Jahia’s next-generation, open source CMS stems from a widely acknowledged vision of enterprise application convergence – web, document, search, social and portal – unified by the simplicity of web content management.
1 Overview..................................................................................................................................................... 4
  1.1 Introduction.......................................................................................................................................... 4
  1.2 What's in this documentation?............................................................................................................ 4
2 Prerequisites............................................................................................................................................... 5
  2.1 Minimal system requirements ............................................................................................................. 5
  2.2 Java Virtual Machine (JVM) .................................................................................................................. 5
  2.3 Database ............................................................................................................................................... 7
  2.4 Other preparations and checks .......................................................................................................... 10
3 Installation................................................................................................................................................ 11
  3.1 Main installation steps: ...................................................................................................................... 11
  3.2 Settings during installation ................................................................................................................. 11
  3.3 Folder structure after installation with bundled Tomcat ................................................................... 18
  3.4 Discovering Jahia - first usage ............................................................................................................ 21
  3.5 Installing a production server – additional steps ............................................................................. 21
  3.6 Different types of environment.......................................................................................................... 22
  3.7 Application server specific installations ............................................................................................. 23
  3.8 Modules .............................................................................................................................................. 35
4 Configuring some Jahia features .............................................................................................................. 37
  4.1 Personalizing URLs .............................................................................................................................. 37
  4.2 Caching ............................................................................................................................................... 39
  4.3 Clustering ............................................................................................................................................ 42
  4.4 LDAP .................................................................................................................................................... 46
  4.5 Authentication .................................................................................................................................... 46
  4.6 Document converter .......................................................................................................................... 54
  4.7 Document viewer ............................................................................................................................... 56
  4.8 Document thumbnails ........................................................................................................................ 57
  4.9 Video thumbnails ................................................................................................................................ 58
  4.10 Image service ................................................................................................................................... 58
  4.11 Configuration files externalization .................................................................................................. 59
  4.12 Error and thread dump directories ................................................................................................. 61
5 Fine Tuning ............................................................................................................................................... 63
5.1 Tomcat ................................................................................................................................................ 63
5.2 Database ............................................................................................................................................. 64
5.3 Module generation queue .................................................................................................................. 65
5.4 Operating mode .................................................................................................................................. 66
5.5 JCR DataStore garbage collector ........................................................................................................ 66
5.6 Storing binary files .............................................................................................................................. 67
5.7 Increasing bundleCacheSize ............................................................................................................... 68
6 Monitoring ................................................................................................................................................ 70
   6.1 Stack trace dumps .............................................................................................................................. 70
   6.2 Memory dumps .................................................................................................................................. 71
   6.3 Java profilers ....................................................................................................................................... 72
   6.4 Tools .................................................................................................................................................... 72
   6.5 Other Issues ........................................................................................................................................ 75
7 Frequently asked questions (FAQ) ........................................................................................................... 76
   7.1 How to backup Jahia? ......................................................................................................................... 76
   7.2 How to restore an environment from a backup? ............................................................................... 78
   7.3 Modifying the Logging Level ........................................................................................................... 80
   7.4 How to handle module generation timeouts? ................................................................................... 81
   7.5 How to upgrade Tomcat for an existing Jahia installation? ............................................................... 83
   7.6 How to clean referencesKeeper nodes? ............................................................................................ 84
1 Overview

1.1 Introduction

Jahia’s next-generation, open source CMS stems from a widely acknowledged vision of enterprise application convergence – web, document, search, social and portal – unified by the simplicity of web content management.

By leveraging state of the art Open Source frameworks and libraries, Jahia offers a complete solution for developing, integrating, delivering, and managing content across intranets, extranets, and the internet with a much lower total cost of ownership than proprietary systems.

1.2 What’s in this documentation?

This document is intended to give an overview of the various aspects of advanced installation, configuration and the fine-tuning of Enterprise Jahia v6.6.

It is intended for system administrators and advanced users.

This guide is structured in the following way:

Chapter 2: Prerequisites and system requirements

Chapter 3: Installation of Jahia on various platforms

Chapter 4: Application server specific installations

Chapter 5: Configuring main Jahia features

Chapter 6: Fine tuning your Jahia server

Chapter 7: Monitoring your server for performance

Chapter 8: FAQ
Should you have questions, please do not hesitate to contact us as mentioned on our website (http://www.jahia.com).

2 Prerequisites

2.1 Minimal system requirements

Please find below the minimum system requirements in order to properly run Jahia Enterprise Edition v6.6.

OS:

- Windows
- Linux
- Solaris
- Mac OSX

Suggested Min. Development Configuration:

- Dual Core 2GHz or +
- 2 GB RAM
- 5 GB HD

Suggested Min. Production Environments:

- Quad Core (64 bit CPU and OS)
- 4 GB RAM
- 100 GB HD

Warning: 32 bit JVM are limited in max memory (1.5 GB under Windows - 2 or 3 GB under Linux/Solaris). Jahia v6.6 tries to cache a maximum of data in order to boost performance. So we highly recommend 64 bit environments with enough memory available at least for all production environments.

2.2 Java Virtual Machine (JVM)

In order to run Jahia, you first need to install a Java SE (Java Platform, Standard Edition) 5 as minimum on your system (Java SE 6 is highly recommended). As Jahia needs to compile some JSP files, the Java Runtime Environment (JRE) only won’t be sufficient.
To check if Java is already installed on your system, type the following command line at the prompt of your system:

```bash
java -version
```

You should get a message indicating which Java version is installed on your system. Please note that the same message will be displayed if you only have a JRE installed. If an error is returned, you probably don’t have a Java Platform installed.

If you have installed other versions of the Java Platform, Java Runtime Environment or other Java servers on your system, we recommend that you run a few checks before starting the installation in order to be sure that Jahia will run without problems.

If you need to obtain and install a new Java SE, you can find both Linux and Windows versions on the Oracle Web site:  [http://www.oracle.com/technetwork/java/javase/downloads](http://www.oracle.com/technetwork/java/javase/downloads)

To install a Java Virtual Machine on a Windows system, you need to have administrator rights on your computer. Please contact your system administrator if you don’t have sufficient permissions.

It is recommended that the installation path of the Java Platform does not contain any spaces (not like in the default `C:\Program Files\Java\jdk1.6.0_xx`, where “`xx`” is the release number – so please change it to a path without spaces, like `C:\Java\jdk1.6.0_xx`).

After the installation, you have to set the `JAVA_HOME` environment variable to the directory where you have installed the Java SE. Note that at run time Jahia will check that this variable is correctly set, and will stop if it is not the case.

To setup this variable, follow these steps:

### 2.2.1 Under Windows

i) Open the Control Panel, and the System option. In Windows 7 and Vista it is: Control Panel → System and Security → System → Advanced System Settings. Then, depending on your system:

- Select the Advanced tab and click on the Environment Variables button (Windows 7/Vista/XP/2000)
- Select the Properties tab and click on the Environment button (Windows NT)
ii) Click on New in the "System variables" section to add a new environment variable. Enter the following information:

- Variable name: JAVA_HOME
- Variable value: c:\Java\jdk1.6.0_xx (replace this value with the correct path)

Click on OK to validate your entry. The Java Virtual Machine should now be correctly set-up. Please note that on Windows NT you will need to restart your computer to apply the changes.

2.2.2 Under Linux

Set the JAVA_HOME variable to the root directory of your JDK installation. Both examples below suppose you have installed the JDK version 1.6 in your /usr/java directory. The classpath is usually set by typing:

export JAVA_HOME=usr/java/jdk1.6.0_xx (in bash or ksh)

export JAVA_HOME=usr/java/jdk1.6.0_xx (in csh or tcsh)

2.2.3 Under Solaris

Set the JAVA_HOME variable to the root directory of your JDK installation. The examples below suppose you have installed the JDK version 1.6 in your /usr/java directory. The classpath is usually set by typing:

export JAVA_HOME=/usr/java (in ksh)

JAVA_HOME=/usr/java;export (in sh)

setenv JAVA_HOME /usr/java (in csh or tcsh)

2.3 Database

Enterprise Jahia v6.6 is by default distributed with the Sun Java DB / Apache Derby database engine. If you wish to get started rapidly or for rapid prototyping purposes, you can use the provided database as is.

But in production, and also for developing a serious project, you should use a standalone database instead. This section addresses only the mandatory configurations. Please refer to the “Fine tuning” section, before going live.
Your database should be UTF-8 compliant! Have this in mind when creating a new database for Jahia

Default settings are currently already predefined to allow Jahia to run on Sun Java DB / Apache Derby, PostgreSQL, MySQL and the Enterprise Jahia v6.6 also supports Microsoft SQL Server and Oracle. During the Jahia installation you will have to provide the URL to the database you have created for Jahia. These connection strings are different for each database.

Jahia may have also detected bugs in certain DB versions, which would cause errors in Jahia, so we integrated validations during installation, which will not allow installing Jahia with these database versions.

### 2.3.1 MySQL

The default database URL (the connection string) for MySQL is:

```
jdbc:mysql://localhost/jahia?useUnicode=true&characterEncoding=UTF-8&useServerPrepStmts=false
```

where `localhost` should be replaced by the fully qualified domain name (mysql.mydomain.com) or IP address of the MySQL server if it is not located on the same machine as the Jahia server, and `jahia` is just the default name of the database where Jahia tables will be created.

If your MySQL server is not running on the standard port (3306), you should add `:port` after the domain name where `port` is the port number.

Jahia is using InnoDB engine for its database engine on MySQL, so be sure that you have configured your MySQL for InnoDB. Here are some default configuration options for your database to be put in your my.cnf or my.ini file.

```ini
# * InnoDB
#
# InnoDB is enabled by default with a 10MB datafile in /var/lib/mysql/.  
# Read the manual for more InnoDB related options. There are many!  
#
# You can write your other MySQL server options here  
# ...
# Data files must be able to hold your data and indexes.  
# Make sure that you have enough free disk space.  
innodb_data_file_path = ibdata1:100M:autoextend  
# Set buffer pool size to 50-80% of your computer's memory  
innodb_buffer_pool_size=1024M  
innodb_additional_mem_pool_size=256M
```
Configuration and fine tuning Guide Enterprise Jahia v6.6.2

# Set the log file size to about 25% of the buffer pool size
innodb_log_file_size=256M
innodb_log_buffer_size=64M
#
innodb_flush_log_at_trx_commit=1

max_allowed_packet has to be at least set to 100M, otherwise Jahia will prohibit installation. In case you have chosen to store the files in the database, it should be at least the same size as the biggest file that will be uploaded on your server. Said differently, your users won’t be able to upload any file bigger than the size you specify here. You should also configure jahiaFileUploadMaxSize in WEB-INF/etc/config/jahia.properties accordingly. The Jahia limitation should not be bigger than the database limitation, otherwise the Jahia UIs will allow files to be uploaded that the database will not be able to store.

max_allowed_packet = 1024M

2.3.1 MySQL on Mac OS X

Please note that for MySQL versions from 5.5.9 to 5.5.12 on MacOSX, you must set the value of lower_case_table_names to 1 (http://bugs.mysql.com/bug.php?id=60309).

