

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. Installing the Tomcat bundle	2
2.1.1. Startup	2
2.1.2. Shutdown	2
2.1.3. Startup scripts	2
2.2. Installing the JBoss EARS	4
2.2.1. Copy these files in jboss-root/server/default/deploy	4
2.2.2. Create a folder jboss-root/server/default/conf/gatein	4
2.2.3. Copy these files in jboss-root/bin	4
2.2.4. Configure JVM parameters	4
2.2.5. Add eXo logging categories in jboss-root/server/default/conf/jboss-log4j.xml	5
2.2.6. Startup	5
2.2.7. Shutdown	5
2.3. eXo Profiles	6
3. Configuration	7
3.1. eXo 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. Connecting to a production database	9
3.2.2. Changing the datasources names	12
3.3. FileSystem pathes	13
3.4. Mail Server	14
3.5. WebDAV Cache Control	14
3.6. Chat Server	15
3.6.1. XMPPMessenger	15
3.6.2. Chat server configuration	15
3.7. Logging	17
3.8. JCR	17
3.9. Users configurations	18
3.9.1. Super User configuration	18
3.9.2. eXo Platform default users list definition	18
4. Management	19
4.1. Management of eXo Platform	19
4.1.1. JMX interface	19
4.1.2. REST interface	19
4.2. Management view	19
4.2.1. Kernel management view	19
4.2.2. Cache management view	20
4.2.3. Content management view	21
4.2.4. Portlet Container management view	21
4.2.5. Portal management view	21
5. Security	26
5.1.	26
5.1.1. Changing the JAAS realm	26
6. Clustering	29

6.1. About Platform clustering	29
6.1.1. When should you consider clustering ?	29
6.1.2. Shared file system	29
6.2. Cluster setup	29
6.2.1. advanced configuration	30
6.3. Cluster management	30
6.3.1. Cluster profile	30
6.3.2. Initial Startup	30
6.3.3. Startup and shutdown	30
6.4. Clustering FAQ	31
6.4.1. How do I migrate from local to cluster mode ?	31
6.4.2. Why is startup failing with a <i>Port value out of range</i> error ?	31
7. Deployment	32
7.1. How to remove the sample applications	32
7.1.1. Removing Acme Website	32
7.1.2. Removing Acme Social Intranet	32
7.2. How to deploy a custom extension	32
7.3. How to setup an Apache Frontend	33
7.3.1. Base configuration for apache	33
7.3.2. Connection via HTTP protocol (Apache mod_proxy)	33
7.3.3. Connection via AJP protocol	34

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, per the Java EE specification, and as a configuration directory.

2.1. Installing 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
```

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 configuration.properties and configuration.xml
<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>	desactivates javascript and css merging for easier debugging

2.2. Installing 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:

2.2.1. Copy these 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`

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

Copy these files:

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

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

- `exokey.pem`
- `oauthkey.pem`

2.2.4. 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.jmxrem
EXO_XML="-Djavax.xml.stream.XMLOutputFactory=com.sun.xml.stream.ZephyrWriterFactory -Djavax.xml.stream.XMLInputF
JAVA_OPTS="$JAVA_OPTS $EXO_OPTS $EXO_PROFILES $EXO_XML"
```

On **Windows**, at 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.jmxrem
set "EXO_XML=-Djavax.xml.stream.XMLOutputFactory=com.sun.xml.stream.ZephyrWriterFactory -Djavax.xml.stream.XMLInputF
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

2.2.5. 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>
```

2.2.6. Startup

- 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
```

2.2.7. Shutdown

- 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 Configuration

In eXo platform, the configuration lives 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's 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

Let's stop and explain some parts of the eXo internals in order to 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 his name.

The default portal container in eXo Platform is called simply "portal". This is 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 done mostly in configuration.properties. In most cases, this should be the only file 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 builtin configuration for the "portal" portal container.

