

eXo Platform 3.0 Administrators Guide

eXo Platform ()

Copyright © 2010

1. Introduction	1
1.1. Welcome to eXo Platform	1
1.2. Who should read this guide?	1
2. Installation	2
2.1. Install the Tomcat bundle	2
2.1.1. Startup	2
2.1.2. Shutdown	2
2.1.3. Startup scripts	3
2.2. Install the JBoss EARs	4
2.3. eXo Profiles	6
3. Configuration	7
3.1. eXo Platform Configuration	7
3.1.1. Portal Containers, Customization and Configurations	7
3.1.2. configuration.properties	8
3.1.3. configuration.xml	8
3.1.4. portal/portal/configuration.xml	8
3.2. Database Configuration	8
3.2.1. Connect to a production database	9
3.2.2. Change the datasources names	12
3.2.3. Database configuration FAQ	13
3.3. FileSystem paths	14
3.4. Mail Server	15
3.5. WebDAV Cache Control	15
3.6. Chat Server	16
3.6.1. XMPPMessenger	16
3.6.2. Chat server configuration	16
3.7. Office server	18
3.8. Log-in	19
3.9. JCR	19
3.10. Users configurations	20
3.10.1. Super User configuration	20
3.10.2. eXo Platform default users list definition	20
3.11. How to call ClearOrphanSymlinksJob	20
4. Management	22
4.1. Introduction to eXo Platform Management	22
4.1.1. JMX interface	22
4.1.2. REST interface	22
4.2. Management views of eXo Platform	22
4.2.1. PortalContainer management view	22
4.2.2. Cache management view	23
4.2.3. eXo Content management view	26
4.2.4. JCR management view	28
4.2.5. Portal management view	30
4.2.6. eXo Knowledge management view	34
4.2.7. eXo Collaboration management view	38
5. Security	39
5.1.	39
5.1.1. Change the JAAS realm	39
5.1.2. Update the password encryption key of the RememberMe token	41
6. Backup	42
6.1. How to back up eXo Platform	42
6.1.1. Plan backup	42

6.1.2. Perform backup	45
6.1.3. Perform restore	45
6.1.4. Third party tools	46
7. Clustering	47
7.1. About Platform clustering	47
7.1.1. When should you consider clustering?	47
7.1.2. Shared file system	47
7.2. Cluster setup	47
7.2.1. Advanced configuration	48
7.3. Cluster management	48
7.3.1. Cluster profile	48
7.3.2. Initial Startup	48
7.3.3. Startup and shutdown	48
7.4. Clustering FAQ	49
7.4.1. How to migrate from local to cluster mode?	49
7.4.2. Why is startup failed with a <i>Port value out of range</i> error?	49
7.4.3. How to solve the "failed sending message to null" error?	49
8. Deployment	50
8.1. How to remove the sample applications	50
8.1.1. Remove Acme Website	50
8.1.2. Remove Acme Social Intranet	50
8.1.3. Remove the docs webapp	50
8.1.4. Remove crash	51
8.2. How to deploy a custom extension	51
8.3. How to setup an Apache Frontend	51
8.3.1. Base configuration for apache	51
8.3.2. Connection via HTTP protocol (Apache mod_proxy)	52
8.3.3. Connection via AJP protocol	52
8.4. How to configure the session-timeout for the web server	54
8.4.1. Server Tomcat	54
8.4.2. Sever Jboss	55
9. How to upgrade Platform	56
9.1. Purpose	56
9.2. Modifications details between versions	56
9.2.1. Modifications between Platform v.3.0.1 and 3.0.2	56
9.2.2. Modifications between Platform v.3.0.2 to 3.0.3	56
9.2.3. Modifications between Platform v.3.0.3 and 3.0.4	57
9.3. Steps to upgrade	57
9.3.1. Development mode	57
9.3.2. Production mode	58

Chapter 1. Introduction

1.1. Welcome to eXo Platform

eXo Platform 3.0, the user experience platform for Java, is comprised of Core and Extended Services.

- **Core Services**

- *GateIn Portal*: a powerful framework for developing portlets and other web-based user interfaces
- *eXo Content*: extends portal-based applications with Enterprise Content Management (ECM) capabilities
 - *eXo WCM*: web content management services
 - *xCMIS*: an implementation of the full stack of Java-based CMIS (Content Management Interoperability Specification) services on top of eXo WCM
 - *eXo Workflow*: integrated BPM (business process management) capabilities
- *eXo IDE*: an intuitive web-based development environment that allows developers to build, test and deploy client applications (such as gadgets and mashups) and REST-ful services online
- *CRaSH*: enables easy browsing of JCR trees, and serves as a shell for executing JCR operations

- **Extended Services**

- *eXo Social*: a framework for building gadgets that can display and mash-up activity information for contacts, social networks, applications and services
- *eXo Collaboration*: easily add Mail, Chat, Calendar and Address Book services to portal-based web applications
- *eXo Knowledge*: adds Forum, Answers and FAQ functionality to portal-based apps, for collecting, organizing and sharing user knowledge

eXo Platform is a fully supported and commercially licensed product based on eXo open source projects. Designed for enterprise use, it has been packaged and tested to optimize production readiness and administration. eXo Platform runs on JBoss, Spring, Tomcat, WebSphere, and other Java applications servers, and can be used with most relational database systems, including MySQL and Oracle.

1.2. Who should read this guide?

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

- *System Administrators* who want to use, deploy and manage eXo Platform system in their enterprise.
- *Developers* who want to know how to leverage eXo Platform in their customer projects.

This document will guide you through the most important tasks for eXo Platform administration and management. At the end, you will be able to install, configure, migrate, and manage your eXo Platform system.

Chapter 2. Installation

eXo Platform is packaged as a deployable enterprise archive defined by the Java EE specification, and as a configuration directory.

2.1. Install the Tomcat bundle

The easiest way to install eXo Platform is to take the default bundle. This is a ready-made package on top of Tomcat 6 application server, so you simply need to copy the `bin/tomcat6-bundle/` directory to your server.

2.1.1. Startup

eXo Platform leverages the application server on which it is deployed. This means, to start and stop eXo Platform, 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 has started when you see the following message in your log/console:

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

2.1.2. Shutdown

- On Linux and OS X:

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

- On Windows:

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

If you have a similar message when you try to stop Tomcat:

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

then you must stop the tomcat process by a Ctrl+C or kill 9 command. To perform a kill automatically, you can type:

```
stop_eXo.sh -force
```

instead. It is available only on Linux and OS X systems.