2.3.2 PostgreSQL

The default database URL (the connection string) for PostgreSQL 8.x/9.x is:

jdbc:postgresql://localhost:5432/jahia

where jahia is the default name of the database where Jahia tables will be created. If your PostgreSQL server is located on a distant computer and/or on a non-default port (5432), please, adjust the connection URL accordingly.

Make sure your PostgreSQL server is accepting TCP connections. Please refer to your database documentation for detailed instructions on how to configure PostgreSQL to accept TCP connections.
2.3.3 Oracle

Enterprise Jahia v6.6 also comes with JDBC drivers for Oracle. These drivers work with Oracle 10g, 11g.

The default database URL (the connection string) for Oracle is:

jdbc:oracle:thin:@localhost:1521:jahia

where localhost should be replaced by the fully qualified domain name (oracle.mydomain.com) or the IP address of the Oracle Server if it is not located on the same machine as the Jahia server, and jahia is the default name of the database where Jahia tables will be created.

1521 is the standard port for Oracle. If you Oracle server is running on a different port, please change it here.

2.3.4 Microsoft SQL Server

Enterprise Jahia v6.6 is provided with JDBC drivers for MS SQL 2008.

The default database URL (the connection string) for Microsoft SQL Server is:

jdbc:sqlserver://localhost;databaseName=jahia

where localhost should be replaced by the fully qualified domain name (sqlserver.mydomain.com) or the IP address of the MS SQL Server if it is not located on the same machine as the Jahia server, and jahia is the default name of the database where Jahia tables will be created.

If your MS SQL Server is not running on the standard port (1433), you should add :port after the domain name, where port is the port number.

jdbc:sqlserver://localhost:port;databaseName=jahia

2.4 Other preparations and checks

Check that you have no TOMCAT_HOME and no CATALINA_HOME environment variable set.
3 Installation

Jahia’s official and nightly builds are distributed as installation packages, which contain the entire software suite (Jahia Composite Content Platform, Jahia xCM, Jahia Studio) as well as the Jahia ACME Demo and ACME-Space demo, several template sets and hundreds of composite modules.

3.1 Main installation steps:

- Download the latest stable Jahia 6.6 build from [http://www.jahia.com](http://www.jahia.com) by choosing the right downloadable package for your operating system.
- Double-click on the downloaded installation package, which will start the installation wizard.
  - On Unix servers with graphical environment, you can start the installation wizard running `java -jar <your-downloaded-jahia-jar>`.
  - On Unix servers where you have no graphical environment, you can start the installation also in the Console Mode: `java -jar <your-downloaded-jahia-jar> -console`.
  - In case you would require running the wizard in Console Mode on Windows, you will need to open your command prompt with administrator privileges.
- Follow the installation wizard. See next sections for a detailed description of the settings.
- At the end, you can let the wizard launch Jahia (if the bundled Apache Tomcat server was selected as an option). Otherwise, you can launch Jahia using the generated shortcut or within the created installation folder using a console window launch the command `./startJahia.sh` (on Linux/MacOSX) or “startJahia.bat” (on Windows).
- **Important**: the first start of your Jahia may take up to 3 minutes, depending on your hardware (initial templates publication and modules deployment). The next starts will be much faster.

3.2 Settings during installation

3.2.1 Installation path

There shouldn’t be any spaces in your folder naming. For example `C:\jahia66\` is OK while `C:\jahia 66\` won’t work.
3.2.2 **Installation type – Express install**

Jahia will be installed with a default application server and database with pre-defined configuration settings:

3.2.2.1 **Default application server**

The default Enterprise Jahia v6.6 is distributed with an Apache Tomcat 6.0.35 application server.

No manual configuration of the server is required, as it will be directly setup during the Jahia installation. By default Tomcat will use standard ports (8080, 8009 and 8005). Please ensure that you do not have any other servers/services running and using those ports. Optionally, you can change Tomcat ports during the “Custom install” installation type (see “3.2.3 Installation type – Custom install”).

3.2.2.2 **Default database**

Enterprise Jahia v6.6 is by default distributed with the embedded Sun Java DB / Apache Derby database engine. If you wish to get started rapidly, you can use the provided database as is. With the “Custom Install” option you can choose to install Jahia to another more robust standalone database during the configuration wizard of Jahia.

Please note that you cannot simply switch the database at a later stage on the same installation. You will have to export the content and import it into a new Jahia installation configured with the different database.

3.2.3 **Installation type – Custom install**

If you want to install Jahia on a custom environment (application server, database, mail server configuration, different port numbers), you need to choose the “Custom Install” option.

3.2.3.1 **Application server**

Enterprise Jahia v6.6 can be installed with an Apache Tomcat 6.0.35 application server. If you want to install into your own server, you need to deselect the “Apache Tomcat” checkbox on Step 5 of the
installation wizard and click Next. On the next page you will be able to choose whether the installation is targeted into one of these application servers:

- Apache Tomcat (in case you want to deploy Jahia yourself, into an existing Tomcat server other than the one bundled with Jahia)
- Red Hat JBoss

The installed Jahia will then include or exclude some specific configurations and libraries, which are needed to make Jahia run smoothly in the targeted application server. See the next chapter “3.7 Application server specific installations” for further information.

### 3.2.3.2 Database

The embedded Sun Java DB / Apache Derby database engine is only recommended for rapid testing or prototyping. For production we recommend a standalone database, and during installation you can choose between:

- Microsoft SQL Server
- MySQL 5.x
- Oracle 10g / 11g
- PostgreSQL 8.x/9.x
- Sun Java DB / Apache Derby (standalone)

Please note, that you cannot simply switch the database at a later stage on the same installation. You will have to export the content and import it into a new Jahia installation configured with the different database.

During installation you will be asked to provide the connection URL (see chapter “2.3 Database” for details) and the user/password for accessing the database.

Furthermore you also will be able to set whether binary data should be stored in the database or directly on a file system (for clustered Jahia setup the file system need to be shared by all cluster nodes) and if the
Jahia DB structure (tables, indexes, sequences) needs to be created first (this option needs to be unchecked e.g. when running the installation wizard for installing second, third, etc. cluster nodes).

### 3.2.3.3 Application and server settings

**Apache Tomcat configuration:**

*This section is available only if you have chosen to use the bundled Tomcat application server.*

Here you have the possibility to configure the different ports used by Tomcat.

**Web application settings:**

Here you have the possibility to specify the context path for Jahia. If you want to deploy it in the root context (“/”), just leave the field blank.

You need also to specify a login and password that will be required to access the administration and debugging tools embedded in Jahia.

### 3.2.3.4 Jahia operating mode

Here you have to choose in which mode you want to install Jahia.

- **Development Mode:** includes the set of all Jahia optional modules, template sets and pre-packaged sites. Enables the Development Mode for template deployment

- **Production:** includes the "core" set of Jahia modules. Disables the Development Mode for template deployment. Studio Mode access is also disabled

- **Remote server:** same as "Production". Additionally, content editing activities are limited to live content only.

Just take care that even if you can switch later between the modes (you can reconfigure it in `jahia.properties`), some modules will be packaged only when you perform the installation in Development Mode. Installing in Production Mode, and then switching to Development Mode will activate the development dedicated features (like the Studio), but will not deploy the additional modules. You will
have to deploy them manually using either the `deployModule.sh` or `deployModule.bat` module deployment script. Please refer to the “3.8 Module” section for more information.

**Differences between Development Mode and Production Mode**

<table>
<thead>
<tr>
<th></th>
<th>Development Mode</th>
<th>Production Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Studio</strong></td>
<td>Yes</td>
<td>Not installed</td>
</tr>
<tr>
<td><strong>Modules</strong></td>
<td>All modules are installed, including all default template-sets</td>
<td>No module is installed, they must be deployed according to your specific needs</td>
</tr>
<tr>
<td><strong>Cache</strong></td>
<td>Display extra information directly into the rendered page by passing <code>parametrized</code> in the URL</td>
<td>No</td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td>Display extra information directly into the rendered page using <code>moduleinfo=true</code> as a param in the URL</td>
<td>No</td>
</tr>
<tr>
<td><strong>Exceptions</strong></td>
<td>Exception trace is rendered in HTML</td>
<td>No</td>
</tr>
<tr>
<td><strong>Rules</strong></td>
<td>Auto reload of changed rules in WEB-INF/etc/</td>
<td></td>
</tr>
<tr>
<td><strong>Jobs</strong></td>
<td>All jobs are treated as new and will override any existing job</td>
<td>Jobs are rescheduled if already existing</td>
</tr>
<tr>
<td><strong>Url rewriting service</strong></td>
<td>Scanned every 5 seconds, reloaded if changes</td>
<td>No scan</td>
</tr>
<tr>
<td><strong>New modules in WEB-INF/shared_modules</strong></td>
<td>Scanned every 5 seconds, new module is loaded if found</td>
<td>Scanned every 5 seconds. Scanning interval can be defined or deactivated in jahia.properties</td>
</tr>
<tr>
<td><strong>Module changes</strong></td>
<td>Scanned every 5 seconds for changes in first level files and all files under META-INF and WEB-INF of each module</td>
<td>No scan</td>
</tr>
<tr>
<td><strong>Groovy patcher</strong></td>
<td>Scan modules every 5 seconds, apply patches if changes are found</td>
<td>Scanned every 5 seconds. Scanning interval can be defined or deactivated</td>
</tr>
<tr>
<td><strong>Log4J</strong></td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
**LDAP configuration**

In case you will use LDAP directory as a provider for application users or/and groups, you can choose to configure LDAP provider settings during installation. If you check this option, you will then access an additional screen, where you can setup your configuration for user and group providers.

If you do not configure them during the installation process, you will still be able to do it later from the configuration files. Please refer to the “5.4 LDAP” section for more information.

**Cluster configuration**

You can also configure Jahia to be run in cluster mode. If you check this option, you will then access an additional screen where you can setup your cluster configuration.

Here, you will have to specify if the node you are installing is the processing server. Remember that only one node of this type is allowed in the same cluster. Please refer to the “4.3 Clustering” section for more information.

You will also have to specify a unique server identifier and declare the IP and listening ports of each node from your cluster (including the current one).

**3.2.3.5 System administrator settings**

You need to at least provide the password for the root user, who, like a super-user, always has all of the privileges in Jahia. So you should choose a strong password and keep it secret.

**3.2.3.6 Mail server**

**Mail server**: this field contains the SMTP host name, with advanced options.

Jahia now uses the Apache Camel framework for messaging, and the format of the mail endpoint should conform to the one, required by the Camel Mail Component ([http://camel.apache.org/mail.html](http://camel.apache.org/mail.html)), i.e.:

```
[smtp|smtps]://[username@]host[:port][?options]
```

All parts except the **host** are optional. See use cases below.
Mail administrator: the field contains a single e-mail address or multiple addresses (separated by a comma) of users who will receive system-level notifications (e.g. about errors, if this option is enabled).

