04.html#ADM.Management.Cache_management_view]) to see the description about the cache configuration.

Portal Cache Configuration

```
# Portal Cache Configuration - TemplateService
cache.exo.portal.TemplateService.capacity=3000
cache.exo.portal.TemplateService.liveTime=600

# Portal Cache Configuration - ResourceBundleData
cache.exo.portal.ResourceBundleData.capacity=3000
cache.exo.portal.ResourceBundleData.liveTime=600

# Portal Cache Configuration - MOPSessionManager
cache.exo.portal.MOPSessionManager.Capacity=5000
cache.exo.portal.MOPSessionManager.TimeToLive=60000
cache.exo.portal.MOPSessionManager.ExpirationTimeout=60000
```

Social Cache Configuration

```
# Social Cache Configuration - ActivityManager
## ActivityManager.ActivityCache
cache.exo.social.ActivityManager.ActivityCache.Capacity=4000
cache.exo.social.ActivityManager.ActivityCache.TimeToLive=43200

## ActivityManager.ActivityListCache
cache.exo.social.ActivityManager.ActivityListCache.Capacity=1000
cache.exo.social.ActivityManager.ActivityListCache.TimeToLive=7200

## ActivityManager.CommentsCache
cache.exo.social.ActivityManager.CommentsCache.Capacity=1000
cache.exo.social.ActivityManager.CommentsCache.TimeToLive=86400

# Social Cache Configuration - IdentityManager
## IdentityManager.IdentityCache
cache.exo.social.IdentityManager.IdentityCache.Capacity=1000
cache.exo.social.IdentityManager.IdentityCache.TimeToLive=3600
```

```
## IdentityManager.IdentityCacheById
cache.exo.social.IdentityManager.IdentityCacheById.Capacity=3000
cache.exo.social.IdentityManager.IdentityCacheById.TimeToLive=3600

## IdentityManager.IdentityListCache
cache.exo.social.IdentityManager.IdentityListCache.Capacity=300
cache.exo.social.IdentityManager.IdentityListCache.TimeToLive=600

# Social Cache Configuration - RelationshipManager
##  RelationshipManager.RelationshipIdCache
cache.exo.social.RelationshipManager.RelationshipIdCache.Capacity=300
cache.exo.social.RelationshipManager.RelationshipIdCache.TimeToLive=600

## RelationshipManager.RelationshipListCache
cache.exo.social.RelationshipManager.RelationshipListCache.Capacity=1000
cache.exo.social.RelationshipManager.RelationshipListCache.TimeToLive=3600

# Social Cache Configuration - SpaceStorage
cache.exo.social.SpaceStorage.SpaceCache.Capacity=300
cache.exo.social.SpaceStorage.SpaceCache.TimeToLive=7200
```

ECMS Cache Configuration

```
# ECMS Cache Configuration - Viewer
cache.exo.ecms.Viewer.PDFViewer.Capacity=300
cache.exo.ecms.Viewer.PDFViewer.TimeToLive=3600

# ECMS Cache Configuration - Drives
cache.exo.ecms.Drives.ManageDrive.Capacity=300
cache.exo.ecms.Drives.ManageDrive.TimeToLive=86400

# ECMS Cache Configuration - Scripts
cache.exo.ecms.Scripts.ScriptService.Capacity=300
cache.exo.ecms.Scripts.ScriptService.TimeToLive=86400

# ECMS Cache Configuration - Templates
cache.exo.ecms.Templates.TemplateService.Capacity=300
cache.exo.ecms.Templates.TemplateService.TimeToLive=86400

# ECMS Cache Configuration - Webcontent
cache.exo.ecms.Webcontent.InitialWebContentPlugin.Capacity=300
```

```
cache.exo.ecms.Webcontent.InitialWebContentPlugin.TimeToLive=600

# ECMS Cache Configuration - WCM Composer
cache.exo.ecms.WCMComposer.Capacity=1000
cache.exo.ecms.WCMComposer.TimeToLive=3600
```

Users configurations

Super-user configuration

In eXo Platform, the user "root" is defined as the super-admin by default. You could override this configuration by modifying the system property named *exo.super.user* defined in configuration.properties.

Default users list definition of eXo Platform

In eXo Platform, the default users, excluding Super-admin user, are defined in "Acme WebSite" and "Office Intranet" extensions. By deleting those extensions, the users "john", "demo", "james" and "mary" will not be created.

Grant users access to toolbar

Grant all users access to toolbar but not CMS

If all new members require access to the toolbar (but NOT the Content Management System (CMS)), administrators can change the default portal behavior to meet your request by doing the following steps:

1. Define the group of new members in the *organization-configuration.xml* file defined in the path *tomcat/webapps/ecmdemo/WEB-INF/conf/sample-portal/portal/organization-configuration.xml*.

```
<component-plugin>
    <name>wcm.new.user.event.listener</name>
    <set-method>addListenerPlugin</set-method>
    <type>org.exoplatform.services.organization.impl.NewUserEventListener</type>
    <description>this listener assign group and membership to a new created user</description>
    <init-params>
      <object-param>
        <name>configuration</name>
        <description>description</description>
        <object type="org.exoplatform.services.organization.impl.NewUserConfig">
          <field name="group">
            <collection type="java.util.ArrayList">
```

```
            <value>
           <object type="org.exoplatform.services.organization.impl.NewUserConfig$JoinGroup">
              <field name="groupId"><string>/platform/users</string></field>
              <field name="membership"><string>member</string></field>
            </object>
           </value>
          </collection>
         </field>
         <field name="ignoredUser">
          <collection type="java.util.HashSet">
           <value><string>james</string></value>
          </collection>
         </field>
        </object>
       </object-param>
     </init-params>
</component-plugin>
```

The sample configuration above sets the group of new users to the */platform/users* group with the **member** role.

2. Customize the value of `<access-permissions>` in the *sharedlayout.xml* file defined in the path *tomcat/webapps/ecm-wcm-extension/WEB-INF/conf/portal/portal/sharedlayout.xml* file as follows:

```
<container template="system:/groovy/portal/webui/container/UIContainer.gtmpl">
    <container template="system:/groovy/portal/webui/container/UIToolbarContainer.gtmpl">
        <!-- users containing to the following group can see the top toolbar -->
      <access-permissions>*:/platform/users</access-permissions>
   .........................
    </container>
</container>
```

3. Restart the server.

Grant all new users access to toolbar and CMS

If all users require access to both the toolbar and the underlying CMS implementation, administrators can change the default portal behavior to grant the appropriate permissions to newly registered users:

1. Do the same as Step 1 and Step 2 of "*Grant all users access to toolbar but not CMS*".

2. Edit the *organization-configuration.xml* file and add */platform/web-contributors* to **wcm.new.user.event.listener**. See the example below:

```
<component-plugin>
    <name>wcm.new.user.event.listener</name>
    <set-method>addListenerPlugin</set-method>
    <type>org.exoplatform.services.organization.impl.NewUserEventListener</type>
    <description>this listener assign group and membership to a new created user</description>
    <init-params>
      <object-param>
        <name>configuration</name>
        <description>description</description>
        <object type="org.exoplatform.services.organization.impl.NewUserConfig">
          <field name="group">
            <collection type="java.util.ArrayList">
              <value>
              <object type="org.exoplatform.services.organization.impl.NewUserConfig$JoinGroup">
                  <field name="groupId"><string>/platform/users</string></field>
                  <field name="membership"><string>member</string></field>
                </object>
              </value>
                  <!-- new users should also be a member of the web-contributors group to see the
 top toolbar -->
              <value>
              <object type="org.exoplatform.services.organization.impl.NewUserConfig$JoinGroup">
                  <field name="groupId"><string>/platform/web-contributors</string></field>
                  <field name="membership"><string>member</string></field>
                </object>
              </value>
            </collection>
          </field>
          <field name="ignoredUser">
            <collection type="java.util.HashSet">
              <value><string>james</string></value>
            </collection>
          </field>
        </object>
      </object-param>
    </init-params>
</component-plugin>
```

3. Restart the server.

Gadget configuration

Gadget proxy configuration

In eXo Platform, you could allow gadgets to load remote resources. However, this could be a potential security risk, as it will make the Gadget deployed as an open web proxy. This implies configuring ProxyFilterService.

The default configuration is set as below:

```
<component>
  <key>org.exoplatform.web.security.proxy.ProxyFilterService</key>
  <type>org.exoplatform.web.security.proxy.ProxyFilterService</type>
  <init-params>
    <values-param>
      <!-- The white list -->
      <name>white-list</name>
      <!-- Accept anything not black listed -->
      <value>*</value>
    </values-param>
    <values-param>
      <name>black-list</name>
      <value>*.evil.org</value>
    </values-param>
  </init-params>
</component>
```

This configuration can be added into *<GATEIN_CONF_DIR>/portal/<PORTAL_CONTAINER_NAME>/configuration.xml*.

- *<GATEIN_CONF_DIR>= TOMCAT_HOME/gatein/conf/* if you are using Tomcat and *JBOSS_HOME/server/<PROFILE>/conf/gatein/* if you are using JBoss.
- *<PORTAL_CONTAINER_NAME>*= the name of the used portal container, by default it is set to *portal*.

How does it work

The proxy service allow accessing any site which matches the *white-list* domains, unless it belongs to the *black-list*. If the site is not defined in neither the white list nor black list, access will be denied. Multiple values can be added for each list and wildcards can also be used.

The following is an example of the Gadget Proxy configuration:

```
<component>
  <key>org.exoplatform.web.security.proxy.ProxyFilterService</key>
  <type>org.exoplatform.web.security.proxy.ProxyFilterService</type>
  <init-params>
    <values-param>
      <name>white-list</name>
      <value>*.example.com</value>
      <value>www.example.net</value>
    </values-param>
    <values-param>
      <name>black-list</name>
      <value>evil.example.com</value>
    </values-param>
  </init-params>
</component>
```

Default OAuth key configuration

In eXo Platform, OAuth gadgets use a OAuth key to authorize with external service providers. There is always a default key defined in the *oauthkey.pem* file. This key will be used in case the OAuth gadgets do not indicate a key.

- If you are using Tomcat, the *oauthkey.pem* file is found at the *TOMCAT_HOME/gatein/gadgets* path.
- If you are using Jboss, the *oauthkey.pem* file is found at the *JBOSS_HOME/server/<PROFILE>/conf/gatein/gadgets* path.

It is strongly recommended that you create your own *oauthkey.pem* file by using the **openssl** tool and some commands as follows:

```
openssl req -newkey rsa:1024 -days 365 -nodes -x509 -keyout testkey.pem -out testkey.pem -subj
 '/CN=mytestkey'
openssl pkcs8 -in testkey.pem -out oauthkey.pem -topk8 -nocrypt -outform PEM
```

After creating the new *oauthkey.pem* file, you can use it to replace the *oauthkey.pem* default file in the *gatein/gadgets/* folder.

JCR Links cleaner job

The *links cleaning service* deletes all non valid links, which target nodes has been deleted. This property is the "cron expression" of the service that is responsible of JCR links cleaning.