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

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

2.1.3. Startup scripts

eXo comes with several builtin startup scripts:

- `start_eXo.sh`: start eXo on Linux and OS X
- `start_eXo.bat`: start eXo on Windows
- `bin/gatein-dev.sh`: start eXo on Linux and OS X in developer mode
- `bin/gatein-dev.bat`: start eXo on Windows in developer mode

2.1.3.1. Normal Mode

`start_eXo` scripts launch eXo 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
```

<code>-Xms</code>	Minimal heap size (defaults to 256 MB)
<code>-Xmx</code>	Maximal Heap Size of (defaults to 1 GB)
<code>-Djava.security.auth.login.config</code>	path to the JAAS security file where the security domains are and JAAS authentication modules are declared
<code>-Dexo.conf.dir.name</code>	path where eXo will start looking at <code>configuration.properties</code> and <code>configuration.xml</code>
<code>-Dexo.profiles</code>	the list of comma-separated exo profiles to activate

This is enough to start and run a demo, but you will need to adjust these values for a production setup.

2.1.3.2. Developer mode

`gatein-dev` scripts launch eXo in developer mode with a few more 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
```

<code>-Dcom.sun.management.jmxremote</code>	activates the JMX remoting
<code>-Xdebug</code> <code>-Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n</code>	enables remote debugging
<code>-Dorg.exoplatform.container.configuration.debug</code>	the container will log to the console what xml files it loads
<code>-Dexo.product.developing=true</code>	deactivates javascript and css merging for easier debugging

2.2. Install the JBoss EARs

We provide EARs packages to deploy in your existing JBoss application server. They are located in the folder `bin/jboss5-eap-ears/`.

To install eXo Platform on JBoss, follow these steps:

1. Copy files in `jboss-root/server/default/deploy`

- `gatein-ds.xml`
- `gatein.ear` (it must remain a folder named `gatein.ear`)
- `starter-gatein.ear`
- `acme-website.ear`
- `office-portal.ear`
- `platform-extension.ear`
- `exo-collaboration.ear`
- `exo-social-extension.ear`
- `gatein-exo-ks.ear`
- `gatein-wcm-extension-plf.ear`
- `gatein-workflow-extension-plf.ear`

2. Create a folder `jboss-root/server/default/conf/gatein`

Copy these files:

- `configuration.properties`
- `configuration.xml`

3. Copy these files in `jboss-root/bin`

- `exokey.pem`

- oauthkey.pem

4. Configure JVM parameters

On **Linux**, add these lines at 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"
EXO_XML="-Djavax.xml.stream.XMLOutputFactory=com.sun.xml.stream.ZephyrWriterFactory -Djavax.xml.stream.XMLInputFactory=com.sun.xml.stream.ZephyrReaderFactory"
JAVA_OPTS="$JAVA_OPTS $EXO_OPTS $EXO_PROFILES $EXO_XML"
```

On **Windows**, add these lines at 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"
set "EXO_XML=-Djavax.xml.stream.XMLOutputFactory=com.sun.xml.stream.ZephyrWriterFactory -Djavax.xml.stream.XMLInputFactory=com.sun.xml.stream.ZephyrReaderFactory"
set "JAVA_OPTS=%JAVA_OPTS% %EXO_OPTS% %EXO_PROFILES% %EXO_XML%"
```

Adapt to your needs:

- You can use another implementation of SAX by changing the class names in the `EXO_XML` variable, for example: `com.sun.xml.internal.stream.XMLOutputFactoryImpl`
- To debug the application, simply add `$REMOTE_DEBUG` in the `JAVA_OPTS` variable

5. Add eXo logging categories in `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

- On Linux and OS X:

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

- On Windows:

```
%JBOSS_HOME%binrun.bat
```

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

```
INFO [org.jboss.bootstrap.microcontainer.ServerImpl] (main) JBoss (Microcontainer) [5.0.1 (build: ...)] Started
```


7. Shut down

- On Linux and OS X:

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

- On Windows:

```
%JBOSS_HOME%binshutdown.bat
```

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

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

2.3. eXo Profiles

eXo comes with different runtime profile, that you can use if you want to customize what modules you want to enable/disable in you eXo instance.

`start_eXo` commands accept a comma-separated list of profiles. The following profiles are supported :

Profile	Description
collaboration	enables eXo Collaboration module
knowledge	enables eXo Knowledge module
social	enables eXo Social module
workflow	enables the Workflow add-on within the eXo Content module

Additionally, you can use these composite profiles:

Profile	Description
minimal	contains GateIn + WCM
default	contains all except workflow (gatein,ide,wcm,collaboration,social,knowledge)
all	all available modules

For example:

```
./start_eXo.sh default,workflow # start all modules including workflow
./start_eXo.sh collaboration,knowledge # start exo with gatein,wcm, collaboration and knowledge enabled
./start_eXo.sh minimal,social # start with social, gatein and wcm
```

Chapter 3. Configuration

3.1. 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.

3.1.1. Portal Containers, Customization and Configurations

This section will explain some parts of the eXo internals in so that you will understand the roles of these configuration files.

eXo Platform kernel groups runtime components in *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 for the samples is http://localhost:800/*portal*.

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

But 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 exact 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 xml configuration files like `configuration.xml`. Such files exists in jars, wars and below `exo.conf.dir`.

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

Additionally, xml files may contain variables that are populated via properties defined in

configuration.properties. Hence, the configuration.properties serves as exposing some chosen variables that are necessary to configure eXo Platform in a server environment.

3.1.2. configuration.properties

The system configuration is mostly done in configuration.properties. In most cases, this should be the only file that a system administrator will need to configure.

In the Tomcat bundle, this file is located at /gatein/conf/configuration.properties

3.1.3. 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 your project does not want to use "portal" as the default portal, this file can be used to to import another *PortalContainerDefinition* into the root container.

Note

Details on how to configure a new portal container are out of the scope of this guide but is extensively covered in the kernel reference guide.

3.1.4. 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. But this file is the last loaded by the kernel. It has a higher priority over any other configuration file, including extensions. So it let's you override any internal component configuration.

This may turn very handy for services or configurations that are not exposed in configuration.properties, but you'd like to tune anyway.

For example, you could decide to change the default transaction timeout to 2 minutes with this piece of xml:

```
<component>
  <key>org.exoplatform.services.transaction.TransactionService</key>
  <type>org.exoplatform.services.transaction.jboss.cache.JBossTransactionsService</type>
  <init-params>
    <value-param>
      <name>timeout</name>
      <value>120</value>
    </value-param>
  </init-params>
</component>
```

3.2. Database Configuration

eXo Platform relies on the application server for its database access, so the database must be configured as a datasource at the AS level. That datasource is obtained by accessing the enterprise naming context (ENC) through the Java Naming and Directory Interface (JNDI) service.

3.2.1. Connect to a production database

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

Note

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

3.2.1.1. Prepare your database server

You need to prepare two database schema and a technical user to access them.

1. Connect to your database server using ssh :

```
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
```

4. You are prompted for the password

5. Create 2 databases : one for idm (*\$dbname-idm*) and the other for jcr *\$dbname-jcr*)

6. To create one :

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

- Configure a user that has rights it access it remotely (not only from the host server) :

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

- \$IP = AS hostname
- \$IP = IP with wildcard (eg 192.168.1.% = all IPs on 192.168.1.x network)
- \$username = username that eXo Platform will connect with (i.e. 'dbnameuser')

- b. Create the second DataBase

7. Verify that both DataBases were created :

```
•
show databases;
```

8. Quit the server with

```
exit
```

Note

eXo Platform does not require tables to be created before it starts because it is capable of doing it automatically on the first startup. If you prefer to run a DDL script to create the database objects, please contact eXo Support to obtain the script for your database.