Mail from: the default sender e-mail address for an e-mail message.

Here are several use cases for "Mail server" field values:

1. SMTP server does not require authentication and uses the standard port 25:

smtp.acme.com

2. SMTP server requires authentication and uses non-standard port 11019:

smtp.acme.com:11019?username=myuser&password=secretpassword

3. GMail example: SMTP server requires authentication and SSL enabled (or TLS):

smtps://smtp.gmail.com?username=acme@gmail.com&password=mypassword

or

smtp.gmail.com:587?username=acme@gmail.com&password=mypassword&mail.smtp.starttls.enable=true

4. Enable the mail server debugging option to see the details of e-mail server communication:

smtp.acme.com?mail.debug=true

3.2.3.7 Configuration externalization

The last step in the Jahia Installer provides an optional possibility to externalize the configuration. What this means is that it is possible to avoid storing Jahia’s properties and custom settings inside the deployed web application. This may be useful for many reasons, such as proper separation of the configuration and software, JEE non-exploded deployment, ease cluster deployments, etc.

By default the configuration externalization is not activated, which means that Jahia will store the configuration in the WEB-INF/etc/config directory, in the jahia.properties file.
### 3.3 Folder structure after installation with bundled Tomcat

Here is a brief overview of the folders structure in Jahia along with the important files that will be used throughout this guide. The files and folders in the Web application (here under `webapps/ROOT`) should be the same as what is on the other application servers.

```
tomcat
|-- bin
  |-- catalina.bat
  |-- catalina.sh
  |-- shutdown.bat
  |-- shutdown.sh
  |-- startup.bat
  `-- startup.sh
|-- conf
  |-- server.xml
  `-- web.xml
|-- lib
|-- webapps
  |-- ROOT
    |-- admin
    |-- css
    |-- engines
    |-- errors
    |-- gwt
    |-- html
    |-- icons
    |-- iphone
    |-- javascript
    |-- META-INF
      `-- context.xml
    |-- modules
    |-- tools
    |-- WEB-INF
      |-- classes
      `-- etc
        `-- config
          |-- jahia.properties
          |-- jahia.advanced.properties
          |-- log4j.xml
          `-- urlrewrite.xml
        `-- repository
          |-- export
          `-- jackrabbit
            `-- repository.xml
          |-- nodetypes
          |-- rules
          |-- jahiapp.xml
          |-- profileModule.xml
          |-- root.xml
          |-- site.xml
          `-- templatesSet.xml
```
Here is a brief overview of the important folders:

**webapps/ROOT/engines**: This directory contains all the JSP, images and JavaScript files of Jahia engines (Content Manager, Content Picker, Live Content Picker etc.).

**webapps/ROOT/modules**: These are template-sets, JahiApps and component modules, deployed on your server (shared among all virtual sites).

**webapps/ROOT/META-INF/context.xml**: Database connection information. This configuration is applicable only for Apache Tomcat server and only for the first server start. After the first start, Tomcat copies this file to `<tomcat>/conf/Catalina/localhost/ROOT.xml` (file name depends on the chosen context name). So any modifications to the DB datasource after the first Tomcat start should be done on the file in `<tomcat>/conf/Catalina/localhost/` and in the context.xml (for consistency).

**webapps/ROOT/WEB-INF/classes**: Besides some configuration files, here you will find mainly resource bundle files used to translate the Jahia interface in other languages. There are normally at least 2 files for each language: `JahiaInternalResources.properties` and `JahiaTypesResources.properties`.

**webapps/ROOT/WEB-INF/etc**: The etc directory contains most of the configuration files of Jahia. The config sub-directory contains the main configuration file(s), `jahia.properties`, in EE there is also the `jahia.advanced.properties` and the error logging `log4j.xml`. These are the main configuration files you
will need to modify in order to adapt Jahia to your environment. Note that if you have used the Configuration Externalization option in the Installer, the configuration files will be located inside a jahia-config.jar at the location you have specified (by default in tomcat/lib).

The repository directory contains the configuration files for Jackrabbit repository.

The spring directory may contain multiple Jahia service configurations, but is empty by default. The files have been moved inside Jahia’s JARs as normally they shouldn’t be modified. It is still possible to override them by placing a copy of a file in this location.

**webapps/ROOT/WEB-INF/var/db**: The database scripts to create the DB schema of Jahia and to connect to the corresponding database can be found here.

**webapps/ROOT/WEB-INF/var/repository**: The Jackrabbit repository home, where the workspace configuration, and version storage is located as well as search indexes.

The search indexes will be stored in these directories.

**webapps/ROOT/WEB-INF/var/repository/index**

and **webapps/ROOT/WEB-INF/var/repository/workspaces/*/index**: The search indexes will be stored in these directories.

**webapps/ROOT/WEB-INF/var/shared_modules**: Modules and template-sets located in that directory will be deployed to the server on startup or whenever a file changes during runtime. Template-sets will be available in the drop-down list when you create a new virtual site, and modules will be seen in the left panel of the Jahia Studio or in the Jahia Edit mode.

**work**: This directory will contain both a compiled (for instance *simple_jsp.class*) and a readable version (for instance *simple_jsp.java*) of your modules, or the templates of Jahia engines if you don’t use the precompiled engines files. This can prove helpful in case you have an error in a template showing in the Tomcat logs, for instance: sitemap_jsp.java:984: illegal start of expression. If you want to make sure that all JSP files of the templates are recompiled after a change, you may want to delete the Standalone directory in Work. Next time you access a page, Tomcat will recompile all JSP files used by the page.
3.4 Discovering Jahia - first usage

This applies only if you want to discover Jahia 6.6, using the prepackaged demonstration site. It assumes that you have installed Jahia in Development Mode, so that the example templates and the modules they require have been automatically deployed.

- Open a browser and go to [http://localhost:8080/start](http://localhost:8080/start). Use the root user credentials set up during the installation process. You will discover the new Jahia landing page. Click on the Server administration link then "Web project management" and you're ready to create your first site.
- Import the new ACME Website Demo package. After successful import, click on the “Go to Edit Mode” tab to see the Edit Mode for this ACME website. Click on the "Publication" menu on the top toolbar and select "Publish site". Then select "Bypass workflow" when prompted. Switch to the other available language and publish the entire site again for this language.
- Switch to the Live or Contribute Mode and enjoy!

3.5 Installing a production server – additional steps

This applies when you install your production server, and assumes that you have installed Jahia in Production Mode.

Before being able to create your first website, you will have to deploy your custom set of templates and modules. But during the development process, you may have used some Jahia standard modules, automatically available on your installation. Notice that most of those modules were available because you installed your development server using the development mode. As your production server uses the production mode, only the core modules will be available. So, you also need to deploy yourself the standard modules you want to use.

- Prepare all the war files for your custom templates and modules, and the one for each standard module you want to use. For the standard modules, you can either download them from the Jahia Forge, or retrieve them from your development server (they are available in WEB-INF/var/shared_modules/). In case you download the modules from the Jahia Forge, take
care to download the same version of the module as the one you have tested during your validation process.

- Install all the war files of your modules using the deployModules script provided by Jahia on your production server, while the server is shut down.

At the root level of your Jahia installation folder: deployModule.(sh|bat)

<path_to_war_file_to_deploy><path_to_jahia_web_app_dir>, e.g.:

```
deployModule.bat article-1.0.war c:\Jahia-6.6\tomcat\webapps\ROOT
```

- Start Jahia. The modules will be automatically deployed during the startup process. As your server is configured in Production Mode, the modules will be deployed only during the startup of the application; whereas when Development Mode is configured, they will be automatically detected and deployed at any time while Jahia is running.

- Now you can either import your site data from an export of your integration/development platform, or create a new empty site.

- Now let your users enter content on their site.

### 3.6 Different types of environment

During the life-cycle of a project you will need different types of environments:

- Development environment - each of your developers will have their own environment. Those developer environments are normally much lighter than the one needed for production. Your developers can either use the integrated DBMS (Apache DERBY) or use another DBMS (MySQL, MS SQL Server, PostgreSQL, Oracle).

- Integration environment - this environment will help you integrate the work of all your developers on the same platform and prepare the site(s) you are going to deploy in production.

- Production environment - this one is the last step in the development life-cycle of your project.
3.7 Application server specific installations

3.7.1 Apache Tomcat 6.0.x / 7.0.x

3.7.1.1 Installation

The installation procedure for an existing Apache Tomcat 6 or 7 is as follows:

- Launch the Installer.
- Choose the Custom Install (advanced) installation type.
- Select only Jahia xCM + Jahia Composite Content Platform pack.
- Choose Apache Tomcat 6.0.x as the target application server type.
- Follow the next steps of the Installer.

Once the Installer is finished in your install directory you should find a file named `ROOT.war`, a file named `shared-libraries.zip` and, if you chose to externalize the configuration, you will also have a `jahia-config.jar` file.

3.7.1.2 Deployment

Further, it is assumed that the Apache Tomcat server is installed in the `<tomcat>` folder.

1. Unzip the content of the `shared-libraries.zip` file into your `<tomcat>/lib` directory.
2. In case ROOT was configured as the Web application context name, please, remove or rename the default Tomcat ROOT Web application at `<tomcat>/webapps/ROOT`, if it exists, to e.g. `tomcat`.
3. Unzip the contents of the `ROOT.war` file to the `<tomcat>/webapps/ROOT` directory.
4. Adjust the `serverHome` variable value in the `<tomcat>/webapps/ROOT/WEB-INF/etc/config/jahia.properties` file to point to your `<tomcat>` folder path.
5. If you chose to externalize the configuration into a JAR file, either copy the `jahia-config.jar` file into the `<tomcat>/lib` directory, or modify the `catalina.sh/bat` files to add it to the classpath.
6. Adjust the JVM, connector and servlet container options appropriately (see next sections).

---

1 The name of the WAR file corresponds to the configured Web application context name
2 The name of the WAR file corresponds to the configured Web application context name
3 The name of the directory corresponds to the configured Web application context name
JVM tuning options

The default JVM options in the Tomcat's startup script (<tomcat>/bin/catalina.bat or <tomcat>/bin/catalina.sh) should be adjusted to use server JVM ("-server" option), have at least 1 GB\(^4\) heap size (-Xms1024m -Xmx1024m) and at least 256 MB\(^5\) as a limit for the permanent generation heap size (-XX:MaxPermSize=256m), if applicable, also adding other tuning options.