You can schedule the job by adding *wcm.linkjob.cron.expression* in the *configuration.properties* file.

```
# Clear WCM orphan symlinks at 1.30 a.m every day

wcm.linkjob.cron.expression=0 30 1 * * ?
```

Note

To understand more about CRON expression, refer to *http://en.wikipedia.org/wiki/CRON_expression#CRON_expression*.

Other properties

```
# navigation controller file
gatein.portal.controller.config=${gatein.conf.dir}/controller.xml

# global portlet.xml
gatein.portlet.config=${gatein.conf.dir}/portlet.xml

# JNDI Name of the IDM datasource
# portal name will be appended to this name before the JNDI lookup
# example : java:/comp/env/exo-idm in "portal" portal will result in a JNDI lookup on context : java:/comp/env/exo-idm_portal
gatein.idm.datasource.name=java:exo-idm

# IDM
gatein.portal.idm.createuserportal=false
gatein.portal.idm.destroyuserportal=true

# IDM caches
gatein.idm.api.cache.config=file:${gatein.conf.dir}/idm/jbosscache/${gatein.jcr.config.type}/api-cache-config.xml
```

```
gatein.idm.store.cache.config=file:${gatein.conf.dir}/idm/jbosscache/${gatein.jcr.config.type}/
store-cache-config.xml

# Key files for gadget
gatein.gadgets.securitytokenkeyfile=${exo.shared.dir}/gadgets/key.txt
gatein.gadgets.signingkeyfile=${exo.shared.dir}/gadgets/oauthkey.pem
```

Management

In this chapter, the following topics are included:

Introduction to eXo Platform management

Managing resources of eXo Platform is critical for IT operators and system administrators to monitor and supervise the production system.

The eXo Platform product is exposed as a manageable set of resources that can be inspected at runtime to monitor and manage servers.

When it comes to Java, the Java Management Extension (also known as JMX) is the de-facto standard to expose managed resources externally.

This chapter explains various resources provided by the eXo Platform server, possible management actions, and how to obtain relevant metrics.

JMX interface

The resources management is exposed via the JMX layer. eXo Platform registers a set of MBean entities in an MBeanServer.

At runtime, MBeans are registered by the eXo Kernel in the MBeanServer and directly viewable in the JMX console. However, it is strongly recommended that you use a better JMX client, such as JVisualVM, available since Java 6.

To enable JMX monitoring in Tomcat, you need to pass the following system property to the VM: *-Dcom.sun.management.jmxremote*.

REST interface

The built-in REST Management Provider of eXo Platform makes some of the MBeans operations accessible as REST endpoints. Administrators can handle the system simply with a browser without performing any complex configurations.

Only members of the *platform/administrators* group are given permission to work on the REST management. The authentication requires you to log in by your own account.

The base URL to access the REST endpoints is *http://localhost:8080/rest/management*, with the last one followed by the parameter parsed in the managed resource's @RESTEndpoint annotation, leading slash then targeted operation. Consider the SkinService, which is annotated @RESTEndpoint("skinservice"); the full URL to access JMX 'getSkinList' method through the REST request is *http://localhost:8080/rest/management/skinservice/getSkinList*.

Management views of eXo Platform

PortalContainer management view

PortalContainer manages all objects and configurations of a given portal.

- The JMX name template of PortalContainer MBeans: *exo:container=portal,name="portal"*.

Attribute	Description
`ConfigurationXML`	Configuration information of the specified portal container in the XML format.
`Name`	The name of the portal container.
`RegisteredComponentNames`	The list of the registered component names.
`Started`	Indicate the portal container is started or not.

Operation	Description
`getConfigurationXML`	Return configuration information of the portal container calculated by the loading mechanism. The returned value is an XML document in the eXo Kernel format.
`getName`	Return the portal container name.
`getRegisteredComponentNames`	Return the list of all registered component names.
`isStarted`	Check if the portal container is started or not. The portal container is only started once all its components have been started.

Note

PortalContainer can be controlled through the following path:

- *http://localhost:8080/rest/management/pcontainer.*

Cache management view

eXo Platform uses caches at several levels. Monitoring them can provide the critical performance information, especially useful for tuning the server. Each cache is exposed with statistics and management operations.

- The JMX name template of Cache MBeans: *exo:service=cache,name={CacheName}* where *CacheName* is the name of each cache instance.

Attribute	Description
`Name`	The name of the cache.
`Capacity`	The maximum capacity of the cache.
`HitCount`	The total number of times the cache was successfully queried.
`MissCount`	The total number of times the cache was queried without success.
`Size`	The number of entries in the cache.
`TimeToLive`	The valid period of the cache entry in seconds. If the value is set to **-1**, the entries are never expired.

Operation	Description
`clearCache()`	Evict all entries from the cache. This method can be used to force a programmatic flush of the cache.
`getName`	Return the cache name.
`getLiveTime`	Return the valid lifetime of an entry in the cache in seconds.
`setLiveTime`	Set the valid lifetime of an entry in the cache in seconds.
`getCacheHit`	Return the total number of successful hits.
`getCacheMiss`	Return the total number of unsuccessful hits.
`getMaxSize`	Return the maximum capacity of the cache.
`setMaxSize`	Set the maximum capacity of the cache.

Operation	Description
`getCacheSize`	Return the number of entries in the cache.

Cache instances

- Portal

Cache Name	Description
`MOPSessionManager`	Cache all model objects of portal by storageId, such as pages, navigations, and preferences.
`ResourceBundleData`	Cache all resource bundles by name and locale.
`TemplateService`	Cache all Groovy templates of portal by its template path and ResourceResolver.

- JCR

Cache Name	Description
`org.exoplatform.services.jcr.impl.core.lock.LockManagerImpl`	Cache lockData by the lockToken. Cache lockData by the internal identifier of node.

- eXo Content

Cache Name	Description
`org.exoplatform.ecm.REST.viewer.PDFViewerRESTService`	Cache data of PDF files by the ObjectKey object.
`org.exoplatform.services.cms.drives.ManageDrivesService`	Cache all drives of Sites Explorer by the drive group name (String constant).
`org.exoplatform.services.cms.scripts.impl.CacheScriptService`	Cache all Groovy script files by the script name.
`org.exoplatform.services.cms.templates.TemplateService`	Cache all Groovy templates by the mechanism of org.exoplatform.groovyscript.text.TemplateService.
`org.exoplatform.services.wcm.webcontent.WebContentSchemaHandler`	Cache all artifact data by sourcePath of deploymentDescriptor. These data are reused when a new portal is deployed.
`wcm.composer`	Cache published contents by the hash generated from path, version, remoteUser, language, recursive, orderBy, orderType, primaryType of cached nodes in eXo Content.

- Social

Cache Name	Description
`org.exoplatform.social.core.manager.ActivityManager.activityCache`	Cache an activity by its id.
`org.exoplatform.social.core.manager.ActivityManager.activityFeedCache`	Cache the list of all activities by identityId and their segments.
`org.exoplatform.social.core.manager.ActivityManager.commentsCache`	Cache all comments of an activity by its id.
`org.exoplatform.social.core.manager.IdentityManager.identityGlobalId`	Cache an identity by its globalId.

`org.exoplatform.social.core.manager.IdentityManager.identityCache`
 	
`org.exoplatform.social.core.manager.IdentityManager.identityCacheById`
	Cache an identity by its id.
	
	Cache an identity by *providerId:remoteId.*

`org.exoplatform.social.core.manager.IdentityManager.identityProviderCache`	Cache the list of identities by identityProvider.
`org.exoplatform.social.core.manager.RelationshipManager.relationshipIdCache`	Cache the relationship by its id.
`org.exoplatform.social.core.manager.RelationshipManager.relationshipListCache`	Cache the list of relationships by identityId

CacheManager

The CacheManager management view enables you to control different caches.

- The JMX name template of CacheManager Mbeans: *exo:service=cachemanager*.

Operation	Description
`clearCaches()`	Force a programmatic flush of all the registered caches.

PicketLinkIDMCacheService

PicketLinkIDMCacheService is the default implementation for the organization model.

- The JMX name template of PicketLinkIDMCacheService MBeans: *exo:portal="portal",service=PicketLinkIDMCacheService,name=plidmcache*.

Operation	Description
`invalidateAll`	Invalidate all caches.
`invalidate(namespace)`	Invalidate a specific cache namespace.

Note

PicketLinkIDMCacheService can be controlled through the following path:

- *http://localhost:8080/rest/management/plidmcache.*

However, the REST View managements of Cache instances and CacheManager are not currently exposed in this version.

eXo Content management view

eXo Content provides a management view for three following services:

WCMComposer

WCMComposer is responsible for assembling pages, and is key for serving pages efficiently.

- The JMX name template of WCMComposer MBeans: *exo:portal="portal",service=composer,view=portal,type=content*.

Attribute	Description
`Cached`	Indicate the cache is used or not.
`CachedEntries`	The number of nodes in the cache.
`UsedLanguages`	The list of all languages accessible in the composer.
`UsedOrderBy`	The list of OrderBy properties accessible in the composer.
`UsedPrimaryTypes`	The list of primary types accessible in the composer.

Operation	Description
`cleanTemplates`	Clean all templates in the composer.
`setCached(iscached)`	Enable/Disable caching in the composer.
`useDefaultLanguage`	Check if the default language is used in case the translation is not published.
`getUsedPrimaryTypes`	Return the list of primary types accessible in the composer.
`getCachedEntries`	Return the number of nodes in the cache.
`isCached`	Check if the cache is used in the composer.
`getUsedLanguages`	Return the list of all languages accessible in the composer.
`getUsedOrderBy`	Return the list of OrderBy properties accessible in the composer.

FriendlyService

FriendlyService is to make URIs more friendly.

- The JMX name template of FriendlyService MBeans: *exo:portal="portal",service=friendly,view=portal,type=content*.

Attribute	Description
`Enabled`	Indicate the service is enabled or not.
`Friendlies`	The list of registered Friendly URIs.
`ServletName`	The name of the servlet referenced in the service.

Operation	Description
`addFriendly(friendlyUri, unfriendlyUri)`	Add a new Frienly Uri to the list with two parameters: friendlyUri and unfrienlyUri. The value entered in the **friendlyUri** field replaces that of the **unfriendlyUri** field.
`removeFriendly(friendlyUri)`	Remove a friendly URI from the list by entering that Uri into the **friendlyUri** field.
`isEnabled`	Check if the service is enabled or not. If the value is returned as "True", the service is enabled. If "False", the serive is disabled.
`setEnabled(isEnabled)`	Set the service as activated or deactivated by entering "True" or "False" respectively into the **isEnabled** field.
`getServletName`	Return the name of servlet referenced in the service.
`getFriendlies`	Return the list of registered friendly URIs.

WCMService

- The JMX name template of WCMService MBeans: *exo:portal="portal",service=wcm,view=portal,type=content*.

Attribute	Description
`PortletExpirationCache`	The expiration period of portlet cache in seconds.