3.2.1.2. Configure eXo Platform

Now that the database is ready, you need to configure eXo to be able to connect to it. The steps depend on the application server. We provide instructions for Tomcat and JBoss

3.2.1.2.1. Tomcat bundle

Configuring a datasource for eXo under Tomcat involves two steps:

- Edit gatein-ds.xml
- Add the JDBC driver

3.2.1.2.1.1. Edit server.xml

Edit `$TOMCATHOME/conf/server.xml`

To declare the binding of the datasources in the GlobalNaming context:

1. Change the driver :

```
org.hsqldb.jdbcDriver
```

to

```
com.mysql.jdbc.Driver
```

2. Change the username and password to the values set above

3. Change the url to access your DataBase :

```
"jdbc:hsqldb:file:../gatein/data/hsqldb/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" name="jdbc/mysql" type="javax.sql.DataSource"/>
```

```
<!-- eXo IDM Datasource for portal -->
<Resource auth="Container" driverClassName="com.mysql.jdbc.Driver" logAbandoned="true" maxActive="20" maxIdle="10" name="jdbc/portal" password="" type="javax.sql.DataSource" url="jdbc:mysql://localhost:3306/exo-xxx_portal-localDB" username="" />
```

3.2.1.2.1.2. Add the JDBC driver

You need to add the MySQL connector library in Tomcat. Add **mysql-connector-java-5.1.x.jar** to **\$TOMCATHOME/lib/**

Tip

You can get the latest MySQL connector from : <http://dev.mysql.com/downloads/connector/j/>

3.2.1.2.1.3. JBoss

Configuring a datasource for eXo under JBoss involves two steps:

- Edit server.xml
- Add the JDBC driver

3.2.1.2.1.4. Edit gatein-ds.xml

Open **\$JBOSSHOME/server/default/deploy/gatein-ds.xml**

To declare the binding of the datasources in the GlobalNaming context:

1. Change the driver:

```
org.hsqldb.jdbcDriver
```

to

```
com.mysql.jdbc.Driver
```

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

3. 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>16</min-pool-size>
    <max-pool-size>128</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>16</min-pool-size>
    <max-pool-size>128</max-pool-size>
    <idle-timeout-minutes>0</idle-timeout-minutes>
    <prepared-statement-cache-size>32</prepared-statement-cache-size>
  </no-tx-datasource>
</datasources>

```

3.2.1.2.1.5. Add the JDBC driver

You need to add the MySQL connector library in JBoss. Add **mysql-connector-java-5.1.x.jar** to **\$JBOSSHOME/server/default/lib/**

Tip

You can get the latest MySQL connector from : <http://dev.mysql.com/downloads/connector/j/>

3.2.2. Change the datasources names

By default, eXo Platform defines two datasources:

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

You may want to give them a different name. To do this, do as follows :

- Edit `configuration.properties`
- Change the datasource name in the application server

3.2.2.1. Edit configuration.properties

```
$TOMCAT_HOME/gatein/conf/configuration.properties
```

Indicate to eXo the name of the datasources.

```

# 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

```

Caution

eXo will automatically append the portal container name (*"portal"* by default) to these values before it performs a JNDI lookup.

3.2.2.2. Change the datasource name in the application server

Now, you need to change the name under which the datasources are bound in the JNDI tree by the app server. This is application sever dependent.

3.2.2.2.1. Tomcat bundle

In Tomcat, the datasources configuration requires to edit two files :

- server.xml
- starter.xml

Tip

Please refer to Tomcat's [JNDI Resources How To](#) for more details on JNDI resources binding in Tomcat.

3.2.2.2.2. server.xml

```
$TOMCAT_HOME/conf/server.xml
```

Declare the binding of the datasources in the GlobalNaming context:

```
<!-- eXo JCR Datasource for portal -->
<Resource auth="Container" driverClassName="org.hsqldb.jdbcDriver" logAbandoned="true" maxActive="128" maxIdle=

<!-- eXo IDM Datasource for portal -->
<Resource auth="Container" driverClassName="org.hsqldb.jdbcDriver" logAbandoned="true" maxActive="128" maxIdle=
```

3.2.2.2.3. starter.xml

```
$TOMCAT_HOME/conf/Catalina/localhost/starter.xml
```

We declare resource links that make our datasources accessible to the starter webapp which is used when starting eXo.

```
<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"/>
```

3.2.3. Database configuration FAQ

3.2.3.1. How to remove idle MySQL connections

Some RDBMS, like MySQL, close idle connections after a while (8h by default on MySQL). Thus, a connection from the pool will be invalid and of course any application SQL command will fail, resulting in errors like:

```
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, a solution is to use DBCP to monitor 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"
```

- **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 in the datasource configuration file of your application server (i.e. conf/server.xml on Tomcat).

For more details about 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-database-connection>
- <http://confluence.atlassian.com/display/JIRA/Surviving+Connection+Closures>

3.3. FileSystem paths

eXo needs read/write access to several paths in the local filesystem.

```
gatein.data.dir=../gatein/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 what path. The `temporary` column indicates if the data is temporary

or persistent.

variable	content	temporary
gatein.data.dir	jta transactional data	yes
gatein.jcr.data.dir	jcr swap data	yes
gatein.jcr.storage.data.dir	binary value storage for jcr	no
gatein.jcr.index.data.dir	lucene index for JCR	no

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

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

3.4. Mail Server

eXo Platform requires an SMTP server in order to send emails such as notifications or password reminders.

```
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
```

More details can be found in the [JavaMail API documentation](#).

For KS, 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 authenticated SMTP systems:

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

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

3.5. WebDAV Cache Control

The embedded WebDAV server lets you control the cache-control http header that it transmits to clients by mimetype. 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 list of key=pair values separated by a semicolon, where keys are a list of mimetypes followed by the cache-control value to set.