This can be done by adding the following line to your <tomcat>/bin/catalina.bat (Windows OS):

```
set CATALINA_OPTS=%CATALINA_OPTS% -Dsun.io.useCanonCaches=false -Xms1024m -Xmx1024m -XX:MaxPermSize=256m -server -Dhibernate.jdbc.use_streams_for_binary=true -verbose:gc -XX:+HeapDumpOnOutOfMemoryError -Djava.net.preferIPv4Stack=true
```

or to the <tomcat>/bin/catalina.sh file (non Windows OS):

```
CATALINA_OPTS="$CATALINA_OPTS -Xms1024m -Xmx1024m -server -Dhibernate.jdbc.use_streams_for_binary=true -XX:MaxPermSize=256m -verbose:gc -XX:+HeapDumpOnOutOfMemoryError -Djava.net.preferIPv4Stack=true"
```

### 3.7.1.3 HTTP/AJP connector tuning options

The following configuration for the HTTP and AJP connectors (configured in the <tomcat>/conf/server.xml file) is recommended\(^6\), which includes UTF-8 URI encoding, compression of the response content etc.

For Tomcat 6.0.x:

```
<Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443"
    maxThreads="300" acceptCount="100"
    enableLookups="false"
    emptySessionPath="true"
    URIEncoding="UTF-8"
    compression="on"
    compressableMimeType="text/plain,text/html,text/xml,text/css,text/javascript,application/x-javascript,application/javascript" />

<Connector port="8009" protocol="AJP/1.3" redirectPort="8443"
    enableLookups="false" emptySessionPath="true"
```

---

\(^4\) For production systems the memory options should be adjusted accordingly to achieve high performance and scalability.

\(^5\) For production systems the memory options should be adjusted accordingly to achieve high performance and scalability.

\(^6\) Connector settings, especially maxThreads and acceptCount values, should be adjusted accordingly to achieve high performance and scalability in production run.
3.7.1.4  Session cookie path (Tomcat 7.0.x only)

The following attribute needs to be set on the `<Context/>` element in the `<tomcat>/conf/context.bat` file for Tomcat 7.0.x (in Tomcat 7 it replaces the `emptySessionPath="true"` attribute on the HTTP/AJP connectors):

```xml
<Context sessionCookiePath="/"/>
```

3.7.1.5  Module deployment listener

Jahia provides a Tomcat host lifecycle listener that is checking for new / updated modules and performs deployment of their classes and JARs before the Web application startup.

The following line should be added into the `<Host/>` element in the `<tomcat>/conf/server.xml` file:

```xml
<Listener className="org.jahia.configuration.server.tomcat.ModuleClassesHostLifecycleListener"/>
```

3.7.1.6  JSP Compiler/Servlet tuning options
JSP Servlet parameters (configured in the `<tomcat>/conf/web.xml` file) can be optimized in the following way:

```xml
<servlet>
  <servlet-name>jsp</servlet-name>
  <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
  <init-param>
    <param-name>fork</param-name>
    <param-value>true</param-value>
  </init-param>
  <init-param>
    <param-name>trimSpaces</param-name>
    <param-value>true</param-value>
  </init-param>
  <init-param>
    <param-name>genStrAsCharArray</param-name>
    <param-value>true</param-value>
  </init-param>
  <init-param>
    <param-name>development</param-name>
    <param-value>true</param-value>
  </init-param>
  <init-param>
    <param-name>checkInterval</param-name>
    <param-value>300</param-value>
  </init-param>
  <init-param>
    <param-name>xpoweredBy</param-name>
    <param-value>false</param-value>
  </init-param>
  <load-on-startup>3</load-on-startup>
</servlet>
```

In production, you can set the `development` parameter to “false” to prevent the compiler from checking for JSP modifications too often and enables background compilation with `checkInterval` seconds.

### 3.7.1.7 Jahia configuration externalization

This section focuses on how to manually build and deploy an externalized configuration in Tomcat, and how to reference it by the deployed Jahia application. For more information about the externalized configuration itself, please refer to the “4.11 Configuration files externalization” section. If you have used the Installer’s option to create an externalized configuration, you can also skip this section, although it might be interesting to read to see an alternative to the JAR configuration deployment.

1. Create a folder on your server’s filesystem for your configuration files (for example `/app/jahia/externalizedConf/`).
2. In this folder, create the following subfolder structure: `org/jahia/config/` (to have `/app/jahia/externalizedConf/org/jahia/config/`).

3. Choose one of the 2 following options:
   
a. Add the externalized folder to the Tomcat classpath: in the `<tomcat>/bin/setclasspath.sh` file, add the custom folder to the classpath variable: 
   
   ```
   CLASSPATH=$CLASSPATH:<My externalized folder> (CLASSPATH=$CLASSPATH:/app/jahia/externalizedConf for example).
   ```

   b. Add the externalized folder to the Tomcat common loader: in <tomcat>/conf/catalina.properties file, add your externalized configurations path to the common.loader property.

4. You can now put your externalized configurations inside the `config/` folder.

### 3.7.2 Red Hat JBoss 4.2.x/4.3.x

#### 3.7.2.1 Installation

The installation procedure for Red Hat JBoss is as follows:

1. Launch the Installer.
2. Choose Custom Install (Advanced) installation type.
3. Select only Jahia xCM + Jahia Composite Content Platform pack.
4. Choose Red Hat JBoss 4.2.x/4.3.x as the target application server type.
5. Follow the next steps of the Installer.

Once the Installer is finished with your install directory, you should find a file named `ROOT.war` and a file named `jahia-jboss-config.sar`.

#### 3.7.2.2 Deployment

Further, it is assumed that the Red Hat JBoss server is installed in the `<jboss>` folder and the default server configuration file set is used with the path `<jboss>/server/default` that will be referred to as `<server-home>`.

---

7 The name of the WAR file corresponds to the configured Web application context name.
Two files (WAR and SAR), generated by the Installer, need to be deployed into the `<server-home>/deploy` folder in an expanded form.

1. Create the folder `<server-home>/deploy/jahia-jboss-config.sar` and unzip the contents of the `jahia-jboss-config.sar` file into that folder.
2. Create the folder `<server-home>/deploy/ROOT.war`\(^8\) and unzip the contents of the `ROOT.war`\(^9\) file into it.
3. In case ROOT was configured as the Web application context name, please remove or rename the default JBoss ROOT Web application at `<server-home>/deploy/jboss-web.deployer/ROOT.war` to e.g. `jboss.war`.
4. Adjust the `serverHome` variable value in the `<server-home>/deploy/ROOT.war/WEB-INF/etc/config/jahia.properties` file to point to your `<server-home>` folder path.
5. Adjust the JVM, connector and servlet container options (including portlet TLD) appropriately (see next sections).

### 3.7.2.3 Jahia configuration externalization

This section focuses on how to deploy the externalized configuration in JBoss, and how to reference it by the deployed Jahia application. For more information about the externalized configuration itself, please refer to the “4.11 Configuration files externalization” section. If you used the Jahia Installer to generate an externalized configuration JAR, you can skip to step 3 to setup the CLASSPATH to include the `jahia-config.jar`.

1. Create a folder on your server filesystem for your configuration files (for example `/app/jahia/externalizedConf/`)
2. Create in this folder the following subfolder structure: `org/jahia/config/` (to have `/app/jahia/externalizedConf/org/jahia/config/`)
3. Adjust the `JBOSS_CLASSPATH` environment variable:
   - Windows OS: in the `<jboss-home>/bin/run.bat` file add the following lines:

---

\(^8\) The name of the folder corresponds to the configured Web application context name, with adding a `.war` postfix.

\(^9\) The name of the WAR file corresponds to the configured Web application context name.
if "%JBOSS_CLASSPATH%" == "" (
set JBOSS_CLASSPATH=/app/jahia/externalizedConf
) else (
set JBOSS_CLASSPATH=%JBOSS_CLASSPATH%;/app/jahia/externalizedConf
)

or if you are using a configuration externalization JAR generated using Jahia’s Installer:

if "%JBOSS_CLASSPATH%" == "" ( 
set JBOSS_CLASSPATH=/PATH_TO_JAR/jahia-config.jar
) else ( 
set JBOSS_CLASSPATH=%JBOSS_CLASSPATH%;/PATH_TO_JAR/jahia-config.jar
)

• Non Windows OS - in the <jboss-home>/bin/run.sh file add the following lines:

    if [ "x$JBOSS_CLASSPATH" = "x" ]; then
        JBOSS_CLASSPATH="/app/jahia/externalizedConf"
    else
        JBOSS_CLASSPATH="$JBOSS_CLASSPATH:/app/jahia/externalizedConf"
    fi

or if you are using a configuration externalization JAR generated using Jahia’s Installer:

    if [ "x$JBOSS_CLASSPATH" = "x" ]; then
        JBOSS_CLASSPATH="/PATH_TO_JAR/jahia-config.jar"
    else
        JBOSS_CLASSPATH="$JBOSS_CLASSPATH:/PATH_TO_JAR/jahia-config.jar"
    fi

4. You can now put your externalized configurations inside the config/ folder if you’re not using an externalization JAR.

3.7.2.4 JVM tuning options

The default JVM options in the JBoss' startup script (<jboss>/bin/run.bat or <jboss>/bin/run.conf) should be adjusted to use server JVM ("-server" option), have at least 1 GB\textsuperscript{10} heap size (-Xms1024m -Xmx1024m) and at least 256 MB\textsuperscript{11} as a limit for the permanent generation heap size (-XX:MaxPermSize=256m), if applicable, also adding another tuning options, if not already present.

\textsuperscript{10} For production systems, the memory options should be adjusted accordingly to achieve high performance and scalability.