Operation	Description
`getPortletExpirationCache`	Return the expiration period of porlet cache in seconds.
`setPortletExpirationCache(expirationCache)`	Set the expiration period of portlet cache by entering the value into the **expirationCache** field.

Note

WCMComposer, FriendlyService, and WCMService can be controlled through the following paths respectively:

- *http://localhost:8080/rest/management/wcmcomposerservice/.*
- *http://localhost:8080/rest/management/friendlyservice/.*
- *http://localhost:8080/rest/management/wcmservice/.*

JCR management view

Java Content Repository (JCR) provides a management view to monitor sessions, locks, repository configurations, and workspace configurations.

SessionRegistry

- The JMX name template of SessionRegistry MBeans: *exo:portal="portal",service=SessionRegistry,repository="portal-repository"*.

Attribute	Description
`TimeOut`	The expiration period of a JCR session.
`Size`	The number of currently active sessions.

Operation	Description
`runCleanup`	Clean all JCR sessions timed out.
`getTimeOut`	Return the timeout of a JCR session.
`getSize`	Return the number of currently active sessions.

LockManager

LockManager stores lock objects and is reponsible for removing expired locks.

- The JMX name template of LockManager MBeans: *exo:portal="portal",service=lockmanager,repository="repository",workspace={WorkspaceName}* where *WorkspaceName* is the name of each workspace.

Attribute	Description
`NumLocks`	The number of active locks.

Operation	Desription
`cleanExpiredLocks`	Remove all expired JCR locks.

Operation	Desription
`getNumLocks`	Return the number of active JCR locks.

- Each LockManager instance controls all locks of each corresponding workspace, including:

Workspace Name	Description
`backup`	Data backed up.
`collaboration`	Data of collaboration, such as sites content, documents, groups, records space, tags, and users.
`dev-monit`	Data of the IDE application.
`dms-system`	Data of DMS, including node types, templates, views, taxonomy trees.
`knowledge`	Data of knowledge, including exo:applications, and groups.
`pc-system`	State information of producer portlets and remote portlet registry.
`portal-system`	Data of the Portal model objects, such as navigations, pages, portals, application registry.
`portal-work`	Information of Gadget token and Remember me token.
`social`	Data of Social, including activity, identity, profile, relationship and space.
`system`	Data of system, including versions storage, node types, namespaces.
`wsrp-system`	Data of remote portlets.

Repository

- The JMX name template of Repository MBeans: *exo:portal="portal",container=repository,name="repository"*.

Attribute	Description
`Name`	The name of the repository container.
`RegisteredComponentNames`	The list of registered component names in the repository.

Operation	Description
`getName`	Return the repository container name.

Operation	Description
`getRegisteredComponentNames`	Return the list of registered component names in the repository.

Workspace

- The JMX name template of Workspace MBeans: *exo:portal="portal",container=workspace,repository="repository",name={WorkspaceName}* where *WorkspaceName* is the name of each workspace.

Attribute	Description
`Name`	The name of the workspace container.
`RegisteredComponentNames`	The list of registered component names in the workspace.

Operation	Description
`getName`	Return the workspace container name.
`getRegisteredComponentNames`	Return the list of registered component names in the workspace.

Note

Currently, the REST View managements of SessionRegistry, LockManager, Repository and Workspace are not exposed in this version.

Portal management view

Template statistics

Template statistics exposes various templates used by the portal and its components to render markups. Various statistics are available for individual templates, and aggregated statistics, such as the list of the slowest templates. Most management operations are performed on a single template; those operations take the template identifier as an argument.

- The JMX name template of Template statistics MBeans: *exo:portal="portal",service=statistic,view=portal,type=template*.

Attribute	Description
`TemplateList`	The list of templates loaded.
`SlowestTemplates`	The list of the 10 slowest templates.
`MostExecutedTemplates`	The list of the 10 most used templates.
`FastestTemplates`	The list of 10 fastest templates.

Operation	Description
`getAverageTime(templateId)`	Return the average rendering time of a specified template in seconds.
`getExecutionCount(templateId)`	Return the number of times the specified template has been executed.
`getMinTime(templateId)`	Return the minimum rendering time of the specified template in seconds.
`getMaxTime(templateId)`	Return the maximum rendering time of the specified template in seconds.
`getSlowestTemplates`	Return the list of the 10 slowest templates.
`getMostExecutedTemplates`	Return the list of the 10 most used templates.
`getTemplateList`	Return the list of templates loaded.
`getFastestTemplates`	Return the list of the 10 fastest templates.

Template management

Template management provides the capability to force the reload of a specified template.

- The JMX name template of Template management MBeans: *exo:portal="portal",service=management,view=portal,type=template*.

Operation	Description
`reloadTemplates`	Clear the template cache.
`listCachedTemplates`	List identifiers of the cached templates.
`reloadTemplate(templateId)`	Clear the template cache for a specified template identifier.

Skin management

- The JMX name template of Skin management MBeans: *exo:portal="portal",service=management,view=portal,type=skin*.

Attribute	Description
`SkinList`	The list of loaded skins by the skin service.

Operation	Description
`reloadSkin(skinId)`	Force a reload of the specified skin and the operation.
`reloadSkins`	Force a reload of the loaded skins.

Operation	Description
`getSkinList`	Return the list of loaded skins by the skin service.

TokenStore

- The JMX name template of TokenStore MBeans: *exo:service=TokenStore, name={Name}* where *Name* is the name of each specific token.

Attribute	Description
`Name`	The name of one specific token.
`ValidityTime`	The expiration period of one specific token in seconds.
`PeriodTime`	The expiration daemon period of one specific token in seconds. The token is deleted after the specified period.

Operation	Description
`cleanExpiredTokens`	Remove all expired tokens.
`size`	Return the number of tokens, including valid tokens and expired tokens undeleted yet.
`getName`	Return the token name.
`getValidityTime`	Return the expiration time of one specific token in seconds.
`getPeriodTime`	Return the expiration daemon period of one specific token in seconds.

eXo Platform provides the following TokenStore instances:

Token Name	Description
`gadget-token`	Store tokens of the Oauth gadget into the JCR node, such as **org.exoplatform.portal.gadget.core.GadgetTokenInfoService**.
`jcr-token`	Store common tokens into the JCR node, such as **org.exoplatform.web.security.security.CookieTokenService**, and **org.exoplatform.web.security.security.RemindPasswordToken**
`memory-token`	Store temporary tokens into the transient memory, such as **org.exoplatform.web.security.security.TransientTokenService**

Token Name	Description
`getPortalList`	Return the list of identifiers of all loaded portals.

Portal statistics

- The JMX name template of Portal statistics MBeans: *exo:portal="portal",service=statistic,view=portal,type=portal*.

Attribute	Description
`PortalList`	The list of identifiers of loaded portals.

Operation	Description
`getThroughput(portalId)`	Return the number of requests for the specified portal per second.
`getAverageTime(portalId)`	Return the average execution time of the specified portal in seconds.
`getExecutionCount(portalId)`	Return the number of times the specified portal has been executed.
`getMinTime(portalId)`	Return the minimum time of the specified portal in seconds.
`getMaxTime(portalId)`	Return the maximum time of the specified portal in seconds.
`getPortalList`	Return the list of identifiers of loaded portals.

Application statistics

Various applications are exposed to provide relevant statistics.

- The JMX name template of Application statistics MBeans: *exo:portal="portal",service=statistic,view=portal,type=application*.

Attribute	Description
`ApplicationList`	The list of loaded applications.
`SlowestApplications`	The list of the 10 slowest applications.
`MostExecutedApplications`	The list of the 10 most executed applications.
`FastestApplications`	The list of the 10 fastest applications.

Operation	Description
`getAverageTime(applicationId)`	Return the average time spent of the specified application.

Operation	Description
`getExecutionCount(applicationId)`	Return the number of times the specified application has been executed.
`getMinTime(applicationId)`	Return the minimum time spent of the specified application.
`getMaxTime(applicationId)`	Return the maximum time spent of the specified application.
`getSlowestApplications`	Return the list of the 10 slowest applications.
`getMostExecutedApplications`	Return the list of the 10 most executed applications.
`getFastestApplications`	Return the list of the 10 fastest applications.
`getApplicationList`	Return the list of application identifiers classified in the alphabetic order.

Note

Template statistics, Template management, Skin management, Portal statistics and Application statistics can be controlled through the following paths respectively:

- *http://localhost:8080/rest/management/templatestatistics.*
- *http://localhost:8080/rest/management/templateservice.*
- *http://localhost:8080/rest/management/skinservice.*
- *http://localhost:8080/rest/management/portalstatistic.*
- *http://localhost:8080/rest/management/applicationstatistic*

However, the REST View management of TokenStore is currently not exposed in this version.

eXo Knowledge management view

eXo Knowledge provides a management view, enabling you to control rules, statistics, information of data storage in Forum and Answers.

Forum

With the Forum Service management view, you can view ADMIN rules, statistics, such as the number of online users, and information of Mail Service configuration.

- The JMX name template of Forum MBeans: *exo:portal="portal",service=forum*.

Attribute	Description
AdminRules	The list of rules defining administrators.
ContactProvider	The string containing the specific ContactProvider implementation name which provides user profile to the forum, including org.exoplatform.ks.common.user.BusinessProfileContactProvider, org.exoplatform.ks.common.user.DefaultContactProvider, and org.exoplatform.ks.common.user.PersonalProfileContactProvider.
MailServiceConfig	The string containing the configuration of the Mail service used for the notifications in eXo Knowledge.
OnlineUsers	The list of currently online users.

Operation	Description
countOnlineUsers	Return the number of currently online users.
hasForumAdminRole(String username)	Check if the user is the forum administrator or not.
getAdminRules	Return the list of rules defining administrators.
getOnlineUsers	Return the list of online users.
getContactProvider	Return the name of a specific ContactProvider implementation.
setContactProvider(String contactProviderClassName	Set a specific ContactProvider implementation. The user profile on portal is obtained to populate into that of Forum.
getMailServiceConfig	Return the Mail service configuration used to send notifications in eXo Knowledge.

Job

The Job management view enables you to view state information of Jobs used in eXo Knowledge.

- The JMX name template of Job MBeans: *exo:portal="portal",service=forum,view=jobs,name={Name}* where *Name* is the name of each specific Job instance.

Attribute	Description
DataMap	The map containing the state information for Job instances.
Name	The name of the Job.

Operation	Description
`getName`	Return the names of Job instances.
`getDataMap`	Return the state information of Job instances.

eXo Knowledge provides the following Job instances:

Job	Description
`DeactiveJob`	Deactivate topics which meet TWO predefined deactivation properties: *inactiveDays* and *forumName* in Forum.
`LoginJob`	Update information of users logged in, serving for statistics.
`NotifyJob`	Send email notifications in Answers.
`RecountActiveUserJob`	Indicate the number of active users in Forum.
`SendMailJob`	Send email notifications in Forum.
`UpdateDataJob`	Update Forum statistics, such as posts or topics of users.