In most cases, you should not need to 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 book 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. Connecting 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. Preparing 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 to do it automatically on 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. Configuring 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 server.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 should now look like :

```
<!-- eXo JCR Datasource for portal -->
<Resource auth="Container" driverClassName="com.mysql.jdbc.Driver" maxActive="20" maxIdle="10" maxWait="10000" r
```

```
<!-- eXo IDM Datasource for portal -->
<Resource auth="Container" driverClassName="com.mysql.jdbc.Driver" maxActive="20" maxIdle="10" maxWait="10000" r
```

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 setearier

3. Change the url to access your DataBase :

```
<connection-url>jdbc:hsqldb:${jboss.server.data.dir}${/}exo${/}hypersonic${/}exo-xxx_portal-localDB</connection-
```

to

```
<connection-url>jdbc:mysql://_${host}_:3306/_$dbname_</connection-url>
```

The configuration should now look like :

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

```

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

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

```

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. Changing 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. For this, two steps are needed :

- 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" maxActive="20" maxIdle="10" maxWait="10000" r
<!-- eXo IDM Datasource for portal -->
<Resource auth="Container" driverClassName="org.hsqldb.jdbcDriver" maxActive="20" maxIdle="10" maxWait="10000" r
```

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"/>
```

3.3. FileSystem pathes

eXo needs read/write access to several pathes 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
    /hsql
  /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 notificaitons 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](#).

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 comma-separated list of key=pair values, 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

Property	Description	Default value
env		
eXoAuthProvider		
authenticationURL	URL to authenticate users	/organization/authenticate/
authenticationMethod	HTTP method used to pass parameters	POST
eXoUserProvider		
findUsersURL	URL to find all users	/organization/xml/user/find-all/
findUsersMethod	HTTP method for user/find-all	GET
getUsersURL	URL to retrieve a range of users	/organization/xml/user/view-range/
getUsersMethod	HTTP method for user/view-range	GET
usersCountURL	URL to count users	/organization/xml/user/count/
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

Port	Type	Description
		this port with <code>exo.chat.port</code> property.
9090 && 9091	Admin Console (http)	The port used for respectively the unsecured and secured Openfire Admin Console access.
3478 & 3479	STUN service	The port used for the service that ensures connectivity between entities when behind a NAT.

3.7. Logging

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.

3.8. 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.jbosscache.JBossCacheIndexChangesFilter
gatein.jcr.index.changefilterclass=org.exoplatform.services.jcr.impl.core.query.DefaultChangesFilter

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

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

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

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

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

3.9. Users configurations

3.9.1. 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`.

3.9.2. 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.

Chapter 4. Management

4.1. Management of eXo Platform

Management of the eXo Platform resources is critical for production usage.

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 the various resources provided by the eXo Platform server, and management actions which can be performed, and how to obtain relevant metrics.

4.1.1. JMX interface

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

At runtime, the MBeans are registered by the eXo Kernel in the MBeanServer, which the application server creates, and the MBeans are directly viewable in the JMX console. However, we do advocate using a better JMX client such as the JVisualVM, available since Java 6.

In order 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 GateIn 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 login/password of the user's 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 view

4.2.1. Kernel management view

4.2.1.1. PortalContainer

PortalContainer manages all objects and configurations for a given portal. The JMX name is `exo:container=portal,name="portal"`.

- `configurationXML`: provides the effective runtime configuration calculated by the loading mechanism. URLs used to load parts are indicated to help.
- `registeredComponentNames`: provides the list of all components registered.

4.2.2. Cache management view

Cache is essential for improving the performance of a server infrastructure.

4.2.2.1. Cache management and statistics

Each cache is exposed and provides statistics and management operations. The JMX name of each cache MBean uses the template `exo:service=cache,name=CacheName` where `CacheName` is the name of the cache.

Attribute	Description
Capacity	the maximum capacity of the cache
HitCount	the number of times the cache was successfully queried
MissCount	the number of times the cache was queried without success
Size	gives the number of entries in the cache
TimeToLive	attribute gives the maximum lifetime that an entry is considered valid. A value of <code>-1</code> means that the entries will never become stale

Operation	Description
<code>clearCache()</code>	evicts all the entries from the cache. It can be used to force a programmatic flush of the cache

4.2.2.2. Cache service

The cache service manages the different caches. The JMX name is `exo:service=cachemanager`.

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

4.2.2.3. PicketLink Cache

PicketLink is the default implementation for the organization model. It uses its own cache JMX management interface under the name `exo:portal="portal",service=PicketLinkIDMCacheService,name=plidmcache`.

`invalidateAll()`: invalidate all caches `invalidate(namespace)`: invalidates a specific cache namespace

4.2.3. Content management view

WCM provides a management view for:

4.2.3.1. WCM Composer

The WCM Composer is responsible for assembling the pages, and is key for serving pages efficiently. The JMX name is `exo:portal="portal",service=composer,view=portal,type=content`

- `Cached`: whether the cache is used or not.
- `CachedEntries`: how many nodes are in the cache.
- `cleanTemplates()`: cleans all templates in the cache.
- `setCached(cached)`: enables/disables caching in the composer.