\textsuperscript{11} For production systems, the memory options should be adjusted accordingly to achieve high performance and scalability.
Ensure the following options are present in the `JAVA_OPTS` variable:

- server
- Xms1024m
- Xmx1024m
- XX:MaxPermSize=256m
- Dhibernate.jdbc.use_streams_for_binary=true
- Djava.net.preferIPv4Stack=true
- verbose:gc
- XX:+HeapDumpOnOutOfMemoryError

### 3.7.2.5 HTTP/AJP connector tuning options

The following configuration for the HTTP and AJP connectors (configured in the `<server-home>/deploy/jboss-web.deployer/server.xml` file) is recommended\(^\text{12}\), which includes UTF-8 URI encoding, compression of the response content etc.:

```xml
<Connector port="8080" address="${jboss.bind.address}" protocol="HTTP/1.1" maxThreads="300" acceptCount="100"
emptySessionPath="true" maxHttpHeaderSize="8192"
enableLookups="false" redirectPort="8443"
connectionTimeout="20000" disableUploadTimeout="true"
URIEncoding="UTF-8"
compression="on"
compressableMimeType="text/plain,text/html,text/xml,text/css,text/javascript,application/x-javascript,application/javascript" />

<Connector port="8009" address="${jboss.bind.address}" protocol="AJP/1.3"
emptySessionPath="true" enableLookups="false" redirectPort="8443"
maxThreads="300" URIEncoding="UTF-8"/>
```

### 3.7.2.6 JSP Compiler/Servlet tuning options

JSP Servlet parameters (configured in the `<server-home>/deploy/jboss-web.deployer/conf/web.xml` file) can be optimized as follow. Please note, the registration of the portlet tag libraries using `tagLibJar2` parameter:

---

\(^{12}\) Connector settings, especially `maxThreads` and `acceptCount` values, should be adjusted accordingly to achieve high performance and scalability in production run.
<servlet>
<servlet-name>jsp</servlet-name>
<servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
<init-param>
@param-name>fork</param-name>
@param-value>true</param-value>
</init-param>
<init-param>
@param-name>trimSpaces</param-name>
@param-value>true</param-value>
</init-param>
<init-param>
@param-name>genStrAsCharArray</param-name>
@param-value>true</param-value>
</init-param>
<init-param>
@param-name>development</param-name>
@param-value>true</param-value>
</init-param>
<init-param>
@param-name>checkInterval</param-name>
@param-value>300</param-value>
</init-param>
<init-param>
@param-name>xpoweredBy</param-name>
@param-value>false</param-value>
</init-param>
<init-param>
@param-name>compilerSourceVM</param-name>
@param-value>1.5</param-value>
</init-param>
<!-- Use a custom options class to allow the shared tag lib descriptors to be loaded from jars in the tomcat sarconf/tlds directory. The standard options implementation can only find taglibs based on the class loader classpath. -->
<init-param>
@param-name>engineOptionsClass</param-name>
@param-value>org.jboss.web.tomcat.service.jasper.JspServletOptions</param-value>
</init-param>
<!-- Specify the jars relative to the jbossweb-tomcat6.sar that should be scanned for common tag lib descriptors to include in every war deployment. -->
<init-param>
<description>JSF standard tlds</description>
@param-name>tagLibJar0</param-name>
@param-value>jsf-libs/jsf-impl.jar</param-value>
</init-param>
<init-param>
<description>JSTL standard tlds</description>
@param-name>tagLibJar1</param-name>
@param-value>jstl.jar</param-value>
</init-param>
<init-param>
<description>Portlettaglibs</description>
@param-name>tagLibJar2</param-name>
@param-value>../jahia-jboss-config.sar/lib/pluto-taglib-2.0.2.jar</param-value>
</init-param>
In production you can set the development parameter to false to prevent the compiler from checking for JSP modifications too often and enable background compilation with checkInterval seconds.

3.7.3 Red Hat JBoss 5.1.x

3.7.3.1 Installation

The installation procedure for Red Hat JBoss 5.1.x is as follows:

1. Launch the Installer.
2. Choose Custom Install (Advanced) installation type.
3. Select only Jahia xCM + Jahia Composite Content Platform pack (uncheck the Apache Tomcat one).
4. Choose Red Hat JBoss 5.1.x as the target application server type.
5. Follow the next steps of the Installer.

Once the Installer is finished with your install directory, you should find a file named ROOT.war\(^{13}\) and a file named jahia-jboss-config.zip.

Further, it is assumed that the Red Hat JBoss 5.1.x server is installed in the <jboss> folder and the default server configuration file set is used with the path <jboss>/server/default that will be referred to as <server-home>.

3.7.3.2 JVM tuning options

Some JVM options have to be adjusted to run Jahia.

1. In order to have temporary data (Jahia caches, errors, thread dumps etc.) inside JBoss directory structure:
   - on Windows add the following line into <jboss>\bin\run.bat before the line “set DIRNAME=“:

\(^{13}\) The name of the WAR file corresponds to the configured Web application context name.
- Set "JAVA_OPTS=%JAVA_OPTS% -Djava.io.tmpdir=%JBOS_HOME%/server/default/tmp"

  - On non-Windows platform add the following line into `<jboss>/bin/run.sh` after the line "export JBOSS_HOME":

    `JAVA_OPTS="$JAVA_OPTS -Djava.io.tmpdir=$JBOSS_HOME/server/default/tmp"`

2. In case the embedded Apache Derby DBMS is used, a Derby home must be set:

   - On Windows add the following line into `<jboss>/bin/run.bat` before the line "set DIRNAME=":

     `set "JAVA_OPTS=%JAVA_OPTS% -Dderby.system.home=%JBOSS_HOME%/server/default/deploy/ROOT.war/WEB-INF/var/dbdata"

   - On non-Windows platform add the following line into `<jboss>/bin/run.sh` after the line "export JBOSS_HOME":

     `JAVA_OPTS="$JAVA_OPTS -Dderby.system.home=$JBOSS_HOME/server/default/deploy/ROOT.war/WEB-INF/var/dbdata"

3. Adjust JVM memory options to have at least 1 GB heap size (`-Xms1024m -Xmx1024m`) and at least 256 MB as a limit for the permanent generation heap size (`-XX:MaxPermSize=256m`):

   - On Windows change the corresponding values in the `<jboss>/bin/run.conf.bat` and on non-Windows platform in the `<jboss>/bin/run.conf` to have e.g.:

     `-Xms2048M -Xmx2048M -XX:MaxPermSize=384M`

4. Add further JVM options, required by Jahia:

   - On Windows add the line into the `<jboss>/bin/run.conf.bat`:


---

14 In case you've chosen non-root context for the Jahia WAR file, please replace ROOT.war in the path with the appropriate one.

15 For production systems, the memory options should be adjusted accordingly to achieve high performance and scalability.
on non-Windows platform add the following line into <jboss>/bin/run.conf:

```
```

### 3.7.3.3 JMX Console user

In order to be able to shut down JBoss properly the JMX Console user needs to be configured.

Uncomment the line in the file `<server-home>/conf/props/jmx-console-users.properties` and configure the password for admin user:

```
# A sample users.properties file for use with the UsersRolesLoginModule
admin=admin
```

### 3.7.3.4 Deployment

1. If you have chosen the ROOT context for Jahia Web application during installation wizard (default option), please rename the JBoss’ `<server-home>/deploy/ROOT.war` into e.g. `jboss.war`.

2. Unzip the content of the `jahia-jboss-config.zip` file, generated by the installer, into the `<server-home>/` folder forcing overwriting existing files (`server.xml` and `web.xml` will be overwritten with the Jahia-tailored ones).

3. Create the folder `<server-home>/deploy/ROOT.war` and unzip the contents of the `ROOT.war` file into it.

4. Adjust the `serverHome` variable value in the `<server-home>/deploy/ROOT.war/WEB-INF/etc/config/jahia.properties` file to point to your `<jboss>` folder path.

### 3.7.3.5 Jahia configuration externalization

---

16 The name of the folder corresponds to the configured Web application context name, with adding a `.war` postfix.

17 The name of the WAR file corresponds to the configured Web application context name.
3.8 Modules

3.8.1 Module deployment

Modules are extensions to Jahia, which are packaged in a war file and can be deployed on a server. A module can contain different kinds of resources: JSPs, definitions in CND files, CSS and images, resource bundles, XML or ZIP import files, rules definitions, libraries, spring files...

Modules are deployed by using either the `deployModule.sh` or `deployModule.bat` module deployment script. At the root level of your Jahia installation folder, type:

```
deployModule.(sh|bat) <path_to_war_file_to_deploy> ... <path_to_jahia_web_app_dir>
```

So for instance, to deploy the article-1.0.war module, For Windows OS:

```
deployModule.bat article-1.0.war c:\Jahia-6.6\tomcat\webapps\ROOT
```

Or, for non-Windows OS:

```
deployModule.sh article-1.0.war /opt/local/jahia-6.6/tomcat/webapps/ROOT
```

The file will be copied to the `WEB-INF/var/shared-modules` folder, and embedded libraries will be deployed to `WEB-INF/classes`.

If the module does not contain any java class or library, it can be deployed when the server is running. If it contains java classes, the server must be properly shut down before deployment.

If a previous version of the module was deployed, with a different version number, it has to be properly removed from `WEB-INF/var/shared-modules` (and `WEB-INF/lib`) before deploying the new version.

3.8.2 Cluster deployment

In cluster environments, we must differentiate between cold deployment and hot deployment of modules.

3.8.2.1 Cold deployment
Cold deployment means that modules are being deployed when the cluster is completely shut down. In this case, cold deployment should be done on all the nodes using the deployModules.sh script. **When starting up the cluster, it is critical that the processing server be started first**, and that its initialization is completed before starting the other nodes.

### 3.8.2.2 Hot deployment

For hot deployments of modules (when the cluster is up and running), installation on different cluster nodes should be done in sequence, but **the processing server must be the last one on which the module is deployed**. Modules will be available only when they have been deployed on the processing server.

If the module contains new classes or libraries, or classes or libraries updates, the server must be properly shut down before the deployment, as explained in Deploy modules with classes. The module can be deployed on every server, one after the other. Deploy on the first server, restart it, and then go to the second server, and so on. The processing server must be the last one on which the module is deployed. There should be no downtime of the whole cluster.

### 3.8.3 Deployment on sites

Once the war file has been deployed, modules become available. They can then be deployed to web projects with the studio by using the "deploy" menu, or with the "Templates and JahiApps management" panel in the Jahia Administration.

If a new version of the module is uploaded on the server, it will be automatically deployed on all sites that are currently using it. All updates will be immediately available in the site.
4 Configuring some Jahia features

4.1 Personalizing URLs

4.1.1 URL Rewriting

Please refer to the URL rewriting section in the JahiaPedia for more information.


4.1.2 Removing jsessionid from URLs

Jahia requires that the user’s session is correctly handled. Usually, applications use cookies to track the session. If cookies are not available on the client or the client connects for the first time, the application server adds a parameter in all links to handle session tracking. This parameter can create issues when indexing links by search engines. Jahia can automatically remove it from all links. This feature can be enabled in the jahia.properties file:

```properties
## Disable the jsessionid parameter added by application server to track session when no cookie is present.
disableJsessionIdParameter = true
jsessionIdParameterName = jsessionid
```

By default, the sessionid parameter removal is enabled, and won’t appear in links. If you need to support browsers which do not handle cookies, you can disable this feature. The jsessionIdParameterName should match the jsession parameter name used by your application.

4.1.3 Changing context and port number

During the installation

Changing the Jahia Web application context path (the default is ROOT) as well as default Apache Tomcat port numbers – in case Tomcat pack is selected – is possible during the Installer UI, by choosing and completing the “Custom installation” option at the beginning.

After the installation
Once you have installed Jahia, you will still be able to change those values if required.

To change the port, you just need to configure it at application server level. Please refer to your application server documentation.