3.6. Chat Server

3.6.1. XMPPMessenger

If you changed the hostname and port for the chat server, then you'll 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
```

3.6.2. Chat server configuration

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

Configuration is based on properties expressed in an XML syntax. For example, to set property `prop.name.is.blah=value`, 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](#)

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 provider.authorizeduser.name	password
eXoAuthProvider		
authenticationURL	URL to authenticate users	/organization/authenticate/

Property	Description	Default value
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/
usersCountMethod	HTTP method for user/count	GET
userInfoURL	URL to get user info	/organization/xml/user/info/
userInfoMethod	HTTP method for user/info	GET
eXoGroupProvider		
groupInfoURL	URL to get group info	/organization/xml/group/info/
groupInfoMethod	HTTP method for info	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

3.6.2.1. Ports

In order to run properly the chat server needs several ports to 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 <code>exo.chat.port</code> property.
9090 && 9091	Admin Console (http)	The port used for respectively the unsecured and secured Openfire

Port	Type	Description
		Admin Console access.
3478 & 3479	STUN service	The port used for the service that ensures connectivity between entities when behind a NAT.

3.7. Office server

eXo Platform allows users to view various types of documents directly in the **Content Explorer** through the office server. To do so, the office application must be available in your local device first.

Then, **JODConverter** will use the [default values](#) below to start the office server.

Default values are customized by adding parameters below in the *configuration.properties* file:

```
# JODConverter 3.0
#wcm.jodconverter.portnumbers=8100, 8101, 8102, 8103, 8104
#wcm.jodconverter.officehome=/usr/lib/libreoffice
#wcm.jodconverter.taskqueuetimeout=30000
#wcm.jodconverter.taskexecutiontimeout=120000
#wcm.jodconverter.maxtasksperprocess=200
#wcm.jodconverter.retrytimeout=120000
```

Key	Default values	Description
wcm.jodconverter.portnumbers	2002	List of ports, separated by commas, those used by each JODConverter processing thread. Number of office instances is equal to number of ports.
wcm.jodconverter.officehome	See here	Absolute path of office home on the current local device. It means that office needs to be installed in the local device before using it.
wcm.jodconverter.taskqueuetimeout	30000	Maximum living time of tasks in the conversation queue. Task will be removed from this queue if waiting time is longer than <i>taskQueueTimeout</i> .
wcm.jodconverter.taskexecutiontimeout	120000	If a task has been executing within an soffice process for longer than this period, soffice process will be terminated and restarted.
wcm.jodconverter.maxtasksperprocess	200	To ensure that potential memory leaks do not accumulate and affect performance, the JodConverter will restart an soffice process after a

Key	Default values	Description
		given number of tasks have been performed.
wcm.jodconverter.retrytimeout	120000	Interval time to try to restart the office services after an unexpected crash.

The default office home

- **Linux**

```
"/opt/openoffice.org3"
"/opt/libreoffice"
"/usr/lib/openoffice"
"/usr/lib/libreoffice"
```

- **Windows**

```
"<SYSTEM_PROGRAMFILES>/OpenOffice.org 3"
"<SYSTEM_PROGRAMFILES>/LibreOffice 3"
```

- **Mac**

```
"/Applications/OpenOffice.org.app/Contents"
"/Applications/LibreOffice.app/Contents"
```

3.8. Log-in

Logging in eXo Platform is controlled by the [Java Logging API](#).

By default, logging is configured to:

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

In Tomcat, the logging is configured via the conf/logging.properties file. Please refer to [Tomcat's Logging Documentation](#) for more information on how to adjust this file to your needs.

3.9. JCR

A set of properties control the behaviour of the JCR.

```
# Type of JCR configuration to use. Possible values are :
```

```
# local : local JBC configuration
# cluster : cluster JBC configuration
gatein.jcr.config.type=local

# This is the filter used to notify changes in the jcr index
# in cluster mode, use org.exoplatform.services.jcr.impl.core.query.jbossccache.JBossCacheIndexChangesFilter
gatein.jcr.index.changefilterclass=org.exoplatform.services.jcr.impl.core.query.DefaultChangesFilter

# JCR cache configuration
gatein.jcr.cache.config=classpath:/conf/jcr/jbossccache/${gatein.jcr.config.type}/config.xml

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

# JCR Index configuration
gatein.jcr.index.cache.config=classpath:/conf/jcr/jbossccache/cluster/indexer-config.xml
gatein.jcr.jgroups.config=classpath:/conf/jcr/jbossccache/cluster/udp-mux.xml
```

gatein.jcr.config.type	use cluster if you want to use eXo Platform in cluster mode. Otherwise leave local
gatein.jcr.index.changefilterclass	in cluster mode change it to org.exoplatform.services.jcr.impl.core.query.jbossccache.
gatein.jcr.cache.config	JBoss Cache configuration for the JCR locks
gatein.jcr.index.cache.config	JBoss Cache Configuraiton for the JCR index
gatein.jcr.jgroups.config	JGroups configuration to use for cluster mode

Please refer to the JCR reference guide for the details of configuring these files.

3.10. Users configurations

3.10.1. Super User configuration

In eXo Platform 3, we have defined the user "root" as super admin by default. You could override this configuration, by modifying the system property named `exo.super.user` defined in `configuration.properties`.

3.10.2. eXo Platform default users list definition

In eXo Platform, the default users list (omit Super Admin user) are defined in "Acme WebSite" & "Office Intranet" extensions. So by deleting those extensions, the users "John", "demo", "james" and "mary" will not be created.

3.11. How to call ClearOrphanSymlinksJob

Since ECMS 2.1.6/PLF 3.0.6, the *ClearOrphanSymlinksJob* service is used as a CRON job to periodically clear WCM orphan symlinks that do not point to any target nodes at a specific time or date.

By default, the value of the *ClearOrphanSymlinksCronJob* parameter is "0 0/20 * * * ?". It means that orphan symlinks are cleared every 20 minutes.

In which:

- **Asterisk (*)**: indicates that the CRON expression will match for all values of the field.
- **Slash (/)**: is used to describe increments of ranges (e.g. 0/20 means that every 20 minutes).
- **Question mark (?)**: is used to omit the specification of a value for the day-of-month and day-of-week fields (e.g. every day in a week).

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

Example:

```
# Clear WCM orphan symlinks every 25 minutes
wcm.linkjob.cron.expression=0 0/25 * * * ?
```

Or

```
# 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.

Chapter 4. Management

4.1. 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.

4.1.1. 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.`

4.1.2. 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 login 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>.

4.2. Management views of eXo Platform

4.2.1. 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.

4.2.2. 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
<code>clearCache()</code>	Evict all entries from the cache. This method can be used to force a programmatic flush of the cache.
<code>getName</code>	Return the cache name.
<code>getLiveTime</code>	Return the valid lifetime of an entry in the cache in seconds.
<code>setLiveTime</code>	Set the valid lifetime of an entry in the cache in seconds.
<code>getCacheHit</code>	Return the total number of successful hits.
<code>getCacheMiss</code>	Return the total number of unsuccessful hits.
<code>getMaxSize</code>	Return the maximum capacity of the cache.
<code>setMaxSize</code>	Set the maximum capacity of the cache.
<code>getCacheSize</code>	Return the number of entries in the cache.

4.2.2.1. Cache instances

- Portal

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

- JCR

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

- eXo Content

Cache Name	Description
<code>org.exoplatform.ecm.REST.viewer.PDFViewerRESTService</code>	Cache data of PDF files by the <code>ObjectKey</code> object.