4.2.4. Portlet Container management view

Portlet containers provide a management view that mainly consists of statistic exposure. The JMX name of a portlet container follows this template: `exo:application="Application",portlet="Portlet"` where `Application` is the name of web application and `Portlet` is the portlet name in the web application.

4.2.4.1. Portlet Container statistics

Each portlet container has a corresponding managed resource that exposes statistics as attributes. The Portlet lifecycle has several phases (named action, event, render and resource). These are particular operations that a portlet can execute, and each is monitored.

Attribute	Description
<code>ActionCount</code> , <code>EventCount</code> , <code>RenderCount</code> and <code>ResourceCount</code>	the number of executions of a particular portlet phase
<code>MeanActionTime</code> , <code>MeanEventTime</code> , <code>MeanRenderCount</code> and <code>MeanResourceCount</code>	the average time spent during the execution of a particular portlet phase
<code>MaxActionTime</code> , <code>MaxventTime</code> , <code>MaxRenderCount</code> and <code>MaxResourceCount</code>	the maximum time spent during the execution of a particular portlet phase
<code>MinActionTime</code> , <code>MinEventTime</code> , <code>MinRenderCount</code> and <code>MinResourceCount</code>	the minimum time spent during the execution of a particular portlet phase

4.2.5. Portal management view

The portal management exposes the portal resources either for management or for obtaining statistics.

4.2.5.1. Template statistics

The template management exposes the various templates used by the portal and its components to render markup. Various statistics are available for individual templates, and aggregated statistics, such as the list of the slowest templates. Most of the management operations are performed on a single template; those operations take the template identifier as an argument. The JMX name of the template statistics MBean follows the template: `exo:view=portal,service=statistic,type=template`.

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

Operation	Description
<code>getAverageTime(templateId)</code>	returns the average time spent in the specified template
<code>getExecutionCount(templateId)</code>	returns the number of times the specified template has been executed
<code>getMinTime(templateId)</code>	returns the minimum time spent in the specified template
<code>getMaxTime(templateId)</code>	returns the maximum time spent in the specified template

4.2.5.2. Template management

Template management provides the capability to force the reload of a specified template. The JMX name of the template management MBean follows the template `exo:view=portal,service=management,type=template`.

Operation	Description
<code>reload(templateId)</code>	reloads a specified template. The <code>reload()</code> operation reloads all the templates

4.2.5.3. Skin management

Attribute	Description
SkinList	the list of the loaded skins by the skin service

Operation	Description
<code>reloadAll(skinId)</code>	forces a reload of the specified skin and the operation <code>reloadSkins()</code> forces a reload of the loaded skins

4.2.5.4. Token Store

4.2.5.4.1. Definition of Token store

Token (sometimes called the security token or authentication token) is an object that caches identity or profiles to authorize users. It will be used in authentication to increase the information security. You, as an administrator, can create, delete, retrieve or clean tokens. For example, instead of signing in with your username and password when accessing a site, you only need to login for the first time with the token. After the client's token has been sent, the authentication will be processed from the server.

eXo Platform provides several token store mechanisms via implementing the **TokenStore** interface. The **TokenStore** interface has TWO methods, including **createToken** and **validateToken**.

The **createToken** method is to create and return a token which will be retained until it is removed or its validity period is expired.

The **validateToken** method is to validate the created token. If the token is valid, this method will return the attached credentials. The token will be removed if the removal argument is set to "true" or if the token is no longer valid.

4.2.5.4.2. Configure the token service

All token services used in eXo Platform must be implemented by sub-classing **AbstractTokenService**.

The token service will initialize the validity property by retrieving init-params named as "service.configuration."

```
<component>
  <key>org.exoplatform.web.security.security.CookieTokenService</key>
  <type>org.exoplatform.web.security.security.CookieTokenService</type>
  <init-params>
    <values-param>
      <name>service.configuration</name>
      <value>jcr-token</value>
      <value>7</value>
      <value>DAY</value>
      <value>autologin</value>
    </values-param>
  </init-params>
</component>
```

The token will be deleted when it is invalid or when any **removal** is set programmatically.

Currently, eXo Platform only provides TWO types of token store services, including **CookieTokenService** and **TransientTokenService**.