If you need to change the context path, you will need to redeploy your Jahia application using this new context path. Refer to your application server documentation if you need some additional information about how to do this. In addition, you will also need to change this context path in WEB-INF/etc/config/jahia.properties.

```
# Web application context path
# By default Jahia is deployed into ROOT context (context path is empty in this case). If deployed into another context, e.g. /jahia this value should be adjusted accordingly (in this case the context path starts with a slash).
jahia.contextPath =
```

### 4.1.4 Permanent move for vanity urls

In Jahia, you can define SEO friendly vanity URLs. There can be more than one URL for the same resource, whereas only one can be set as the default one. Within the `permanentMoveForVanityURL` setting in the `jahia.properties` configuration file you can control access with a non-default vanity URL. Jahia should inform the client that the resource has permanently moved (HTTP status code 301). This is the default behavior. If you set the parameter to `false`, then Jahia will serve the request without changing the URL to the new default one.
4.2 Caching

4.2.1 Introduction

Caches are essential to high-performing web systems such as Jahia in order to be able to avoid recreating dynamic content under large system loads. In the graph above, we can see the basic architecture of the cache sub-system. The cache usage has changed a lot in Enterprise Jahia v6.5 due to now completely persisting the content objects with the Java Content Repository (JCR). The JCR implementation (Apache Jackrabbit) performs well (by using internal caches), so there is no longer any need to use the object cache. Also, the Hibernate cache is playing a minor role, as the number of entities persisted in a relational database have been largely reduced.

The main focus in Enterprise Jahia v6.5 now lies on the Module Cache (previously Container Cache) which is now directly using the Ehcache implementation.

Jahia uses multiple cache layers to optimize the performance of page delivery:

- the browser cache
- front-end HTML caches (skeleton/module cache)
- object caches
- content repository/database caches

Each of these cache layers plays a different role in making sure values are only computed once.
4.2.2 The browser cache layer

While not integrated in Jahia, but in the browser, the browser cache plays a critical role in guaranteeing good performance for the end-user.

For example, Jahia’s usage of the GWT framework makes it possible for AJAX source code to be aggressively cached in the browser cache; therefore making sure we don’t reload script code that has not changed. Jahia also properly manages the browser cache to make sure it does not cache page content that has changed. It also controls expiration times for cached content, so that the browser does not request content that is rarely changed.

4.2.3 The front-end HTML cache layer

Historically, Jahia has had many front-end HTML cache layer implementations. The first was the full-page HTML cache. While very efficient when a page was already available in the cache, it did not degrade very well for pages that had a fragment of the HTML that changed from page to page, or from user to user (for example, displaying the user name on the page). Jahia Enterprise v6.5 combines the sheer efficiency of the embedded full-page cache with the fragment handling of a page.

This new cache implementation is called the “Module Cache” (previously Container Cache) and integrates fragment caching at a module level, making the interaction with templates very natural. Template developers usually do not have to add any markup in order to have their fragments correctly cached. Even when they need to control the fragment generation, this is much easier to do than in previous versions of Jahia.

The HTML code of each module is aggregated on runtime to render the page for the end user. For each module we try to maximize its sharing by building complex keys, taking into account several parameters like roles/templates/etc. That way we can share this rendering with a maximum number of other users that have the same rights.

We also detect cases where more than one parallel request tries to obtain the same fragment, which is not yet cached. In such cases, to not waste resources we let just one request do the work and make the other request(s) wait for it. If rendering the module takes too long, the waiting request gets cancelled with an exception saying “Module generation takes too long due to module not generated fast enough (>10000
ms). As such errors should be taken seriously see chapter “7.4 How to handle module generation timeouts?” for hints how to solve such issues.

### 4.2.4 Object caches

The next layer below the front-end HTML cache sub-systems are the object caches. This layer handles all Java objects that represent sites, users, groups, preferences, etc. It serves as a layer on top of the content repository/database caches in order to avoid reconstructing objects for each model request. This is all handled internally by Jahia and it is only important to interact with these caches if integrators are directly calling Jahia’s API to perform object changes that do not automatically update the caches scheduling/batching.

### 4.2.5 Database caches

The last layer of caches is the database cache layer that makes sure that only minimal interaction with the database happens. Database communication requires object (de-)serialization and network communication so the overhead of a database query can be quite substantial. This layer is, in Jahia, completely handled by the Hibernate ORM layer.

### 4.2.6 Content repository caches

As we moved all content objects to the Java content repository, the object and database caches are used less than in previous Jahia versions. Retrieving content objects from the JCR does not require as many additional caches as before. The content repository optimizes the performance with some internal caches.

### 4.2.7 Ehcache configuration

These files contain fine tunings for the expiration and management storage of cache entries that Ehcache handles:

```
webapps/ROOT/WEB-INF/classes/ehcache-hibernate.xml
webapps/ROOT/WEB-INF/classes/ehcache-jahia.xml
```

or the following files, in case of a clustered setup:
The page [http://ehcache.sourceforge.net/documentation/configuration.html](http://ehcache.sourceforge.net/documentation/configuration.html) explains in detail how the storage and configuration can be done. These settings have to be reported when you switch to a clustered mode (please see the cluster configuration section).

### 4.3 Clustering

#### 4.3.1 Introduction

Deploying Jahia in a cluster is a very powerful way of distributing CPU and memory load to handle larger traffic sites.

A typical Jahia cluster installation is illustrated in the graph below.

Enterprise Jahia v6.5 is largely based on Apache Jackrabbit and thus relies on its cluster capabilities and configurations. See [http://wiki.apache.org/jackrabbit/Clustering](http://wiki.apache.org/jackrabbit/Clustering) for more details. While Jahia 6.5 in cluster only supported the storing of all content in the database, with Jahia 6.6 we have now switched to use the
DataStore (see http://wiki.apache.org/jackrabbit/DataStore for more details of how it works). This way it is now possible and recommended to store large files on a shared file-system, while storing everything in the database is still an option.

Indexes in Jackrabbit have to be local to each cluster node and cannot be shared.

For Jackrabbit, every change made by one cluster node is reported in a journal, which can be either file-based or written to a database. Cluster nodes read the journal and update their state at a configurable synchronization interval.

Ehcache is another component, which needs communication between nodes. We are using JGroups as a communication channel, by default.

Apart from that, Jahia nodes still communicate directly with each other, through direct messaging (e.g. to synch and invalidate caches), as well as the database (e.g. Quartz scheduler).

**BROWSING nodes**

Jahia “browsing” nodes are specialized Jahia nodes that only serve as content publishing nodes. They also interact with portlets to render pages. Using this node specialization allows to separate the browsing load from authoring and background processing loads.

**AUTHORING nodes**

Jahia “authoring” nodes are cluster nodes that can be used to either browse or edit Jahia content. This is the most common usage of Jahia nodes, and therefore it is interesting to have multiple instances of these nodes in order to distribute the load.

**PROCESSING nodes**

In Jahia, long-running tasks such as workflow validation operations, copy and pasting, content import and indexing are executed as background tasks, which can be resource intensive. This way, while these long operations are executed, other nodes are still able to process content browsing and editing requests.
4.3.2 Configuration

It is essential that when you choose to not store binary data in the database that all cluster nodes point to the same shared directory to store binary data in a common file data store. During installation of a cluster node you will be asked to enter the “Path to Data Store files”.

To install your Jahia cluster, you will have to install your cluster nodes one after the other.

- For the first one, proceed as if you were installing a standalone Jahia, excepted that you need to specify that you are installing a cluster at the “Jahia operating mode” step. If you have chosen to use the bundled Tomcat as application server, do not start it at the end of the wizard.

- For the other nodes, execute the wizard again, connecting to same database. This time just specify that the schema does not have to be created. On the screen where you configure your cluster, take care to define a new serverID. If you have already set a node to be the processing server, uncheck the option as only one node can have this role in your cluster.

The cluster configuration is the WEB-INF/etc/config/jahia.advanced.properties file. Here is where you define your cluster nodes under:

```
######################################################################
### Cluster settings  ################################################
######################################################################
```

Even if the configurations are generated in this embedded default file, we recommend that you externalize it. Refer to the “4.11 Configuration files externalization” section to do it. It will ease the maintenance operations, and also allow you to deploy a generic EAR if you are using an application server with a deployment cluster feature. If you use this configuration externalization feature, you can also skip the cluster configuration step in the configuration wizard from the second node. In this case, after having extracted your specific configurations from the first node, just adapt and apply it on the other nodes.
4.3.3  **Sharing webapps**

Web applications need to support clustering on their own to be able to fully work in a clustered Jahia environment.

You have to deploy the webapp on each node of the Jahia cluster. If this webapp stores some data, you will have to ensure that each instance of your webapp shares the same data, and do not duplicate it, otherwise you may encounter some functional issues. Refer to your webapp documentation to perform this.

4.3.4  **Sticky sessions**

If you are using a hardware or software load balancer in front of Jahia to handle the distribution of load among all Jahia nodes in the cluster, you will need to activate "sticky sessions" on your application server and the load balancer. This is required to force an open session to make all requests on the same server for the time of the session.

On Tomcat, this is done by adding a unique `jvmRoute` in the `server.xml` file. Uncomment this part:

```xml
<!-- You should set jvmRoute to support load-balancing via AJP ie :
<Engine name="Standalone" defaultHost="localhost" jvmRoute="jvm1">
-->
```

where `jvm1` is the name of the worker as declared in the load-balancer.

You can also see the reference guide for the configuration of the load balancer on the [Apache](http://tomcat.apache.org/connectors-doc/reference/workers.html) web site.

4.3.5  **Troubleshooting cluster configuration**

Most cluster configuration issues rise up from problems with the configuration of the cluster. Make sure you have put all the IP addresses along with the right ports on the `jahia.advanced.properties` of all the cluster nodes.

4.3.6  **Starting up**

The first time the cluster is started, the processing server must be started standalone. Once the initialization process is finished, you can start the other nodes.
4.4 LDAP

LDAP is an application protocol for querying and modifying directory services running over TCP/IP. Jahia has default connectors to retrieve users/groups from an LDAP server. Jahia supports most LDAP servers right out of the box, including OpenLDAP and ActiveDirectory. It is most commonly used with SSO technologies to provide a seamless experience to end-users. Starting from Jahia Enterprise Edition v6.5, the LDAP configuration is now deployed as a module available in the Enterprise Edition.

The LDAP user and group providers can be configured during the Jahia Installation Wizard by activating “Configure an LDAP user/group provider” option and providing your LDAP server specific parameters.

Please refer to the LDAP module documentation for more details.

4.5 Authentication

4.5.1 Single Sign-On: CAS

The Central Authentication Service (CAS) is a single sign-on protocol for the web. Its purpose is to permit a user to access multiple applications while providing their credentials (such as user id and password) only once.

When the client visits Jahia and wants to authenticate, Jahia redirects the user to the CAS server. CAS validates the client’s authenticity, usually by checking a username and password against a database (such as LDAP or Active Directory).

If the authentication succeeds, CAS returns the client to Jahia, passing along a security ticket. Jahia then validates the ticket by contacting CAS over a secure connection and providing its own service identifier and the ticket. CAS then gives the application trusted information about whether a particular user has successfully authenticated.

Since version 6.6 SP1, Jahia uses Jasig CAS 3.1 Java client, which adds support for CAS 2.0 specification features (http://www.jasig.org/cas/protocol), like multi-tier proxy authentication etc.