Cache Name	Description
<code>org.exoplatform.services.cms.drives.ManageDriveCache</code>	Cache all drives of Sites Explorer by the drive group name (String constant).
<code>org.exoplatform.services.cms.scripts.impl.ScriptCache</code>	Cache all Groovy script files by the script name.
<code>org.exoplatform.services.cms.templates.TemplateCache</code>	Cache all Groovy templates by the mechanism of <code>org.exoplatform.groovyscript.text.TemplateService</code> .
<code>org.exoplatform.services.wcm.webcontent.InitialContentArtifactCache</code>	Cache all artifact data by <code>sourcePath</code> of <code>deploymentDescriptor</code> . These data are reused when a new portal is deployed.
<code>wcm.composer</code>	Cache published contents by the hash generated from path, version, <code>remoteUser</code> , language, recursive, <code>orderBy</code> , <code>orderType</code> , <code>primaryType</code> of cached nodes in eXo Content.

- Social

Cache Name	Description
<code>org.exoplatform.social.core.manager.ActivityManagerCacheById</code>	Cache an activity by its id.
<code>org.exoplatform.social.core.manager.ActivityManagerCacheByIdentityId</code>	Cache the list of all activities by <code>identityId</code> and their segments.
<code>org.exoplatform.social.core.manager.ActivityManagerCacheComments</code>	Cache all comments of an activity by its id.
<code>org.exoplatform.social.core.manager.IdentityManagerCacheById</code>	Cache an identity by its <code>globalId</code> .
<code>org.exoplatform.social.core.manager.IdentityManageridentityCacheById</code>	Cache an identity by its uuid. Cache an identity by providerId:remoteId .
<code>org.exoplatform.social.core.manager.IdentityManagerCacheByProvider</code>	Cache the list of identities by <code>identityProvider</code> .
<code>org.exoplatform.social.core.manager.RelationshipManagerCacheById</code>	Cache the relationship by its id.
<code>org.exoplatform.social.core.manager.RelationshipManagerCacheByIdentityId</code>	Cache the list of relationships by <code>identityId</code> .

4.2.2.2. CacheManager

The CacheManager management view enables you to control different caches.

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

Operation	Description
<code>clearCaches()</code>	Force a programmatic flush of all the registered caches.

4.2.2.3. 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
<code>invalidateAll</code>	Invalidate all caches.
<code>invalidate(namespace)</code>	Invalidate a specific cache namespace.

4.2.3. eXo Content management view

eXo Content provides a management view for three following services:

4.2.3.1. 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
<code>Cached</code>	Indicate the cache is used or not.
<code>CachedEntries</code>	The number of nodes in the cache.
<code>UsedLanguages</code>	The list of all languages accessible in the composer.
<code>UsedOrderBy</code>	The list of OrderBy properties accessible in the composer.
<code>UsedPrimaryTypes</code>	The list of primary types accessible in the composer.

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

Operation	Description
	composer.
getUsedOrderBy	Return the list of OrderBy properties accessible in the composer.

4.2.3.2. 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.
Friendlylies	The list of registered Friendly URIs.
ServletName	The name of the servlet referenced in the service.

Operation	Description
addFriendly(friendlyUri, unfriendlyUri)	Add a new Friendly Uri to the list with two parameters: friendlyUri and unfriendlyUri. 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 service 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.
getFriendlylies	Return the list of registered friendly URIs.

4.2.3.3. 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
<code>getPortletExpirationCache</code>	Return the expiration period of portlet cache in seconds.
<code>setPortletExpirationCache(expirationCache)</code>	Set the expiration period of portlet cache by entering the value into the expirationCache field.

Note

WCMComposer, FriendlyService, and WCMSservice can be controlled through the following paths of relevant REST services:

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

4.2.4. JCR management view

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

4.2.4.1. SessionRegistry

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

Attribute	Description
<code>TimeOut</code>	The expiration period of a JCR session.
<code>Size</code>	The number of currently active sessions.

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

4.2.4.2. LockManager

LockManager stores lock objects and is responsible 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	Description
<code>cleanExpiredLocks</code>	Remove all expired JCR locks.
<code>getNumLocks</code>	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 <code>exo:applications</code> , 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.

4.2.4.3. Repository

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

Attribute	Description
Name	The name of the repository container.

Attribute	Description
RegisteredComponentNames	The list of registered component names in the repository.

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

4.2.4.4. 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.

4.2.5. Portal management view

4.2.5.1. 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.

Attribute	Description
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.

4.2.5.2. 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.

4.2.5.3. 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.
getSkinList	Return the list of loaded skins by the skin service.

4.2.5.4. 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.RemindPasswordTokenService .
memory-token	Store temporary tokens into the transient memory, such as

Token Name	Description
	org.exoplatform.web.security.security.TransientTokenService.
getPortalList	Return the list of identifiers of all loaded portals.

4.2.5.5. 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.

4.2.5.6. 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
<code>getAverageTime(applicationId)</code>	Return the average time spent of the specified application.
<code>getExecutionCount(applicationId)</code>	Return the number of times the specified application has been executed.
<code>getMinTime(applicationId)</code>	Return the minimum time spent of the specified application.
<code>getMaxTime(applicationId)</code>	Return the maximum time spent of the specified application.
<code>getSlowestApplications</code>	Return the list of the 10 slowest applications.
<code>getMostExecutedApplications</code>	Return the list of the 10 most executed applications.
<code>getFastestApplications</code>	Return the list of the 10 fastest applications.
<code>getApplicationList</code>	Return the list of application identifiers classified in the alphabetic order.

4.2.6. eXo Knowledge management view

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

4.2.6.1. 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
<code>AdminRules</code>	The list of rules defining administrators.
<code>ContactProvider</code>	The string containing the specific <code>ContactProvider</code> implementation name which provides user profile to the forum, including <code>org.exoplatform.ks.common.user.BusinessProfileContactProvider</code> , <code>org.exoplatform.ks.common.user.DefaultContactProvider</code> , and <code>org.exoplatform.ks.common.user.PersonalProfileContactProvider</code> .
<code>MailServiceConfig</code>	The string containing the configuration of the Mail service used for the notifications in eXo Knowledge.
<code>OnlineUsers</code>	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.

4.2.6.2. 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: <code>inactiveDays</code> and <code>forumName</code> 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.

Job	Description
SendMailJob	Send email notifications in Forum.
UpdateDataJob	Update Forum statistics, such as posts or topics of users.

4.2.6.3. Plugin

4.2.6.3.1. 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.
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.

4.2.6.3.2. 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.

4.2.6.3.3. 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* .

4.2.6.3.4. 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.

4.2.6.4. 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.

4.2.7. eXo Collaboration management view

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

Attribute	Description
DataMap	The map containing the state information for 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.

Chapter 5. Security

5.1.1. 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 will be used by all eXo apps and even propagated to the JCR for [Access Control](#)

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

Since the security configuration is highly dependent of the app server we'll cover each application sever separately.

5.1.1.1. Tomcat

In the Tomcat bundle, the jaas configuration is controled by this `$TOMCATHOME/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.