Plugin

RoleRulesPlugin

- The JMX name template of RoleRulesPlugin MBeans: *exo:portal="portal",service=forum,view=plugins,name="add.role.rules.plugin"*.

Attribute	Description
`AllRules`	The list of all rules of RoleRulesPlugin. For example, the rule defining 'root' user as an administrator follows the form of ADMIN=*root*.
`Description`	The brief description of RoleRulesPlugin functions.
`Name`	The name of RoleRulesPlugin.
`RuleNames`	The list of possible rule names; for example, the rule defining administrators is named ADMIN.

Operation	Description
`addRule`	Add a rule. For example, to add the ADMIN rule for the 'demo' user, you need to input two parameters: ADMIN in the p1 and demo in the p2.

Operation	Description
`getRules`	Return the list of rules defining the user with the role inputed in p1.
`getName`	Return the name of the plugin.
`getRuleNames`	Return the list of possible rule names. For example, if 'demo' and 'mary' are defined as ADMIN (ADMIN=*demo, mary*), the list of returned rule names will be *ADMIN*.
`getDescription`	Return the brief description of the plugin.
`getAllRules`	Return all rules added to the plugin.

BBCodePlugin

- The JMX name template of BBCodePlugin MBeans: *exo:portal="portal",service=ks,view=plugins,name="forum.default.bbcodes"*.

Attribute	Description
`BBCodes`	The list of BBCodes defined in the plugin.

Operation	Description
`getBBCodes`	Return the list of BBCodes. The Forum application currently provides the following BBCodes: URL, JUSTIFY, B, HIGHLIGHT, I, U, RIGHT, EMAIL, LIST, EMAIL, COLOR, URL, CODE, CENTER, CODE, QUOTE, LIST, CSS, LEFT, FONT, GOTO, QUOTE, SIZE, SLIDESHARE, IMG.

ForumInitialDataPlugin

- The JMX name template of ForumInitialDataPlugin MBeans: *exo:portal="portal",service=forum,view=plugins,name={Name}*.

Attribute	Description
`Location`	The location where the Forum export file is stored.

Operation	Description
`getLocation`	Return the location where the Forum export file is stored, for example *war:/data/forum/data-full-forum.zip*.

InitialDataPlugin

- The JMX name template of InitialDataPlugin MBeans: *exo:portal="portal",service=faq,view=plugins,name={Name}*.

Attribute	Description
`Location`	The location where the FAQ export file is stored.

Operation	Description
`getLocation`	Return the location where the FAQ export file is stored, for example *war:/data/Technical-FAQ.zip*.
`isForceXML`	Indicate if the data loaded from the FAQ export file should override any data found in the existing database.

Storage

The Storage management view enables you to get storage information in eXo Knowledge, such as data path, repository and workspace.

- The JMX name template of Forum Storage MBeans: *exo:portal="portal",service=forum,view=storage*.

Attribute	Description
`Path`	The JCR data path of the Forum Storage.
`Repository`	The name of repository containing the workspace where Forum data are stored.
`Workspace`	The name of workspace containing Forum data.

Operation	Description
`getRepository`	Return the name of repository of the Forum Storage.
`getWorkspace`	Return the name of workspace of the Forum Storage.
`getPath`	Return the JCR data path of the Forum Storage.

Note

Currently, the REST View managements of Forum, Job, Plugin, Storage are not exposed in this version.

eXo Collaboration management view

eXo Collaboration provides a management view, enabling you to view some state information of Jobs used in itself.

Attribute	Description
`DataMap`	The map containing the state information for the Job instances.
`Name`	The name of the Job.

Operation	Description
`getName`	Return the name of Job instances.
`getDataMap`	Return the state information of Job instances.

eXo Collaboration provides the following Job instances:

Job Name	Description
`PopupReminderJob`	Show pop-up reminders in the Calendar application.
`ReminderJob`	Send email reminders in the Calendar application.
`messageToHistoricalMessageJob`	Save messages in the Chat application in the data storage of eXo Collaboration.

Note

Currently, the REST View management of eXo Collaboration is not exposed.

Security

This chapter presents *changes on the JAAS realm* with the following sub-topics:

- *Tomcat*
- *Common changes*

Change the JAAS realm

eXo Platform relies on JAAS for propagating the user identity and roles to the different applications deployed on the server.

The JAAS realm is used by all eXo Platform applications and even propagated to the JCR for *Access Control* [http://docs.jboss.org/exojcr/1.12.5-GA/developer/en-US/html_single/#JCR.AccessControl].

By default, eXo Platform uses the JAAS realm named "gatein-domain". If your IT operation rules require you to use another JAAS realm, you will need to modify several files so that eXo Platform can work on your JAAS realm.

Since the security configuration is highly dependent of the app server, each application sever is represented separately.

Tomcat

In the Tomcat bundle, the JAAS configuration is controled by the *$TOMCAT_HOME/conf/jaas.conf*:

```
gatein-domain {
  org.exoplatform.web.security.PortalLoginModule required;
  org.exoplatform.services.security.jaas.SharedStateLoginModule required;
  org.exoplatform.services.security.j2ee.TomcatLoginModule required;
};
```

Replace gatein-domain by your own domain name.

Note

The PortalLoginModule module was designed to support the 'Remember me' feature. The password wrapped in PasswordCallback is the cookie token key. The PortalLoginModule module uses the token key to retrieve Credential object from CookieTokenService, then injects the correct username/password into the shared state (accessible from subsequent login modules). For the moment, to keep the authentication work properly, the PortalLoginModule module is mandatory. On the other hand, to ensure that correct

password is visible to other login modules, it should be the first one in the JAAS configuration file. The custom login module could retrieve the real password through the shared state instead of PasswordCallback.

To learn more about the syntax, or realms in Tomcat, refer to the *JAAS tutorial* [http://download.oracle.com/javase/1.4.2/docs/guide/security/jaas/tutorials/LoginConfigFile.html] or *Tomcat Realm How-To* [http://tomcat.apache.org/tomcat-6.0-doc/realm-howto.html] respectively.

For JBoss, you need to edit the default JAAS security domain in the *02portal.war!WEB-INF/jboss-web.xml* file.

```
<jboss-web>
  <security-domain>java:/jaas/gatein-domain</security-domain>
</jboss-web>
```

Additionally, you need to edit the application-policy to match the security-domain in the *gatein.ear!META-INF/gatein-jboss-beans.xml*.

```
<deployment xmlns="urn:jboss:bean-deployer:2.0">
  <application-policy xmlns="urn:jboss:security-beans:1.0" name="gatein-domain">
    <authentication>
      <login-module code="org.exoplatform.web.security.PortalLoginModule" flag="required">
        <module-option name="portalContainerName">portal</module-option>
        <module-option name="realmName">gatein-domain</module-option>
      </login-module>
            <login-module  code="org.exoplatform.services.security.jaas.SharedStateLoginModule"
 flag="required">
        <module-option name="portalContainerName">portal</module-option>
        <module-option name="realmName">gatein-domain</module-option>
      </login-module>
                    <login-module    code="org.exoplatform.services.security.j2ee.JbossLoginModule"
 flag="required">
        <module-option name="portalContainerName">portal</module-option>
        <module-option name="realmName">gatein-domain</module-option>
      </login-module>
    </authentication>
  </application-policy>
</deployment>
```

To learn more about the JBoss security configuration, refer to *JBoss Web Docs* [http://docs.redhat.com/docs/en-US/JBoss_Enterprise_Web_Platform/5/html-single/Getting_Started_Guide/index.html#Basic_Configuration_Issues-Security_Service].

Common changes

Finally, you need to do some common changes on both app servers.

configuration.properties

First, change the JAAS realm to match your own security constraints and then identify the entry named *exo.security.domain* inside the *configuration.properties* file.

```
# Realm name
exo.security.domain=gatein-domain
```

Note

Internally, eXo Platform uses this setting to set a new variable named "portal.container.realm" that is then used in the Kernel configuration files, such as *platform-extension/WEB-INF/conf/platform/repository-configuration.xml*.

portal.war

Inside *portal.war*, you should declare the Realm name in the *web.xml* file:

```
<login-config>
  <auth-method>FORM</auth-method>
  <realm-name>gatein-domain</realm-name>
  <form-login-config>
    ...
  </form-login-config>
</login-config>
```

rest.war

You also need to modify *rest.war* to provide secured REST services properly.

```
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>gatein-domain</realm-name>
</login-config>
```

Note

This change is very important, allowing you to secure the file downloads via WEBDAV.

Backup

The backup of eXo Platform instance involves backing up the databases and the file systems for the JCR index and value storage. The following topics are covered in this chapter:

- *Pre-backup*
- *eXo Platform backup*
- *Restore*
- *Third-party tools*

Note

The *gatein.data.dir* variable is defined in the *configuration.properties* file and by default points to the *$APPSEVERHOME/gatein/data* folder.

You can use the .tar command for the file backup from your application server home dir:

```
tar cvjf gatein-backup.tar.bz gatein/data
```

Pre-backup

You can start your backup strategy with the concept of a data repository. It is required to store and arrange backup data properly. The backup of one eXo Platform instance produces a set of files which can be located on various storage media (hard disk, tape, optical or solid storages, or even special remote backup services).

The files are organized in catalogs (folders) or used in different media to the concrete Platform implementation. However, it is highly recommended that you apply the *Backup rotation scheme* [http://en.wikipedia.org/wiki/Backup_rotation_scheme] to make the backup implementation effective and reliable.

Also, be sure that your available backup solution of Operating System and Database software are always used, allowing you to simplify the backup organization and avoid mistakes and data loss.

In the pre-backup, you need to stop the whole eXo Platform server.

In case of the eXo Platform clustering, every node should be stopped before the backup is performed.

The backup of eXo Platform consists of the following parts:

- Backup of the JCR data:

  - JCR index files, pointed by the configuration property: *gatein.jcr.index.data.dir*.
  - JCR value storage files, pointed by the configuration property: *gatein.jcr.storage.data.dir*.
  - JCR database backup, database specified in the JDNI configuration of Application server with the *exo-jcr_portal* name.
- Backup of Organization service database specified in the JDNI configuration of Application server with the *exo-idm_portal* name.
- Backup of Transaction service files pointed by the configuration property: *com.arjuna.ats.arjuna.objectstore.objectStoreDir*.

In the pre-backup, it is recommended that you prepare for tools, such as scripts, to restore data quickly and safely.

Note

eXo Platform indicates to one Portal application in this context, which is set by default. However, if your eXo Platform instance runs several portals, each of which has its own JCR, Organization and Transaction services, you should back up data of each portal separately.

Information provided in this chapter only describes the backup of the single-portal Platform. The backup can be repeated for each portal in your system. Only two types of JCR files are important in the backup: index and value storages.

The *gatein.jcr.data.dir* folder (*$gatein.data.dir/jcr* by default) also contains the *swap* sub-folder. The *swap* folder is used for temporary files in case BLOBs are stored in the database (see the JCR configuration guide) and has no meaning for backup.