The following section gives an overview of configuration options.
4.5.1.1  Jahia

Step 1 - The first file to configure is:

webapps\ROOT\WEB-INF\etc\config\jahia.advanced.properties

Here the values that you would want to change are:

```properties
# Enable CAS authentication valve
auth.cas.enabled = false
# URL prefix of the CAS server
auth.cas.serverUrlPrefix = https://localhost:8443/cas
# Redirect URL to the CAS server for login
auth.cas.loginUrl = ${auth.cas.serverUrlPrefix}/login
# Logout URL to invalidate the user session on the CAS server
auth.cas.logoutUrl = ${auth.cas.serverUrlPrefix}/logout
```

Please note, the CAS server should be accessed using HTTPS protocol. See “4.5.1.2 Configuring ticket validator” section for advanced configuration.

Step 2 - Add the login link in a Jahia template:

In the Studio mode you can use the “Login” component to place a link for the login page into your template.

Alternatively, in your template code you can use the following expression to have a proper link (considering CAS server login page):

```html
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<a href="<c:url value='${url.login}'/>">Log in</a>
```

To add only a logout URL, you can use:

```html
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<a href="<c:url value='${url.logout}'/>">Log out</a>
```
The page [https://wiki.jasig.org/](https://wiki.jasig.org/) contains some information in order to configure your CAS server. The following “How-To” can be also helpful: [http://jira.jahia.org/browse/JKB-22](http://jira.jahia.org/browse/JKB-22).

A good architecture would separate the CAS server from the other application servers.

### 4.5.1.2 Configuring ticket validator

By default, Jahia uses the `org.jasig.cas.client.validation.Cas20ServiceTicketValidator` implementation for ticket validation, which validates Service Tickets in compliance with the CAS 2 (using `/serviceValidate` service endpoint).

The validator implementation is pluggable and can be replaced with e.g. the `org.jasig.cas.client.validation.Cas20ProxyTicketValidator` one (`/proxyValidate` endpoint), which also supports ticket validation using configured list of proxies.

To override the default implementation the following configuration option should be added into `jahia.advanced.properties` file with the ID of the Spring bean, representing the validator (implementation of the `org.jasig.cas.client.validation.TicketValidator` interface), for example:

```properties
auth.cas.ticketValidator=Cas20ProxyTicketValidator
```

And the corresponding bean can be defined in a new Spring file, e.g. `ROOT\WEB-INF\etc\spring\applicationcontext-cas.xml`, or in a custom module as follows:

```xml
<bean id="Cas20ProxyTicketValidator" class="org.jasig.cas.client.validation.Cas20ProxyTicketValidator" scope="prototype">
    <constructor-arg index="0" value="${auth.cas.serverUrlPrefix}" />
    <property name="acceptAnyProxy" value="true"/>
    <property name="allowedProxyChains">
        <value>
            http://proxy1 http://proxy2
            http://proxy3 http://proxy4
        </value>
    </property>
</bean>
```

The bean defines a list of proxy chains and can accept other supported parameters, like `renew`, `encoding`, `proxyCallbackUrl`, `proxyGrantingTicketStorage`, etc.
4.5.1.3 Troubleshooting

If you have errors of the form:

```
org.jahia.exceptions.JahiaException: User message=Cannot validate CAS credentials, System message=Cannot validate CAS credentials, root
```

It is most probably due to your SSL certificate, and that the JVM that runs the Jahia does not recognize it.

Refer to [https://wiki.jasig.org/display/CAS/Solving+SSL+issues](https://wiki.jasig.org/display/CAS/Solving+SSL+issues) for more details.

4.5.2 SSO with Kerberos

Jahia is able to plug-in different authentication policies via HTTP filters and a pipeline of authentication valves. Some filters and valves are provided and can be activated by configuration, like NTLM or the integration of a CAS (Central Authentication Service) server.

We now also provide a filter and valve for integration with SPNEGO (Simple and Protected GSSAPI Negotiation Mechanism) to negotiate with the client and use either Kerberos or NTLM as a fallback. This way a non-interactive login is possible, which takes the user logged in to the operating system running the browser.

Such a secure solution is especially interesting for intranets running with Windows servers. This document describes how to configure such an environment.

4.5.2.1 Prerequisites

**JDK 1.5 Update 7 or later** - otherwise the Active Directory native encryption - rc4-hmac - won’t be supported and you will need to activate Kerberos DES encryption types in the Active Directory on each user account.

**Tomcat 6.0.19 or later** - otherwise negotiation with the client will not work, when also using custom error pages for 401 errors in web.xml. If your Jahia release is shipped with an older Tomcat, then you will have to manually upgrade to the required version.
If using **Windows Server 2008**, then at least **Service Pack 2** needs to be installed (otherwise only simple Kerberos user logons, e.g. via CAS, work, but checks against a Service Principle Name, SPN, will not work, as this one contains slashes, see Microsoft KB article: 951191).

For this guide, we assume that you are using the Windows Active Directory server. If you want to use Kerberos, then all clients and servers need to be joined in the same AD domain. If this requirement is not met, then SPNEGO will fall back to NTLM. It should also be possible to use other directory servers supporting Kerberos and you can take this guide to get some useful information also relevant for alternative environments.

In the guide all terms in angle brackets <> should be replaced by terms fitting your environment or your choice. Notice also that realm names are case-sensitive and recommended to be used in UPPER CASE. Also with Kerberos you will not be able to use IP addresses or `localhost` but rather you will have to use the server name (DNS must be properly set up).

### 4.5.2.2 Set up the Active Directory

A Service Principal Name (SPN) account needs to be created for the Jahia server. Note that this account can’t be used to log in.

1. Start the **Active Directory User and Computers** from the **Administration Tools** menu.
2. Right click on the **Users** repository and select **New > User**.
3. Enter the user information (by example `<your-spn-account>` for user login), press **Next**.
4. Enter the `<password>` and select **Password never expires**, click on **Next** and then on **Finish**.

Now in Windows server 2008 there is an extra step, which is not required in Windows server 2003:

In a console enter the command:

```
setspn -a http/<your.jahia.server.name><your-spn-account>
```

### 4.5.2.3 Create the Keytab file
The Keytab file enables a trust link between the Jahia server and the Key Distribution Center (KDC). This file contains a cryptographic key. The ktpass tool, which comes from the Windows Resource Kit, is used to generate this file (in Windows Server 2008 the tool is already part of the product).

In a console, enter the command:

```
ktpass.exe /out <your-spn-account>.keytab /princ
HTTP/<your.jahia.server.name>@<YOUR.AD.REALM> /pass * /mapuser<your-spn-account>/ptype krb5_nt_principal /crypto rc4-hmac-nt
```

This command will generate the `<your-spn-account>.keytab` file, which has to be copied to a secret place on the Jahia server, which only the Jahia server application can read.

### 4.5.2.4 Create Kerberos configuration file (krb5.conf)

On the Jahia server create the Kerberos configuration file (krb5.conf) and place it somewhere on the Jahia server. In the startup script of the Jahia server you need to add the following parameter to pick up this file:

```
set JAVA_OPTS=%JAVA_OPTS% -Djava.security.krb5.conf=<path>\krb5.conf
```

Here is an example:

<YOUR.REALM> is the same as the domain in caps. With evolving versions you may, for instance, have to change the enctypes settings.

```plaintext
[libdefaults]
    ticket_lifetime = 24000
    default_realm = <YOUR.REALM>
    default_tkt_enctypes = rc4-hmac des3-cbc-sha1 des-cbc-md5 des-cbc-crc
    default_tgs_enctypes = rc4-hmac des3-cbc-sha1 des-cbc-md5 des-cbc-crc
    permitted_enctypes = rc4-hmac des3-cbc-sha1 des-cbc-md5 des-cbc-crc

[realms]
    <YOUR.REALM> = {
        kdc = <hostname.of.your.kdc>.<your.domain>
    }

[domain_realm]
    <your.domain> = <YOUR.REALM>
    <your.domain> = <YOUR.REALM>

[logging]
    kdc = FILE:/var/log/krb5kdc.log
    admin_server = FILE:/var/log/kadmin.log
    default = FILE:/var/log/krb5lib.log
```
4.5.2.5  Create JAAS login configuration file (jaas-login.conf)

On the Jahia server create the JAAS login configuration file (jaas-login.conf) and place it in a secret place accessible for the Jahia server only. In the startup script of the Jahia server you need to add the following parameter to pick up this file:

```
set JAVA_OPTS=%JAVA_OPTS% -Djava.security.auth.login.config=<path>\jaas-login.conf
set JAVA_OPTS=%JAVA_OPTS% -Djavax.security.auth.useSubjectCredsOnly=false
```

Here is an example of the content:

```java
com.sun.security.jgss.accept {
    com.sun.security.auth.module.Krb5LoginModule
    required
    storeKey=true
    keyTab="<path>/<your-spn-account>.keytab"
    doNotPrompt=true
    useKeyTab=true
    principal="HTTP/<your.jahia.server.name>@<YOUR.REALM>"
    isInitiator=false
    debug=false;
};
```

4.5.2.6  Test the SPN account

As it is quite easy to make mistakes in the Kerberos configuration, you should test your configuration with the tools provided by Java, before starting Jahia.

In order to have those tests work, you have to copy your krb5.conf file in your windows system directory and rename it krb5.ini (most often c:\windows\krb5.ini)

Verify that you are able to read the keytab file:

```
%JAVA_HOME%/bin/klist -k FILE:<path>/<your-spn-account>.keytab
```

and

```
%JAVA_HOME%/bin/kinit -k -t FILE:<path>/<your-spn-account>.keytab
HTTP/<your.jahia.server.name>@<YOUR.REALM>%JAVA_HOME%/bin/klist
```

4.5.2.7  Set up the browser
4.5.2.7.1 Internet Explorer (min 5.0.1)

1. In Internet Explorer, click **Internet Options** on the **Tools** menu.

2. Click on the **Advanced** tab, click to select the **Enable Integrated Windows Authentication (requires restart)** check box in the **Security** section, and then click **OK**.

3. Click on the **Security** tab, click to select **Local Intranet**, then click on **Sites**, lastly click on **Advanced**.

4. Enter https://<your.jahia.server.name> and validate it by clicking on **Add** and **OK**.

5. Restart Internet Explorer.

4.5.2.7.2 Firefox (min 0.9)

1. In Firefox, enter about:config as the url and click on **Go**.

2. On the line network.negotiate-auth.trusted-uris, right-click on **Modify** and enter https://<your.jahia.server.name>

4.5.2.8 Activate the SPNEGO HTTP filter and authentication valve in Jahia

Kerberos authentication in Jahia 6.6 is only supported with Enterprise Jahia. To activate it, you need to set the auth.spnego.enabled property in jahia.advanced.properties to true and restart the server.