Tip

Learn more about the syntax in the [JAAS tutorial](#). Read more about realms in tomcat in the [Tomcat Realm Howto](#)

5.1.1.2. JBoss

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

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

Additionnaly, you need to edit the application-policy to match the security-domain in `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>  
    </authentication>  
  </application-policy>  
</deployment>
```

```
<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>
```

Tip

Read More about JBoss security configuration in [JBoss Web Docs](#)

5.1.1.3. Common Changes

Finally, there are some common changes to do on both app servers.

5.1.1.3.1. configuration.properties

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

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

Note

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

5.1.1.3.2. portal.war

Inside `portal.war`, you should declare in the `web.xml` file the realm name:

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

5.1.1.3.3. rest.war

`rest.war` also needs to be modified to provide properly secured REST services.

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

Note

This change is very important as it allows to secure the download of files via WEBDAV.

5.1.2. Update the password encryption key of the RememberMe token

In eXo Platform, the password encryption key of the RememberMe token is always a default key defined in the *codekey.txt* file and this key is generated at the first bootstrap.

File location

File	Tomcat	Jboss
<i>codekey.txt</i>	<i>TOMCAT-HOME/gatein/conf/codec</i>	<i>JBOSS-HOME/server/<PROFILE>/conf/gatein</i>

Update the key

The administrators can simply update the key without doing any configuration as follows:

1. Remove the *codekey.txt* file.
2. Restart the server.

Chapter 6. Backup

6.1. How to back up eXo Platform

eXo Platform instance backup involves backing up the databases and the filesystems for JCR index and value storage. JCR backup should be done off-line. Allmost all data under `gatein.data.dir` should be backedup (find details below in "Plan backup" paragraph).

Note

Variable `gatein.data.dir` defined in eXo configuration *configuration.properties* and by default points to `$APPSEVERHOME/gatein/data` folder.

You can use this *tar* command for file backup from your application server home dir:

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

Note

SQL database backup should use backup tools your RDMBS provides. Example: [mysqldump](#) on MySQL

6.1.1. Plan backup

Start you backup strategy with a concept of a data repository. The backup data needs to be stored somehow and probably should be organized to a degree. eXo Platform instance backup will produces set of files. What can be located on various storage medias (hard disk, tape, optical or solid storages, or even a special remote backup services can be used).

Organization of the files in catalogs (folders) or use of different media is up to a concrete Plaform implementation.

But it is highly recommended to apply a [Backup rotation scheme](#) to make the backup implementation effective and reliable. Also always use your Operating System and Database software available backup solutions. It will simplify backup organization and help to avoid mistakes and data lost.

Plan Platform backup operations taking in account full stop of the Platform server(s).

Warning

In case of Platform cluster, every node should be stopped before a backup will be performed.

Platform backup consists of following parts:

- JCR data backup
 - JCR index files, pointed by configuration property `gatein.jcr.index.data.dir`

- JCR value storage files, pointed by configuration property `gatein.jcr.storage.data.dir`
- JCR database backup, database specified in JDNI configuration of Application server with "exo-jcrportal" name
- Organization service database backup, database specified in JDNI configuration of Application server with "exo-idmportal" name
- Transaction service files backup, pointed by configuration property `com.arjuna.ats.arjuna.objectstore.objectStoreDir`

Be ready for Restore: It's recommended to prepare tools (scripts etc) for restore at the backup planning stage. It will make particular restore operation quick and safe.

Note

Platform means one Portal application in this context , what is by default. But if your Platform instance runs several portals, each portal has own JCR, Organization and Transaction services. So, each portal should be backed up separately. In this document a single-portal Platform backup described. It just can be repeated for each portal of your system.

Note

Notes about JCR: Only two kind of JCR files are important in backup sense: index and value storages. The `gatein.jcr.data.dir` folder (by default it's `$gatein.data.dir/jcr` also contain *swap* sub-folder. The *swap* folder used for temporary files in case when BLOBs stored in database (see JCR configuration guide) and has no meaning for backup.

Below given an example of Platform backup process organization. This example introduces basic principles and will help to create your backup implementation.

6.1.1.1. Example backup planning

Environment

- Platform server runs on RedHat Linux server
- We have remote database server MySQL 5.1
 - JCR database - *jcrdb*
 - Organization service database - *idmdb*
- Platform files are on 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 store in `/mnt/netfs/platform/jta`
- Backup storage located on dedicated network mounted storage `/mnt/backups/myplfbackup`

Naming and Rotation

It's a general case when backup organized in two cycles rotation: everyday backup files stored for a last week days, older data stored on weekly basis and we'll plan to keep three years history at all.

To implement this approach we'll run daily backups (at night time when our site isn't in use) and will store result files (database and JCR files) on network storage in following structure:

- /my_plf_backup/2010/... - previous year archive
- /my_plf_backup/current/ - present year folder
- /my_plf_backup/current/weeks - weeks archive for current year folder
- /my_plf_backup/current/weeks/01 - first week in current year folder
- /my_plf_backup/current/weeks/02 - second week in current year folder
- /my_plf_backup/current/weeks/N - N week in current year folder

Files will have following format (using [ISO 8601](#) date format):

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

For files backup we have a shell script running on the Platform server. This script does next steps:

- Stops the Platform server (ensure full stop by log sniffing)
- Runs database backup tool against jcrdb and store result file in archive /mnt/backups/my_plf_backup/current/yyyy-MM-dd_mysql_jcrdb.tar.gz
- Runs database backup tool against idmdb and store result file in archive /mnt/backups/my_plf_backup/current/yyyy-MM-dd_mysql_idmdb.tar.gz
- Copies JCR value files to archive /mnt/backups/my_plf_backup/current/yyyy-MM-dd_jcr_values.tar.gz
- Copies JCR index files to archive /mnt/backups/my_plf_backup/current/yyyy-MM-dd_jcr_index.tar.gz
- Copies Transaction service files to archive /mnt/backups/my_plf_backup/current/yyyy-MM-dd_jta.tar.gz
- On each Sunday copies all 7 days old archive files to a week folder, e.g. /my_plf_backup/current/weeks/02 for backup at January 9, 2011.
- Otherwise deletes files older of 7 days from /my_plf_backup/current/
- If it's [first week](#) of a new year, the script creates a previous year folder in /my_plf_backup/, e.g. /my_plf_backup/2010, and moves content of /my_plf_backup/current/weeks there.

- Starts the Platform server
- Sends mail to Admin in case of error on any step.

Restore procedure planned as a manual operation.

Note

Example script implementation is outside of the scope of this guide.

6.1.2. Perform backup

Main steps should be performed to backup eXo Platform:

- Stop Platform instance and ensure it is stopped fully, use shell command `stop_eXo` (*find details in Installation chapter*)
- Run backup procedure that performs backup of:
 - JCR database
 - Organization service database
 - JCR value storage files
 - JCR index files
 - Transaction service files
- Archive backup files to your backup store
- Start Platform

Note

In case of Platform cluster, start it in an order according the Clustering documentation.

6.1.3. Perform restore

Restore from backup can be used in several cases: failure, duplicating a site (e.g. for testing). Same as for backup it's important to stop the platform fully before the restore.

Main steps should be performed to restore eXo Platform:

- Unarchive backup files from backup store to temporary location
- Stop Platform instance and ensure it is stopped fully, use shell command `stop_eXo` (*find details in Installation chapter*)
- Run restore procedure that performs restore from backup local files for:
 - JCR database

- Organization service database
- JCR value storage files
- JCR index files
- Transaction service files
- Start Platform

Note

In case of Platform cluster, start it in order according to the Clustering chapter steps.

6.1.4. Third party tools

Steps described above based on full backup of data. But there is an [incremental backup approach](#): an incremental backup preserves data by creating multiple copies that are based on the differences in those data, a successive copy of the data contains only that portion which has changed since the preceding copy has been created.

Having in account steps described above it's also possible to implement an incremental backup against Platform data.

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

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

In case of the example it's possible to organize full backup weekly (every Sunday) and incrementals each day of a week. Incremental backup will be faster, it will decrease time of your site everyday maintenance.

Note

It's also possible to use ready solutions as backula.org. Follow a product documentation for the implementation.

Chapter 7. Clustering

7.1. About Platform clustering

7.1.1. When should you consider clustering?

Installing eXo platform in cluster mode should be considered in the following cases:

- Load Balancing: when a single server node is not enough to handle the load
- High Availability: when you want to avoid a single point of failure by having redundant nodes

These characteristics should be handled by the overall architecture of your system. Load Balancing is typically achieved by a front server or device that distributes the request to the cluster nodes. Also, high availability on the data layer can be typically achieved using the native replication implemented by RDBMS.

In this chapter, we will cover only the changes needed by eXo to work in a cluster.

7.1.2. Shared file system

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

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

- the values storage
- the index

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

Note

We strongly advise the use of a mount point on a SAN.

7.2. Cluster setup

The switch to a cluster configuration is done in `configuration.properties`. This `configuration.properties` file must be set in the same way on all the cluster nodes.

First, switch the JCR to cluster mode.