To learn more about the basic principles of eXo Platform backup and how to create your backup implementation, see the following **Backup Planning** example:

Environment:

- The eXo Platform server runs on the RedHat Linux server. eXo Platform provides the remote database server MySQL 5.1:
  - JCR database - *jcrdb*.
  - Organization service database - *idmdb*.
- The eXo Platform files are on the network mounted storage */mnt/netfs/platform*:
  - JCR value storage files in */mnt/netfs/platform/jcr/values*.
  - JCR index files in */mnt/netfs/platform/jcr/index*.

  - Transaction service stored in */mnt/netfs/platform/jta*.

- The backup storage is located on the dedicated network mounted storage: */mnt/backupfs/my_plf_backup*.

Naming and Rotation

It is a general case when the backup is organized in the two-cycle rotation: backup files are stored everyday and older data are stored weekly, and the data storage history will be planned for three years.

To implement this approach, the daily backup is run (at night when the site is not in use) and stores result files (database and JCR files) on the network storage in the following structure:

- /my_plf_backup/2010/... - The archive folder of the previous year.
- /my_plf_backup/current/ - The archive folder of the current year.
- /my_plf_backup/current/weeks - The weekly archive folder of the current year.
- /my_plf_backup/current/weeks/01 - The archive folder of the first week of the current year.
- /my_plf_backup/current/weeks/02 - The archive folder of the second week of the current year.
- /my_plf_backup/current/weeks/N - The archive folder of the week named N of the current year.

The backup files are named to the *ISO 8601* [http://en.wikipedia.org/wiki/ISO_8601] date format:

- yyyy-MM-dd_mysql_jcrdb.tar.gz - For the JCR database.
- yyyy-MM-dd_mysql_idmdb.tar.gz - For the Organization service database.
- yyyy-MM-dd_jcr_values.tar.gz - For the JCR value storage files.
- yyyy-MM-dd_jcr_index.tar.gz - For the JCR index files.
- yyyy-MM-dd_jta.tar.gz - For the Transaction service files.

For the files backup, eXo Platform provides a shell script running on the eXo Platform server. This script does the following actions:

- Stop the eXo Platform server to ensure the full stop by the log sniffering.
- Run the database backup tool against *jcrdb* and store the result file in the archive /mnt/backupfs/my_plf_backup/current/yyyy-MM-dd_mysql_jcrdb.tar.gz.
- Run the database backup tool against *idmdb* and store the result file in the archive /mnt/backupfs/my_plf_backup/current/yyyy-MM-dd_mysql_idmdb.tar.gz.
- Copy the JCR value files to the archive /mnt/backupfs/my_plf_backup/current/yyyy-MM-dd_jcr_values.tar.gz.

- Copy the JCR index files to the archive /mnt/backupfs/my_plf_backup/current/yyyy-MM-dd_jcr_index.tar.gz.
- Copy the Transaction service files to the archive /mnt/backupfs/my_plf_backup/current/yyyy-MM-dd_jta.tar.gz.
- Copy all old archive files of 7 days to a week folder, for example /my_plf_backup/current/weeks/02.
- Delete older files of 7 days as from /my_plf_backup/current/.
- Create a folder for the previous year in /my_plf_backup/ if it is the *first week* [http://en.wikipedia.org/wiki/ISO_8601#Week_dates] of a new year, such as /my_plf_backup/2010, and then move content of /my_plf_backup/current/weeks there.
- Start the eXo Platform server.
- Send email to the administrator in case of any errors on any step.

Note

Examples of the script implementation are out of the scope of this guide.

eXo Platform backup

To back up eXo Platform, do the following main steps:

1. Stop the whole eXo Platform instance by using the shell command: *stop_eXo*. To see more details, see the *Chapter 2. Installation.*

2. Run the backup procedure, including:

- The JCR database.
- The Organization service database.
- The JCR value storage files.
- The JCR index files.
- The Transaction service files.

3. Archive the backup files to your backup storage.

4. Start the eXo Platform server.

Note

In case of the eXo Platform clustering, start the backup to the steps described in *Chapter 7. Clustering.*

Restore

Restoring from the backup can be used in several cases, such as failure, or site duplication. Similar to the backup, it is important to stop the whole eXo Platform before restoring. Next, do the main steps to start restoring from the backup of eXo Platform:

1. Unarchive the backup files from the backup storage to the temporary location.

2. Stop the whole eXo Platform instance by using the shell command stop_eXo. To see more details, see the *Chapter 2. Installation.*

3. Run the restoring procedure to restore from the backup local files for:

- The JCR database.
- The Organization service database.
- The JCR value storage files.
- The JCR index files.
- The Transaction service files.

4. Start the eXo Platform server.

Note

In case of the eXo Platform clustering, start restoring to the steps described in *Chapter 7. Clustering.*

Third-party tools

Steps described above are based on the whole backup of data. However, the *incremental backup approach* [http://en.wikipedia.org/wiki/Incremental_backup] allows you to preserve data by creating multiple copies that are based on the differences in those data. The successive copy of data only contains the portion which has been changed since the preceding copy has been created.

In steps described above, it is also possible to implement an incremental backup against the eXo Platform data.

Users of the Unix platforms can use the *rsync* tool for files synchronization and implement the incremental backup for JCR value and index files. Meanwhile, Microsoft Windows users can use the Backup utility (*Ntbackup.exe*).

These tools can be used in conjunction with a database incremental backup feature of your RDBMS to implement the eXo Platform incremental backup solution. However, all backup targets described above should be counted.

In case of the example, it is possible to organize the full backup weekly (on Sunday) and incrementals daily. The incremental backup will be faster, and reduce your site maintenance daily.

Note

It is also possible to use the ready solutions as *backula.org.* [http://www.bacula.org/]

Clustering

About clustering in eXo Platform

Clustering allows eXo Platfrom users to run various portal instances on several parallel servers which are also called nodes. The load is distributed across different servers, so the portal is still accessible via other cluster nodes in case of any failed servers. Thanks to adding more nodes to the cluster, eXo Platform's performance can be much improved. A cluster is a set of nodes which is managed together and participate in the workload management. Installing eXo Platform in the cluster mode is considered in the following main cases:

- Load Balancing: when a single server node is not enough for handling the load.
- High Availability: if one of nodes is failed, the rest of nodes in the cluster can assume the workload of the system. It means that no access is interrupted.

These characteristics should be handled by the overall architecture of your system. The Load Balancing is typically achieved by a front server or device that distributes the request to the cluster nodes. Also, the High Availability on the data layer can be typically achieved using the native replication implemented by Relation Database Management System (RDBMS) or Shared File Systems, such as SAN and NAS.

In this chapter, only the changes which are necessary for eXo Platform to work in the cluster mode are covered as below:

Set up the eXo Platform cluster

Shared file system

In eXo Platform, the persistence mostly relies on JCR, which is a middleware between the eXo Platform applications (including the Portal) and the database. Hence, this component must be configured to work in the cluster mode.

The embedded JCR server requires a portion of its state to be shared on a file system shared among the cluster nodes:

- The values storage.
- The index (in case of shared index usage).

Note

Since Platform 3.5 a local JCR index can be used on each node of the cluster. It's new feature and it needs a special configuration in Platform (desribed below).

All nodes must have the read/write access to the shared file system.

Note

It is strongly recommended that you use a mount point on a SAN.

Set up eXo Platform cluster

Following steps describe typical setup of Platform cluster:

- Switch to a *cluster* configuration.

This step is done in the *configuration.properties* file. This *configuration.properties* file must be set in the same way on all the cluster nodes. First, point the *exo.shared.dir* variable to a directory shared between cluster nodes.

```
exo.shared.dir=/PATH/TO/SHARED/FS
```

The path is shared, so all nodes will need the *read/write* access to this path. Then, switch the JCR to the cluster mode.

```
gatein.jcr.config.type=cluster
```

In this step, JCR enables the automatic network replication and discovery between other cluster nodes.

- Switch to the *cluster* profile.

You need to indicate the *cluster* kernel profile to eXo Platform. This can be done by editing startup script as below:

```
EXO_PROFILES="-Dexo.profiles=default,cluster"
```

or use the *start*`eXo.sh` *script with such parameters:*

```
./start_eXo.sh default,cluster
```

- Do the initial startup.

For the initial startup of your JCR cluster, you should only start a single node. This node will initialize the internal JCR database and create the system workspace. Once the initial node is definitely started, you can start the other nodes.

> **Note**
>
> This constraint is only for the initial start. As above, you can start the cluster in any order, but it should be started fully from the single node. After that, others can start in any order or in parallel.

- Start up and shut down.

Always start the cluster from a single node, as on initial startup, and then start all others in any order or in parallel. Nodes of the cluster will automatically try to join others at startup. Once they have discovered each other, they will synchronize their state. During the synchronization, the node is not ready to serve requests.

Advanced configuration

The cluster mode is preconfigured to work out of the box. The eXo Platform clustering fully relies on the JBossCache replication which uses JGroups internally. The default configuration of JBossCache lies in *exo.portal.component.common-x.x.x.jar*. Since eXo Platform 3.5, the JCR's JBossCache configuration is externalized to the `gatein.conf.dir` configuration folder:

- *jcr* - folder with cache configuration for JCR
- *cache* - folder with cache configuration for eXo Cache
- *idm* - folder with cache configuration for PicketLink IDM organization service
- *jgrpoups* - folder with JGroups configuration used in all caches

> **Note**
>
> The advanced configuration is optional and don't required in general case. It is recommended to do an advanced configuration only in case of a need.

JBossCache

The JBossCache configuration is done in the *configuration.properties* file using following properties:

```
# JCR cache configuration
gatein.jcr.cache.config=file:${gatein.conf.dir}/jcr/jbosscache/${gatein.jcr.config.type}/cache-config.xml
gatein.jcr.cache.expiration.time=15m

# JCR Locks configuration
gatein.jcr.lock.cache.config=file:${gatein.conf.dir}/jcr/jbosscache/${gatein.jcr.config.type}/lock-config.xml

# JCR Index configuration
gatein.jcr.index.cache.config=file:${gatein.conf.dir}/jcr/jbosscache/${gatein.jcr.config.type}/indexer-config.xml

# JGroups configuration
gatein.jgroups.jmxstatistics.enable=true
# for eXo Cache and IDM org-service (in cluster cache-config.xml files)
gatein.jgroups.config=${gatein.conf.dir}/jgroups/jgroups-udp.xml
# for JCR
gatein.jcr.jgroups.config=file:${gatein.jgroups.config}
```

By default, the nodes discovery is based on *UDP*, in which **JGroups** is responsible for the nodes identification through the UDP transport. The administrator can change the configuration of detection and ports in *jgroups-udp.xml*.

Shared file system

Optionally, if you need separate physical storage for JCR indexes and value storage files, it is possible to configure related paths, each to a separate shared file system:

```
gatein.jcr.storage.data.dir=/PATH/TO/SHARED/VALUES_FS/values
gatein.jcr.index.data.dir=/PATH/TO/SHARED/INDEX_FS/index
```

Local JCR index in cluster

Note

JCR clustering with local index on each node it is a new feature. Find more information about *Indexing in clustered environment* in JCR reference documenation.