- **CookieTokenService** provides a mechanism of token store to JCR via the [Chromatic](#) project. This service stores a token into the defined node in JCR. Thus, you must put the Chromatic lifecycle name when initializing this service. The token is only removed if the database, where the token is stored, is cleaned.
- **TransientTokenService** puts a token into the temporary memory. The token will be removed once the server has been stopped.

4.2.5.4.3. Attributes and operations

All services extending **AbstractTokenService** can become **TokenStore** services if you define them as

components. For example:

```
<component>
  <key>org.exoplatform.web.security.security.TransientTokenService</key>
  <type>org.exoplatform.web.security.security.TransientTokenService</type>
  <init-params>
    <values-param>
      <name>service.configuration</name>
      <value>memory-token</value> [1]
      <value>1</value> [2]
      <value>MINUTE</value> [3]
    </values-param>
  </init-params>
</component>
```

In which, the token service will initialize by retrieving init-params named as "service.configuration" and the validity property must include THREE values:

- [1](#): Service name.
- [2](#): Amount of time.
- [3](#): Timing unit.

In the above code, the service name is "memory-token" and the token will be expired after ONE minute.

eXo Platform supports FOUR timing units, including SECOND, MINUTE, HOUR, DAY.

The **AbstractTokenService** has some methods, such as:

- "getToken(K id) : T" returns the token with the token id.
- "deleteToken(K id) : T" deletes the token with the id and returns the deleted token value.
- "getAllTokens() : K[]" returns all tokens stored in data.
- "createToken(Credentials credentials) : void" creates a token with the defined credential which is an immutable object containing username and password.
- "validateToken(String tokenKey, boolean remove) : Credentials" validates a token. If the token is valid, it returns the attached credential.

4.2.5.5. Portal statistics

The JMX name of the portal statistics MBean follows the template `exo:view=portal,service=statistic,type=portal`.

Attribute	Description
PortalList	returns the list of the loaded portals

Operation	Description
<code>getThroughput(portalId)</code>	returns the number of request per second for a specified portal
<code>getAverageTime(portalId)</code>	returns the average time spent in a specified portal
<code>getExecutionCount(portalId)</code>	returns the number of times a specified portal has been executed
<code>getMinTime(portalId)</code>	returns the minimum time spent in the specified portal
<code>getMaxTime(portalId)</code>	returns the maximum time spent in the specified portal

4.2.5.6. Application statistics

Various applications are exposed to provide relevant statistics. The JMX name of the application statistics MBean follows the template: `exo:view=portal,service=statistics,type=application`.

Attribute	Description
<code>ApplicationList</code>	returns the list of the loaded applications
<code>SlowestApplications</code>	returns the list of the 10 slowest applications
<code>MostExecutedApplications</code>	returns the list of the 10 most executed applications
<code>FastestApplications</code>	returns the list of the 10 fastest applications

Operation	Description
<code>getAverageTime(applicationId)</code>	returns the average time spent in the specified application
<code>getExecutionCount(applicationId)</code>	returns the number of times the specified application has been executed
<code>getMinTime(applicationId)</code>	returns the minimum time spent in the specified application
<code>getMaxTime(applicationId)</code>	returns the maximum time spent in the specified application

Chapter 5. Security

5.1.1. Changing 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, edit the change it need to change the JAAS realm to fit your own security constraints, LInside configuration.properties, identifythe 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.

Chapter 6. Clustering

6.1. About Platform clustering

6.1.1. When should you consider clustering ?

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

- Load Balancing : when a single 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.

6.1.2. 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.

6.2. Cluster setup

The switch to a cluster configuration is done in `configuration.properties`

First, switch the JCR to cluster mode:

```
gatein.jcr.config.type=cluster
gatein.jcr.index.changefilterclass=org.exoplatform.services.jcr.impl.core.query.jbosscache.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.

6.2.1. 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
```

6.3. Cluster management

6.3.1. 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
```

6.3.2. 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

6.3.3. 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.

6.4. Clustering FAQ

6.4.1. How do I 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

6.4.2. Why is startup failing 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

Chapter 7. Deployment

7.1. 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.

7.1.1. Removing Acme Website

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

7.1.2. Removing Acme Social Intranet

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

7.2. 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](#)

7.3. 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](#)

7.3.1. 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](#)

7.3.2. 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](#)

7.3.3. Connection via AJP protocol

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

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

With the first method, you only need the HTTP daemon 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.

7.3.3.1. 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>
```

7.3.3.2. 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
    JkLogStampsFormat "[%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>
```