```
gatein.jcr.config.type=cluster
gatein.jcr.index.changefilterclass=org.exoplatform.services.jcr.impl.core.query.jboss-cache.JBossCacheIndexChange
```

This will tell the JCR to enable automatic network replication and discovery between other cluster nodes.

Next, configure the path for the shared filesystem :

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

The path is shared, so all nodes will need read/write access to this path.

7.2.1. Advanced configuration

The cluster mode is preconfigured to work out of the box. It relies on the JBoss Cache configuration.

```
# JCR cache configuration
gatein.jcr.cache.config=classpath:/conf/jcr/jbossccache/${gatein.jcr.config.type}/config.xml

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

# JCR Index configuration
gatein.jcr.index.cache.config=classpath:/conf/jcr/jbossccache/cluster/indexer-config.xml
gatein.jcr.jgroups.config=classpath:/conf/jcr/jbossccache/cluster/udp-mux.xml
```

7.3. Cluster management

7.3.1. Cluster profile

You need to indicate the *cluster* kernel profile to eXo Platform. This can be done by editing `gatein.sh` like this:

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

or using *starteXo script* :

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

7.3.2. Initial Startup

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

Note

This constraint is only for the very first start. Once the initialization has been done, you can start nodes in any order

7.3.3. Startup and shutdown

Nodes of the cluster will automatically try to join others at startup. Once they discover each other, they will synchronize their state. During the synchronization the node is not ready to serve requests.

7.4. Clustering FAQ

7.4.1. How to migrate from local to cluster mode?

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

- Update the configuration to cluster mode as explained above on your main server
- Use the same configuration on other cluster nodes
- Move the index and value storage to the shared file system
- Start the cluster

7.4.2. Why is startup failed with a *Port value out of range* error?

On Linux Platforms, if you encounter an error at startup like this:

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

This problem arise under specific circumstances when JGroups, the networking library behind the clustering, attempts to detect the IP to use for communication with other nodes. Verify that:

- the hostname is a valid IP address, served by one of the network device of your machine (ie: eth0, eth1...).
- the hostname is NOT defined as localhost or 127.0.0.1

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

When starting up in the cluster mode under Linux and you encounter the following error:

```
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)
```

You must remember that clustering on Linux only works with IPv4.

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

is mandatory for running in the cluster mode under Linux.

Chapter 8. Deployment

8.1. How to remove the sample applications

eXo Platform comes with two sample portals that showcase the capabilities of the product. Before deploying your system in production, you will want to remove these sample apps.

Caution

The instructions below assume that you are using the `hsqldb` embedded database configuration.

8.1.1. Remove Acme Website

The Acme site is a sample extension that demonstrates an example of intranet you could implement with eXo platform. Once you have implemented your own extension, you will not likely need the acme website anymore. To remove the Acme site, do as follows:

1. Stop the server `shutdown.sh`
2. Delete `acme-portal.war`
3. Delete `exo.ecms.ext.acme.config.jar`
4. Delete `gatein/data`
5. Restart

8.1.2. Remove Acme Social Intranet

The Acme Social Intranet is a sample extension that demonstrates an example of intranet you could implement with eXo platform. Once you have implemented your own extension, you will not likely need the Social Intranet anymore. To remove the Acme Social Intranet, do as follows:

1. Stop the server `shutdown.sh`
2. Delete **`office-portal.war`**
3. Delete **`exo.platform.office.config.jar`**
4. Delete `gatein/data`
5. Restart

8.1.3. Remove the docs webapp

The docs are bundled as a convenience to be easily browsable while your portal is started. For production deployments, you will likely not need them. The docs webapp consists in a file **`docs.war`** located in `tomcat/webapps` and in `jboss/server/default/deploy`. In jboss, it is actually a folder named **`docs.war`**.

To remove docs, webapp do as follows:

1. Stop the server `shutdown.sh`
2. Delete file of folder **docs.war**
3. Restart

8.1.4. Remove crash

Crash is a complementary tool for development and maintenance. As it opens telnet and ssh sockets, we strongly advise to remove crash for your production deployments. Crash consists in a file **crash.war** in `tomcat/webapps`.

To remove crash, do as follows:

1. Stop the server `shutdown.sh`
2. Delete file **crash.war**
3. Restart

8.2. How to deploy a custom extension

Extensions are packaged as java EE web applications and come packaged as normal `.war` files. Hence, to deploy a custom extension, you will likely do as for any other webapp. In Tomcat this ends up by copying the war archive to the `webapps` folder

However, note that 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, but you must ensure that for your custom applications. There are several ways to control the loading order of webapps in Tomcat. Please refer to [Tomcat's Deployer How To](#)

8.3. How to setup an Apache Frontend

It may be necessary to use an HTTP server as a frontend 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 separate DNS names, without having to add a port to the URL. There are two methods that allow you to "glue" Apache HTTP Daemon and tomcat application server:

- via HTTP protocol, using [proxy module](#)
- via [Apache JServ Protocol](#), using [tomcat connector](#) or [HTTPD AJP proxy module](#)

8.3.1. 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 about Apache HTTP daemon host [here](#)

8.3.2. Connection via HTTP protocol (Apache mod_proxy)

With the *glue* method, it is necessary to configure Apache HTTP daemon to work as **reverse** proxy, which will redirect the client's requests to the app server's HTTP connector. For this type of connection, you will need to include the **mod_proxy** module in the HTTP demon configuratinon file. This can be found in the **httpd.conf** file, which is usually located here: **/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/
```

Note

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

In this example, HTTP daemon will work in **reverse proxy** mode (ProxyRequests Off) and will redirect all requests to tcp port 8080 on localhost. So, the configuration of a virtual host will look 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 about **mod_proxy**, review this [documentation](#)

8.3.3. Connection via AJP protocol

As described above, the 'glue' method can be implemented in two ways:

- using the native Apache HTTP demon's [AJP proxy module](#)
- using the native Apache Tomcat's [AJP conector](#)

With the first method, you only need the HTTP demon and application server, but settings are limited. With the second method, you can obtain much richer settings, but you will need to download and install additional modules for HTTP Daemon that are not included in the default package.