If cluster will be used with local JCR index on each node apply following changes to the steps described above:

- configure index data to a local directory on each node:

```
gatein.jcr.index.data.dir=/PATH/TO/LOCAL/INDEX
```

- run cluster with additional profile `cluster-index-local`, edit startup script:

```
EXO_PROFILES="-Dexo.profiles=default,cluster,cluster-index-local"
```

or in command line with parameters:

```
./start_eXo.sh default,cluster,cluster-index-local
```

FAQs of clustering

Q1. **How to migrate from local to the cluster mode?**

If you intend to migrate your production system from local (non-cluster) to the cluster mode, follow these steps:

1. Update the configuration to the cluster mode as explained above on your main server.

2. Use the same configuration on other cluster nodes.

3. Move the index and value storage to the shared file system.

4. Start the cluster.

Q2. **Why is startup failed with the "Port value out of range" error?**

On Linux, your startup is failed if you encounter the following error:

```
[INFO] Caused by: java.lang.IllegalArgumentException: Port value out of range: 65536
```

This problem happens under specific circumstances when JGroups-the networking library behind the clustering attempts to detect the IP to use for communication with other nodes.

You need to verify:

- The host name is a valid IP address, served by one of the network devices, such as eth0, eth1.
- The host name is NOT defined as localhost or 127.0.0.1.

Q3. **How to solve the "failed sending message to null" error?**

If you encounter the following error when starting up in the cluster mode on Linux:

```
Dec 15, 2010 6:11:31 PM org.jgroups.protocols.TP down
SEVERE: failed sending message to null (44 bytes)
java.lang.Exception: dest=/228.10.10.10:45588 (47 bytes)
```

Be aware that clustering on Linux only works with IPv4. Therefore, when using a cluster under Linux, add the following property to JVM parameters:

```
-Djava.net.preferIPv4Stack=true
```

Deployment

This chapter covers the following topics:

Remove sample applications

eXo Platform comes with two sample portals that show the capabilities of the product. However, once implementing your own extensions, you may not need the sample applications. In some cases, you usually want to remove them before deploying your system in production.

The following instructions are used in cases where the hsqldb embedded database configuration is used.

Remove Acme website/Acme Social Intranet

Both Acme website and Acme Social Intranet are sample extensions that demonstrate the intranet you can implement with eXo Platform.

To remove the Acme site/Acme Social Intranet, do as follows:

1. Stop the server using the command: *shutdown.sh*.

2. Delete the following relevant files:

- For the Acme website: *acme-portal.war*, *exo.ecms.ext.acme.config.jar* and *gatein/data*.
- For the Acme Social Intranet: *office-portal.war*, *exo.platform.office.config.jar*, and *gatein/data*.

3. Restart the server.

Remove crash

Crash is a complementary tool for development and maintenance. As it opens telnet and ssh sockets, it is highly recommended that you remove crash for your production deployments. The crash is covered in the **crash.war** file in *tomcat/webapps*.

To remove crash, do as follows:

1. Stop the serve by using the *shutdown.sh* command.

2. Delete the *crash.war* file.

3. Restart the server.

Deploy a custom extension

Extensions are packaged as the Java EE web applications and packaged as the normal .war files. To deploy the custom extension, you may do as for any other web apps.

In Tomcat, this ends up by copying the war archive to the *webapps* folder.

However, the GateIn extension mechanism imposes that the *starter.war* webapp starts after all extension wars.

This is the case for the sample applications bundled by default.

There are several ways to control the loading order of webapps in Tomcat. Please refer to *Tomcat's Deployer How To* [http://tomcat.apache.org/tomcat-6.0-doc/deployer-howto.html#A_word_on_Contexts].

Set up Apache front-end

It may be necessary to use the HTTP server as a front-end for Tomcat. For example, you may want to keep more than one application server on the same host, and/or you want to access these app servers with the separate DNS names, without adding a port to the URL.

There are two methods that allow you to "*glue*" Apache HTTP Daemon and Tomcat application server:

- via the HTTP protocol using *proxy module* [http://httpd.apache.org/docs/2.2/mod/mod_proxy.html].
- via the *Apache JServ Protocol* [http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html] using *Tomcat connector* [http://tomcat.apache.org/connectors-doc/] or *HTTPD AJP proxy module* [http://httpd.apache.org/docs/2.2/mod/mod_proxy_ajp.html].

Base configuration for Apache

First, you need to configure a new virtual host in Apache HTTPD for the application server. This is the simplest example of a virtual host:

```
<VirtualHost *:80>
  ServerName      Enter your server DNS name here
  RedirectMatch permanent "^/?$" "/portal/"
```

```
</VirtualHost>
```

You can find more information on the Apache HTTP daemon host *here* [http://httpd.apache.org/docs/2.2/vhosts/].

Connect via HTTP protocol (Apache mod_proxy)

With the *glue* method, it is necessary to configure the Apache HTTP daemon to work as the **reverse** proxy, which will redirect the client's requests to the app server's HTTP connector.

For this connection type, you need to include the **mod_proxy** module in the HTTP daemon configuratinon file. This can be found in the *httpd.conf* file, which is usually located at */etc/httpd/conf/*. However, depending on your OS, this path may vary. You will then need to add some directives to your virtual host configuration.

```
ProxyRequests   Off
ProxyPass  "/" http://YOUR_AS_HOST:AS_HTTP_PORT/
ProxyPassReverse "/" http://YOUR_AS_HOST:AS_HTTP_PORT/
```

Details:

- *YOUR_AS_HOST* - host (IP or DNS name) is the location of your application server. If you run the HTTP daemon on the same host as your app server, you can change this to **localhost**.
- *AS_HTTP_PORT* - port is the location where your app server will listen to incoming requests. For Tomcat, this value is 8080 by default. You can find the value at *tomcat/conf/server.xml*.

In the above example, the HTTP daemon will work in the **reverse proxy** mode (ProxyRequests Off) and will redirect all requests to the tcp port 8080 on the localhost. So, the configuration of a virtual host looks like the following:

```
<VirtualHost *:80>
  ServerName      Enter your server DNS name here
  RedirectMatch permanent "^/?$" "/portal/"
  ProxyRequests   Off
  ProxyPass  "/" http://localhost:8080/
  ProxyPassReverse "/" http://localhost:8080/
</VirtualHost>
```

For more detail on **mod_proxy**, refer to this *documentation* [http://httpd.apache.org/docs/2.2/mod/mod_proxy.html].

Connect via AJP protocol

As described above, the 'glue' method can be implemented via one of the following ways:

- **The first way:** Use the native Apache HTTP Daemon's *AJP proxy module* [http://httpd.apache.org/docs/2.2/mod/mod_proxy_ajp.html].
- **The second way:** Use the native Apache Tomcat's *AJP connector* [http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html].

With the first method, you only need the HTTP daemon and application server, but settings are limited.

With the second method, you can obtain more settings, but you will need to download and install additional modules for the HTTP Daemon that are not included in the default package.

AJP proxy module

Make sure that *mod_proxy_ajp.so* is included in the list of loadable modules. Add the following to your virtual host configuration setting:

```
ProxyPass / ajp://localhost:8009/
```

In this example, the app server is located on the same host as the Apache HTTP daemon, and accepts incoming connections on the port 8009 (the default setting for the Tomcat application server). You can find the full list of virtual host configurations here:

```
<VirtualHost *:80>
  ServerName      Enter your server DNS name here
  RedirectMatch permanent "^/?$" "/portal/"
  ProxyRequests   Off
  ProxyPass / ajp://localhost:8009/
</VirtualHost>
```

Apache Tomcat's AJP connector

1. Download AJP connector module *here* [http://apache.vc.ukrtel.net/tomcat/tomcat-connectors/jk/binaries/].

2. Move the downloaded *mod_jk.so* file to the HTTPD's module directory, for example */etc/httpd/modules*. The directory may be different, depending on the OS.

3. Create the configuration file for the *mod_jk.conf* module.

```
LoadModule    jk_module  modules/mod_jk.so
<IfModule jk_module>
  # ---- Where to find workers.properties
        JkWorkersFile   conf.d/workers.properties
  # ---- Where to put jk logs
  JkLogFile       logs/mod_jk.log
  # ---- Set the jk log level [debug/error/info]
  JkLogLevel    info
  # ---- Select the timestamp log format
  JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
  JkRequestLogFormat "%w %R %T"
  # ---- Send everything for context /examples to worker named worker1 (ajp13)
  JkMountFileReload       "0"
</IfModule>
```

For more details, see the *Tomcat documentation* [http://tomcat.apache.org/connectors-doc/reference/apache.html].

4. Place the *mod_jk.conf* file into the directory where other configuration files for Apache HTTP daemon are located. For example, */etc/httpd/conf.d/*.

5. Create the workers.properties file, which defines *AJP workers* [http://tomcat.apache.org/connectors-doc/generic_howto/workers.html] for the HTTP daemon.

```
worker.list=status, WORKER_NAME
# Main status worker
worker.stat.type=status
worker.stat.read_only=true
worker.stat.user=admin
# Your AJP worker configuration
worker.WORKER_NAME.type=ajp13
worker.WORKER_NAME.host=localhost
worker.WORKER_NAME.port=8109
worker.WORKER_NAME.socket_timeout=120
worker.WORKER_NAME.socket_keepalive=true
```

Note

In the example above, you can change *WORKER_NAME* to any value.

6. Put this file in the same directory as the *mod_jk.conf* file.

7. Update the virtual host configuration.

```
<VirtualHost *:80>
  ServerName      Enter your server DNS name here
  RedirectMatch permanent "^/?$" "/portal/"
  ProxyRequests   Off
  JkMount         /* WORKER_NAME
</VirtualHost>
```

Configure the session timeout for the web server

The session timeout defines the validation period of a session. In the portal environment, such as eXo Platform, it is highly recommended that all web applications have the same session timeout value:

- Tomcat
- JBoss

The session timeout is configurable individually for each web application in the *web.xml* file:

```
<!-- ==================== Default Session Configuration ================ -->
<!-- You can set the default session timeout (in minutes) for all newly   -->
<!-- created sessions by modifying the value below.                       -->
<session-config>
  <session-timeout>30</session-timeout>
</session-config>
```

Tomcat server

In the Tomcat bundle, this file is located at *$TOMCAT_HOME/conf/web.xml*. To configure the session timeout of Server Tomcat, do as follows:

1. Stop the server by the *shutdown.sh* command.

2. Open the *web.xml* file.

3. Change the value of the session-timeout.

4. Save and then restart the server by the *gatein.sh* command.

JBoss server

In the JBoss server, this file is located at *$Jboss_home/server/default/deployers/jbossweb.deployer/web.xml*. To configure the session timeout of the JBoss server, do as follows:

1. Stop the server by using the *shutdown.sh* command.

2. Open the *web.xml* file.

3. Change the value of the session timeout.

4. Save and restart the server.

Organization Integration

The eXo Platform integration with another systems is very important. To make eXo Platform work with *predefined organizational data* properly, it is necessary to initialize some backend settings for each *organizational element*. Operations involved in synchronizing eXo Platform's backend settings with the organizational entities are called **organizational model integration**.