8.3.3.1. 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 port 8009 (the default setting for tomcat application server). \ A full list of virtual host configurations can be found here:

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

8.3.3.2. Apache Tomcat's AJP connector

1. Download AJP connector module from [here](#)
2. Move the downloaded **mod_jk.so** file into HTTPD's module directory. For example: **/etc/httpd/modules** (this may be different, depending on the OS)
3. Create the configuration file for module **mod_jk.conf**

```
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>
```

You can find more details in the [Tomcat docs](#)

1. Place the **mod_jk.conf** file into the directory where other configuration files for Apache HTTP Demon are located. For example, **/etc/httpd/conf.d/**
2. Create a **workers.properties** file, which defines [AJP workers](#) for HTTP demon.

```
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 example above you can change *WORKERNAME* to any value.

1. Place this file in the same directory as the mod_jk.conf file.
2. Update the virtual host configuration:

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

8.4. How to configure the session-timeout for the web server

The session-timeout is a concept that the web server uses to define a time period in which a session is valid/accessible. In a 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 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>
```

8.4.1. Server Tomcat

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 shutdown.sh
2. Open the file web.xml
3. Change the value of the session-timeout
4. Save

5. Restart (gatein.sh)

8.4.2. Sever Jboss

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

1. Stop the server shutdown.sh
2. Open the file web.xml
3. Change the value of the session-timeout
4. Save
5. Restart (run.sh)

Chapter 9. How to upgrade Platform

9.1. Purpose

Updating your Platform server is fairly easy but remains critical. This document present the detailed steps to upgrade Platform server.

9.2. Modifications details between versions

9.2.1. Modifications between Platform v.3.0.1 and 3.0.2

[ECMS Application templates](#)

It is about some Groovy template files (.gtmpl) stored in the JCR. These templates are used by the **Content List Portlet** and **Category Navigation Portlet** to display content lists in a pagination mode. There are four template types available, including:

- **Content List** templates.
- **Parameterized Content List** templates.
- **Category Navigation** templates.
- **Paginator** templates.

In Platform 3.0.1, those templates are stored in the same location in the JCR. But in Platform 3.0.3, their locations are separated to be clearer for the end user.

See the [Templates Structure Upgrade Notice](#) in the document located in *DeliveryFolder/upgrade-notes/PLF302CLVTemplatesStructureUpgradeNotice.pdf*.

9.2.2. Modifications between Platform v.3.0.2 to 3.0.3

Gadgets

Gadgets predefined in Platform are stored in the JCR, so if you have already used Platform in the production mode, delete those gadgets to get the new ones defined in the new Platform package. However, in case of the development mode, meaning that JCR database could be removed, those gadgets are auto-refreshed when upgrading server.

1. Modified gadgets:

- **Upcoming Events**.
- **To-do** is renamed to **Todo**.
- **IDE**.
- **Poll**.

- **Status Update** is renamed to **Activity Stream**.
- **Viewer Friends Gadgets** is renamed to **My Connections**.
- **Upcoming tasks** is renamed to **My Tasks**.
- **RSSFetch** is renamed to **SocialRssReader**.

2. Deleted gadgets:

- **Published Documents**.
- **Last Edited Documents**.
- **My documents**.
- **ApplicationRegistryGadget**.
- **Social Hello World**.
- **Code Sharing**.
- **RssFetcherGadget**.
- **Clock**.
- **CodeRunner - OpenSocial Deve....**
- **Space List**.

3. Added gadgets:

- None.

See the instructions below to refresh those gadgets in production mode.

9.2.3. Modifications between Platform v.3.0.3 and 3.0.4

- It is not required to make any modifications.

9.3. Steps to upgrade

9.3.1. Development mode

In this mode, you are supposed to use Platform with a clean database, which mean you can delete the database tables without impacting your development. Also, try following steps in the section **Production mode** below.

1. Stop the Platform server.
2. Clean up your JCR database tables (Drop JCR Datasource tables).
3. Backup your files before proceeding to the upgrade.

4. Replace the old eXo Platform binaries with the new ones:
 - a. Tomcat: make sure that the new Tomcat content is used.
 - b. JBoss: replace the old EAR package with the new ones, including the **gatein.ear** folder.
5. Apply *modifications* defined in *DeliveryFolder/upgrade-notes/PLF302CLVTemplatesStructureUpgradeNotice.pdf* in your custom configuration files.
6. Plug your custom configuration files and your custom datasources configuration into the new server.
7. Start the Platform Server to complete the upgrade.

9.3.2. Production mode

In this mode, you are supposed to keep your existing JCR data. Follow these steps:

1. Stop the Platform server.
2. Backup your Database and files before proceeding to the upgrade.
3. Replace the old eXo Platform binaries with the new ones.
4. Plug your custom configuration files and your custom datasources configuration into the new server.
5. Deploy crash web application is not deployed.
6. Start the new Platform Server.
7. Connect to Platform [CRaSH](#) console by using the Telnet or SSH console to update gadgets predefined in Platform:

```
telnet localhost 5000
connect -c portal -u root -p gtn portal-system
rm /production/app:gadgets/app:eventslist
rm /production/mop:workspace/mop:customizations/mop:Todo
rm /production/app:gadgets/app:Todo
select * from mop:workspaceclone where mop:contentid='Todo' | set mop:contentid "To-do"
rm /production/mop:workspace/mop:customizations/mop:RSSFetch
rm /production/app:gadgets/app:RSSFetch
select * from mop:workspaceclone where mop:contentid='RSSFetch' | set mop:contentid "SocialRssReader"
rm /production/app:gadgets/app:IDE
rm /production/app:gadgets/app:pollslist
rm /production/app:gadgets/app:Activities
rm /production/app:gadgets/app:ViewerFriends
rm /production/app:gadgets/app:taskslist
commit
```

8. Restart the Platform Server to complete the upgrade.
9. The *modifications* defined in *DeliveryFolder/upgrade-notes/PLF302CLVTemplatesStructureUpgradeNotice.pdf* are automatically applied (See the section *Can you safely upgrade?*):

Assuming you use a Platform 3.0.1 version, you can safely upgrade with no impact on your existing website. All impacted data will be migrated automatically at startup.