The goal of this chapter is to instruct you how to connect eXo Platform to a populated organizational data source, such as LDAP Server, MS ActiveDirectory, or Database, through the following topics:

Terminology

Before learning about how to integrate the organizational model, you should be aware of the following terms:

- **Organizational data** are information of users, user profiles, groups, memberships and membership types.
- **Organizational element** refers to a user, user profile, group, membership or membership type.
- **Active organizational element** is an *organizational element* that eXo Platform has already integrated, and so can be used by eXo Platform's features.
- **Predefined organizational data** are organizational data which are fulfilled in the data source without using eXo Services.
- **Listener** is a part of eXo Platform organization management. When an organizational element is added, a set of listeners is triggered to integrate it into eXo Platform. For example, when a user is added to eXo Platform, there is a listener which will add its private and public drives.

Integrate the organizational model

When an external organizational data source (LDAP server, MS ActiveDirectory) is used, eXo Platform must be notified of any changes on organizational entities, including addition, deletion

and update. These changes are reflected in the backend settings of eXo Platform. These notifications are performed thanks to several means provided by eXo Platform. Pick up one of the following use cases related to your needs.

Pick up one of the following use cases of integration execution related to your needs.

eXo Platform start-up

At the start-up of eXo Platform, all groups are synchronized. This means that the groups which have been added/deleted will be integrated. This operation is mandatory because some of eXo Platform features require some system groups to be integrated, such as the system group of eXo Social **spaces**.

In case you want to disable groups synchronization at start-up, and proceed manually to the system groups synchronization, you must modify an init param of *OrganizationIntegrationService*:

```
<component>
  <type>org.exoplatform.platform.organization.integration.OrganizationIntegrationService</type>
  <init-params>
    ...
    <value-param>
      <name>synchronizeGroups</name>
      <value>false</value>
    </value-param>
  </init-params>
</component>
```

User login

Once users have logged in, their profile, memberships and related groups will be auto-synchronized. Also, administrators can activate the synchronization process manually without depending on the users' login. (See the next sections for more details).

Manual sync

You could enforce the integration of some *organizational elements* via REST or JMX. See the operations you can perform in the following table.

Operation	Description
invokeAllListeners	Synchronize and integrate all organizational elements.
invokeGroupsListeners	Synchronize and integrate all groups stored in the data source.

Operation	Description
invokeGroupListeners	Synchronize and integrate a selected group stored in the data source.
invokeUsersListeners	Synchronize and integrate all users stored in the data source.
invokeUserListeners	Synchronize and integrate a selected user stored in the data source.
invokeMembershipListeners	Synchronize and integrate a specific membership.

Note

For invokeAllListeners, invokeGroupsListeners and invokeUsersListeners, it may take few hours if the organizational data source contains thousands of users.

Scheduled/Periodic sync

You can select the periodic integration of the whole *organizational elements* which are not integrated yet. This feature is not activated automatically in the eXo Platform distribution. To do so, you will have to add this configuration:

```
<external-component-plugins>
            <target-component>org.exoplatform.services.scheduler.JobSchedulerService</target-component>
 <component-plugin>
  <name>OrgInitializerCronJob</name>
  <set-method>addCronJob</set-method>
  <type>org.exoplatform.services.scheduler.CronJob</type>
  <description>Schedule the organization integration operation</description>
  <init-params>
   <properties-param>
    <name>cronjob.info</name>
    <description>Invoke initializer periodically</description>
    <property name="jobName" value="OrgInitializerCronJob"/>
    <property name="groupName" value="group"/>
                                                    <property name="job" value="org.exoplatform.platform.organization.integration.OrganizationIntegrationJob"/>
    <property name="expression" value="0 45 23 * * ? *"/>
   </properties-param>
  </init-params>
 </component-plugin>
</external-component-plugins>
```

You need to modify the **expression** property value to change the period according to the *CRON Expression documentation* [http://www.quartz-scheduler.org/docs/tutorial/TutorialLesson06.html]. The value presented above means that this job is executed at 23:45:00 everyday.

Use REST

The organization integration management API is also exposed as REST. You need to login as an administrator first, then use one of those links:

- */portal/rest/management/orgsync/syncUser?username=USERNAME&eventType=EVENT*

Para-value	Description
USERNAME	Select a username.
EVENT	Select one of the following values. - **ADDED** - Select this option if the user is not yet integrated into eXo Platform. This will integrate the use and his/her memberships and groups. - **UPDATED** - Select this option if the user field or membership has been modified/added/deleted. This will update the user profiles integrated into eXo Platform and all related memberships. - **DELETED** - Select this option if the user has been deleted from the organization data source. So the user profile will be deleted from eXo Platform.

- */portal/rest/management/orgsync/syncAllUser?eventType=EVENT*

Para-value	Description
EVENT	Select one of the following values: - **ADDED** - Search for users who are added to the Organization data source, but not yet integrated. Those users will then be synchronized.

Para-value	Description
	- **UPDATED** - Search for users that are present in the Organization data source and already integrated into eXo Platform. Those profiles will be updated. - **DELETED** - Search for users that are deleted from the Organization data source, but their profiles are still always existing in eXo Platform. Those profiles are then deleted.

Also, you can request for synchronizing all users.

- */portal/rest/management/orgsync/syncGroup?groupId=GROUP*ID&eventType=EVENT

Para-value	Description
GROUP*ID*	Select a groupId, such as /platform/users.
EVENT	Select one of the following values: - **ADDED** - Select this option if the group is not yet integrated into eXo Platform. This will integrate the group. - **UPDATED** - This option is not used any longer. - **DELETED** - Select this option if the group has been deleted from Organization data source. The group profile is then deleted from eXo Platform.

- */portal/rest/management/orgsync/syncAllGroups?eventType=EVENT*

Para-value	Description
EVENT	Select one of the following values: - **ADDED** - Search for groups that are added to the Organization data source, but not yet integrated into eXo Platform. Those groups will be integrated.

Para-value	Description
	- **UPDATED**: This option is not used any longer. - **DELETED** - Search for groups that are deleted from the Organization data source, but their profiles are still existed in eXo Platform. Those profiles are then deleted.

You can also request for synchronizing all groups.

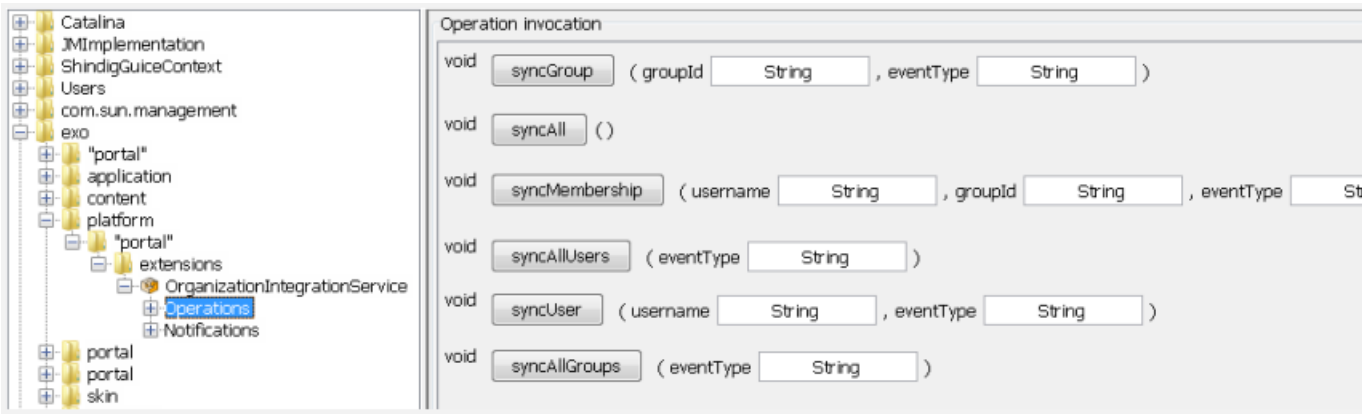
- */portal/rest/management/orgsync/syncMembership?groupId=GROUP*ID&username=USERNAME&eventType=EVENT

Para-value	Description
GROUP*ID*	Select a groupId, such as /platform/users.
USERNAME	Select a username.
EVENT	Select one of the following values: - **ADDED** - Search for memberships that are added to the Organization data source, but not yet integrated into eXo Platform. Those memberships will be integrated. - **UPDATED** - This option is not used any longer. - **DELETED** - Search for memberships that are deleted from the Organization data source. This will synchronize user's memberships related to the selected group.

- */portal/rest/management/orgsync/syncAll*: This will synchronize all groups and users.

Use JMX

To access the Integration Service features via JMX, you can use JMX-compliant monitoring tools, such as *JConsole* [http://download.oracle.com/javase/6/docs/technotes/guides/management/jconsole.html]. The name of the MBean is: exo:portal="portal",service=extensions,name=OrganizationIntegrationService,type=platform.


Catalina
JMImplementation
ShindigGuiceContext
Users
com.sun.management
exo
"portal"
application
content
platform
"portal"
extensions
OrganizationIntegrationService
Operations
Notifications
portal
portal
skin
Operation invocation
void
syncGroup
(groupId
String
, eventType
String
)
void
syncAll
( )
void
syncMembership
(username
String
, groupId
String
, eventType
void
syncAllUsers
(eventType
String
)
void
syncUser
(username
String
, eventType
String
)
void
syncAllGroups
(eventType
String
)

Legacy Organization Models

This chapter covers the following topics:

Legacy organization configurations

If you used one of those Organization Data Models.

- LDAP Organization Service
- Active Directory
- Hibernate

in a previous version of eXo Platform, you will be able to keep working with them on eXo Platform 3.5.

Follow those steps to configure Legacy Organization Services: 1. Add a new file in *exo.conf.dir.name*/portal/portal/configuration.xml

```
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://www.exoplaform.org/xml/ns/kernel_1_2.xsd     http://
www.exoplaform.org/xml/ns/kernel_1_2.xsd"
    xmlns="http://www.exoplaform.org/xml/ns/kernel_1_2.xsd">

 <import>legacy-organization-configuration.xml</import>

 <!-- Remove unnecessary Picket LINK Services -->
                                                                                <remove-
configuration>org.exoplatform.services.organization.idm.PicketLinkIDMCacheService</remove-
configuration>
        <remove-configuration>org.exoplatform.services.organization.idm.PicketLinkIDMService</
remove-configuration>
</configuration>
```

2. Add a new file in *exo.conf.dir.name*/portal/portal/legacy-organization-configuration.xml with one of the above contents.

The Hibernate Organization Service configuration

```
<configuration
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://www.exoplaform.org/xml/ns/kernel_1_2.xsd       http://
www.exoplaform.org/xml/ns/kernel_1_2.xsd"
  xmlns="http://www.exoplaform.org/xml/ns/kernel_1_2.xsd">
 <component>
   <key>org.exoplatform.services.organization.OrganizationService</key>
   <type>org.exoplatform.services.organization.hibernate.OrganizationServiceImpl</type>
 </component>
 <external-component-plugins>
  <target-component>org.exoplatform.services.database.HibernateService</target-component>
   <component-plugin>
    <name>add.hibernate.mapping</name>
    <set-method>addPlugin</set-method>
    <type>org.exoplatform.services.database.impl.AddHibernateMappingPlugin</type>
    <init-params>
     <values-param>
      <name>hibernate.mapping</name>
      <value>org/exoplatform/services/organization/impl/UserImpl.hbm.xml</value>
      <value>org/exoplatform/services/organization/impl/MembershipImpl.hbm.xml</value>
      <value>org/exoplatform/services/organization/impl/GroupImpl.hbm.xml</value>
      <value>org/exoplatform/services/organization/impl/MembershipTypeImpl.hbm.xml</value>
      <value>org/exoplatform/services/organization/impl/UserProfileData.hbm.xml</value>
     </values-param>
    </init-params>
   </component-plugin>
 </external-component-plugins>
 <import>classpath:/conf/portal/organization-configuration.xml</import>
</configuration>
```

The LDAP Organization Service configuration

```
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://
www.exoplaform.org/xml/ns/kernel_1_2.xsd" xsi:schemaLocation="http://www.exoplaform.org/
xml/ns/kernel_1_2.xsd http://www.exoplaform.org/xml/ns/kernel_1_2.xsd">
```

```
 <component>
  <key>org.exoplatform.services.ldap.LDAPService</key>
  <type>org.exoplatform.services.ldap.impl.LDAPServiceImpl</type>
  <init-params>
   <object-param>
    <name>ldap.config</name>
    <description>Default ldap config</description>
    <object type="org.exoplatform.services.ldap.impl.LDAPConnectionConfig">
     <field name="providerURL"><string>ldap://127.0.0.1:389,10.0.0.1:389</string></field>
     <field name="rootdn"><string>CN=Manager,DC=exoplatform,DC=org</string></field>
     <field name="password"><string>secret</string></field>
     <field name="version"><string>3</string></field>
     <field name="minConnection"><int>5</int></field>
     <field name="maxConnection"><int>10</int></field>
     <field name="referralMode"><string>follow</string></field>
     <field name="serverName"><string>default</string></field>
    </object>
   </object-param>
  </init-params>
 </component>
 <component>
  <key>org.exoplatform.services.organization.OrganizationService</key>
  <type>org.exoplatform.services.organization.ldap.OrganizationServiceImpl</type>
  <component-plugins>
   <component-plugin>
    <name>init.service.listener</name>
    <set-method>addListenerPlugin</set-method>
    <type>org.exoplatform.services.organization.ldap.OrganizationLdapInitializer</type>
   <description>this listener populate organization ldap service create default dn</description>
   </component-plugin>
  </component-plugins>
  <init-params>
   <value-param>
    <name>ldap.userDN.key</name>
    <description>The key used to compose user DN</description>
    <value>cn</value>
   </value-param>
   <object-param>
    <name>ldap.attribute.mapping</name>
    <description>ldap attribute mapping</description>
    <object type="org.exoplatform.services.organization.ldap.LDAPAttributeMapping">
      <field name="userLDAPClasses"><string>top,person,organizationalPerson,inetOrgPerson</
string></field>
     <field name="profileLDAPClasses"><string>top,organizationalPerson</string></field>
```

```
    <field name="groupLDAPClasses"><string>top,organizationalUnit</string></field>
        <field  name="membershipTypeLDAPClasses"><string>top,organizationalRole</string></
field>
    <field name="membershipLDAPClasses"><string>top,groupOfNames</string></field>
    <field name="baseURL"><string>dc=exoplatform,dc=org</string></field>
      <field  name="groupsURL"><string>ou=groups,ou=portal,dc=exoplatform,dc=org</string></
field>
                                                                                <field

string></field>
    <field name="userURL"><string>ou=users,ou=portal,dc=exoplatform,dc=org</string></field>
      <field  name="profileURL"><string>ou=profiles,ou=portal,dc=exoplatform,dc=org</string></
field>
    <field name="userUsernameAttr"><string>uid</string></field>
    <field name="userPassword"><string>userPassword</string></field>
    <field name="userFirstNameAttr"><string>givenName</string></field>
    <field name="userLastNameAttr"><string>sn</string></field>
    <field name="userDisplayNameAttr"><string>displayName</string></field>
    <field name="userMailAttr"><string>mail</string></field>
    <field name="userObjectClassFilter"><string>objectClass=person</string></field>
    <field name="membershipTypeMemberValue"><string>member</string></field>
    <field name="membershipTypeRoleNameAttr"><string>cn</string></field>
    <field name="membershipTypeNameAttr"><string>cn</string></field>
     <field name="membershipTypeObjectClassFilter"><string>objectClass=organizationalRole</
string></field>
    <field name="membershiptypeObjectClass"><string>organizationalRole</string></field>
    <field name="groupObjectClass"><string>organizationalUnit</string></field>
        <field  name="groupObjectClassFilter"><string>objectClass=organizationalUnit</string></
field>
    <field name="membershipObjectClass"><string>groupOfNames</string></field>
    <field name="membershipObjectClassFilter"><string>objectClass=groupOfNames</string></
field>
    <field name="ldapCreatedTimeStampAttr"><string>createdTimeStamp</string></field>
    <field name="ldapModifiedTimeStampAttr"><string>modifiedTimeStamp</string></field>
    <field name="ldapDescriptionAttr"><string>description</string></field>
   </object>
  </object-param>
 </init-params>
</component>

<external-component-plugins>
 <target-component>org.exoplatform.services.database.HibernateService</target-component>
 <component-plugin>
  <name>add.hibernate.mapping</name>
```

```
  <set-method>addPlugin</set-method>
  <type>org.exoplatform.services.database.impl.AddHibernateMappingPlugin</type>
  <init-params>
   <values-param>
    <name>hibernate.mapping</name>
    <value>org/exoplatform/services/organization/impl/UserProfileData.hbm.xml</value>
   </values-param>
  </init-params>
 </component-plugin>
</external-component-plugins>
</configuration>
```

The AD Organization Service configuration

```
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://
www.exoplaform.org/xml/ns/kernel_1_2.xsd" xsi:schemaLocation="http://www.exoplaform.org/
xml/ns/kernel_1_2.xsd http://www.exoplaform.org/xml/ns/kernel_1_2.xsd">
 <component>
  <key>org.exoplatform.services.ldap.LDAPService</key>
  <type>org.exoplatform.services.ldap.impl.LDAPServiceImpl</type>
  <init-params>
   <object-param>
    <name>ldap.config</name>
    <description>Default ldap config</description>
    <object type="org.exoplatform.services.ldap.impl.LDAPConnectionConfig">
     <field name="providerURL"><string>ldap://192.168.2.88:389</string></field>
      <field name="rootdn"><string>CN=Administrator,CN=Users, DC=exoplatform,DC=org</
string></field>
     <field name="password"><string>Secret1234</string></field>
     <field name="version"><string>3</string></field>
     <field name="minConnection"><int>5</int></field>
     <field name="maxConnection"><int>10</int></field>
     <field name="referralMode"><string>ignore</string></field>
     <field name="serverName"><string>active.directory</string></field>
    </object>
   </object-param>
  </init-params>
 </component>
 <component>
  <key>org.exoplatform.services.organization.OrganizationService</key>
  <type>org.exoplatform.services.organization.ldap.OrganizationServiceImpl</type>
```

```
  <component-plugins>
    <component-plugin>
      <name>init.service.listener</name>
      <set-method>addListenerPlugin</set-method>
      <type>org.exoplatform.services.organization.ldap.OrganizationLdapInitializer</type>
     <description>this listener populate organization ldap service create default dn</description>
    </component-plugin>
  </component-plugins>
  <init-params>
    <object-param>
      <name>ldap.attribute.mapping</name>
      <description>ldap attribute mapping</description>
      <object type="org.exoplatform.services.organization.ldap.LDAPAttributeMapping">
       <field name="userLDAPClasses"><string>top,person,organizationalPerson,user</string></
field>
        <field name="profileLDAPClasses"><string>top,organizationalPerson</string></field>
        <field name="groupLDAPClasses"><string>top,organizationalUnit</string></field>
        <field name="membershipTypeLDAPClasses"><string>top,group</string></field>
        <field name="membershipLDAPClasses"><string>top,group</string></field>
        <field name="baseURL"><string>DC=test,DC=man</string></field>
       <field name="groupsURL"><string>ou=groups,ou=portal,DC=test,DC=man</string></field>
  <field name="membershipTypeURL"><string>ou=memberships,ou=portal,DC=test,DC=man</
string></field>
        <field name="userURL"><string>ou=users,ou=portal,DC=test,DC=man</string></field>
       <field name="profileURL"><string>ou=profiles,ou=portal,DC=test,DC=man</string></field>
        <field name="userUsernameAttr"><string>sAMAccountName</string></field>
        <field name="userPassword"><string>unicodePwd</string></field>
        <field name="userFirstNameAttr"><string>givenName</string></field>
        <field name="userLastNameAttr"><string>sn</string></field>
        <field name="userDisplayNameAttr"><string>displayName</string></field>
        <field name="userMailAttr"><string>mail</string></field>
        <field name="userObjectClassFilter"><string>objectClass=user</string></field>
        <field name="membershipTypeMemberValue"><string>member</string></field>
        <field name="membershipTypeRoleNameAttr"><string>cn</string></field>
        <field name="membershipTypeNameAttr"><string>cn</string></field>
          <field name="membershipTypeObjectClassFilter"><string>objectClass=group</string></
field>
        <field name="membershiptypeObjectClass"><string>group</string></field>
        <field name="groupObjectClass"><string>organizationalUnit</string></field>
         <field name="groupObjectClassFilter"><string>objectClass=organizationalUnit</string></
field>
        <field name="membershipObjectClass"><string>group</string></field>
        <field name="membershipObjectClassFilter"><string>objectClass=group</string></field>
        <field name="ldapCreatedTimeStampAttr"><string>createdTimeStamp</string></field>
```

```
        <field name="ldapModifiedTimeStampAttr"><string>modifiedTimeStamp</string></field>
        <field name="ldapDescriptionAttr"><string>description</string></field>
      </object>
    </object-param>
  </init-params>
 </component>
 <external-component-plugins>
  <target-component>org.exoplatform.services.database.HibernateService</target-component>
  <component-plugin>
    <name>add.hibernate.mapping</name>
    <set-method>addPlugin</set-method>
    <type>org.exoplatform.services.database.impl.AddHibernateMappingPlugin</type>
    <init-params>
      <values-param>
        <name>hibernate.mapping</name>
        <value>org/exoplatform/services/organization/impl/UserProfileData.hbm.xml</value>
      </values-param>
    </init-params>
  </component-plugin>
 </external-component-plugins>
</configuration>
```