4.5.2.9 Related links

Here are some links for further reading and troubleshooting:

- Kerberos with Java Troubleshooting
- Advanced Security Programming in Java SE ... Single Sign-On
- Developing with Kerberos in Java
- Kerberos Authentication problems – Service Principal Name (SPN) issues
- Setting up CAS with SPNEGO Authentication Handler

4.5.2.10 Tips and Tricks
First of all, we recommend you to take a look at blogs.technet.com/askds/archive/2008/03/06/kerberos-for-the-busy-admin.aspx for information about how Kerberos works.

This Article shows how to resolve common errors

**4.5.2.10.1**  \*ERROR \*[ErrorLoggingFilter\*] -  \*Unexpected exception occurred\*

```
ERROR [SpnegoParser] - Failed to parse: 32
INFO [SpnegoParser] - [0,APPLICATION_CONSTRUCTED_OBJECT,0x4e] Expected type identifier
INFO [SpnegoParser] - Expected length 84 mismatch against actual 30
INFO [SpnegoParser] - [2,OID,0x4c] Expected oid identifier
ERROR [ErrorLoggingFilter] - Unexpected exception occurred
java.lang.NullPointerException
```

This error means that the Kerberos authentication is not done on the client browser.

Resolution: Check your Kerberos configuration with klist and kinit tools. Look at support.microsoft.com/default.aspx to activate Kerberos event logging

**4.5.2.10.2**  \*KrbException: Clock skew too great (37)\*

This error occurs when there is more than 5 minutes between the Kerberos’ domain controller and the Jahia server times.

Resolution: Check time and time zone.

**4.6 Document converter**

Jahia Document Converter Service delegates conversion tasks to an OpenOffice/LibreOffice instance, either to a local one or a remote service. To use the converter service you need OpenOffice/LibreOffice v3 or higher installed (the latest stable 3.x version is recommended). Further in this document, we refer to OpenOffice or LibreOffice as “OpenOffice” for the sake of simplicity.

In order to enable the service the following setting should be set to true in the jahia.properties file:

```
# Set this to true to enable the document conversion service
documentConverter.enabled = true
```
4.6.1 LocalOpenOffice instance

The converter service is capable of creating an OpenOffice process and using it, in case Jahia and OpenOffice are located on the same machine.

In such case, the converter service starts a local instance of the OpenOffice service for processing conversion tasks.

The configuration in this case is pretty simple: a service needs to be enabled (see above) and a path to the OpenOffice folder has to be provided in the jahia.properties file:

```
# Set this to true to enable the document conversion service
documentConverter.enabled = false
# The filesystem path to the OpenOffice
# Usually for Linux it is: /usr/lib/openoffice
# for Windows: c:/Program Files (x86)/OpenOffice.org 3
# and for Mac OS X: /Applications/OpenOffice.org.app/Contents
documentConverter.officeHome = /usr/lib/openoffice
```

For an advanced configuration, the LocalOfficeManagerFactory supports specifying a port number, template profile directory, task execution timeout, task queue timeout and a maximum number of tasks per process. The default configuration is located in the org/jahia/defaults/config/spring/applicationcontext-doc-converter.xml file packaged inside the JAR WEB-INF/lib/jahia-impl-<version>.jar and can be overridden from your custom module if needed.

4.6.2 RemoteOpenOffice service

The converter service is capable of using an OpenOffice process started as a service on a remote machine.

This connection is configured as given below in the snapshot of the applicationcontext-doc-converter.xml file:

```
<bean id="DocumentConverterService" class="org.jahia.services.transform.DocumentConverterService"
     init-method="start" destroy-method="stop">
    <property name="enabled" value="true"/>
    <property name="officeManagerBeanName" value="remoteOfficeManagerFactory"/>
</bean>
```
OpenOffice in this case should be started as a service on the 192.168.1.101 machine.

A sample command for starting OpenOffice as a service looks like:

```
soffice -headless -accept="socket,host=192.168.1.101,port=19001;urp;" -nofirststartwizard
```

More details can be found on the [JODConverter Web Site](#), including the HowTo for Creating an OpenOffice.org Service on Windows and Creating an OpenOffice.org Service on Unix-like systems.

### 4.7 Document viewer

As of version 6.6 SP1 Jahia offers a built-in support for previewing various types of documents (PDF, Office, etc.) as a SWF flash using a player in a Web page. The direct conversion to flash is available for PDF documents only. To have a preview for non-PDF files (Microsoft Office, OpenOffice etc.) the document converter service (see section “4.6 Document converter” above) should be enabled to perform an intermediate conversion of documents to PDF files.

The viewer service requires the pdf2swf utility (from SWFTools) to be installed. The installation guidelines are available on the corresponding Wiki pages.

The following two configuration parameters in the `WEB-INF/etc/config/jahia.properties` file are responsible for enabling and configuring the document viewer service:
jahia.dm.viewer.enabled = false
# Viewer service requires the pdf2swf utility (from SWFTools) to be installed
# The following specifies the path to the pdf2swf executable file
# Usually for Linux it is: /usr/bin/pdf2swf
# for Windows: C:/Program Files (x86)/SWFTools/pdf2swf.exe
# If the SWFTools installation folder is present in your PATH, you can
# specify only the executable name here
jahia.dm.viewer.pdf2swf = pdf2swf

The jahia.dm.viewer.pdf2swf parameter should contain an absolute path to the pdf2swf executable or, in case the corresponding folder is included into the PATH environment variable, just the executable name, i.e. pdf2swf.

### 4.8 Document thumbnails

Starting with Jahia 6.6 SP1 we are pleased to offer an out-of-the-box support for automatic creation of image thumbnails for uploaded documents that significantly improves the usability and user experience when working with Jahia Document Manager or document-related components.

The service is enabled by default for all PDF documents. A thumbnail is automatically created for the first page of an uploaded document.

To have thumbnails for non-PDF files (Microsoft Office, OpenOffice etc.) the document converter service (see section “4.6 Document converter” above) should be enabled to perform an intermediate conversion of documents to PDF files.

The following entry in the WEB-INF/etc/config/jahia.properties file is responsible for enabling/disabling the document thumbnails service:

```
# Document Thumbnails Service
#
# Document thumbnails service enables automatic creation of thumbnail images for uploaded documents.
# The direct creation of a thumbnail is available for PDF files only.
# In order for this service to work with non-PDF files a document converter service (see section above) should be enabled to perform an intermediate conversion of documents to PDF files.
# The following enables/disables the document thumbnails service
jahia.dm.thumbnails.enabled = true
```
4.9 Video thumbnails

For an improved media experience Jahia 6.6 SP1 offers a possibility of automatic thumbnail generation for uploaded video files.

The video thumbnails service requires the ffmpeg utility (ffmpeg.org) to be installed.

The following two configuration parameters in WEB-INF/etc/config/jahia.properties file control the service:

```
# Video thumbnails service enables automatic creation of thumbnail images
# for uploaded video files.
# Set this to true to enable the video thumbnails service
jahia.dm.thumbnails.video.enabled = false
# Video thumbnails service requires the ffmpeg utility to be installed
# The following specifies the path to the ffmpeg executable file
# Usually for Linux it is: /usr/bin/ffmpeg
# for Windows, for example: C:/Program Files (x86)/ffmpeg-20120503-git-c1fe2db-win64-static/bin/ffmpeg.exe
# If the ffmpeg/bin folder is present in your PATH, you can
# specify only the executable name here
jahia.dm.thumbnails.video.ffmpeg = ffmpeg
```

The jahia.dm.thumbnails.video.ffmpeg parameter should contain an absolute path to the ffmpeg executable or, in case the corresponding folder is included into the PATH environment variable, just the executable name, i.e. ffmpeg.

4.10 Image service

The Jahia Image Service is here to manipulate images from Jahia itself. For licensing reasons the service is by default using a java native API named imageJ, but this is not a really powerful API nor really efficient.

So if you want to boost the quality of your thumbnails or the result of your other image manipulation operations, Jahia allows you to define the path to your ImageMagick installation so that we can use it instead of the imageJ API.
4.10.1 **How-to Install ImageMagick?**

Follow the instructions for your system on the [Image Magick Binary Releases page](#). Once ImageMagick is installed, modify your `jahia.properties` file to activate ImageMagick instead of the `imageJ` API.

```
# Image conversion Service
imageService = ImageJImageService
imageMagickPath = /usr/bin:/usr/local/bin:/opt/local/bin
```

4.11 **Configuration files externalization**

4.11.1 **Feature functional description**

The files externalization will allow:

- The isolation of the Jahia application as a bundle. This same bundle could be deployed in an identical manner over an environment using the application server’s deployment tools.

- The ability to configure cluster nodes independently from one another.

This feature is meant to ease the maintenance or e.g. the process of deployment in a clustered environment.

4.11.2 **Feature technical description**

The feature will externalize the following files. Those files contain most of Jahia’s settings: licensing, clustering, LDAP, etc.

- `jahia.properties`
- `jahia.advanced.properties` (Enterprise Jahia only)
Configuration and fine tuning Guide Enterprise Jahia v6.6.2

- applicationcontext-*.xml (these files are optional, thus not externalized by the Installer)
- license.xml

CONFIGURATION

4.11.3 jahia.properties / jahia.advanced.properties

In order to externalize the jahia.properties / jahia.advanced.properties files, the Jahia settings (Properties) have the following lookup order with later resources overriding the previous, if they are found:

- classpath:org/jahia/defaults/config/properties/jahia*.properties: This is the standard properties file, as provided by the Jahia developers. This is the default file from which the values are going to be merged. You will never need to modify this file as it may be overridden in multiple locations.

- /WEB-INF/etc/config/jahia.properties: This is the standard location for the configured properties file, located in the Jahia Web application folder. Most of the properties in this file are commented out and loaded from the default jahia.properties file located in the above classpath. You may of course uncomment any property you need to change, but we recommend that you leave them commented out if you don’t change them, as this will make it possible for Jahia updates to deliver new default properties (or simply new values).

- classpath:org/jahia/config/jahia*.properties: This file is fetched from the CLASSPATH matching the following pattern: org/jahia/config/jahia*.properties. If this pattern is found in the CLASSPATH directories, this file will overwrite the parameters from the properties file located in the Jahia Web application folder (i.e. the CLASSPATH needs to be updated with your custom directory).

- file:${jahia.config}: This is a lookup for a file, specified with the Java system property "jahia.config" (e.g. -Djahia.config=/opt/jahia/custom.properties)

4.11.4 Spring bean definitions

Spring beans can also be externalized.
Configuration and fine tuning Guide Enterprise Jahia v6.6.2

- **classpath*:org/jahia/defaults/config/spring/**/applicationcontext-*.xml**: These are the default spring files containing the Jahia Spring beans.

- **WEB-INF/etc/spring/applicationcontext-*.xml**: If you put files at this location they will override or expand the default Spring configuration files.

- **classpath*:org/jahia/config/**/applicationcontext-*.xml**: Similar to jahia.properties, Spring is going to fetch the following pattern in the CLASSPATH: org/jahia/config/**/applicationcontext-*.xml. (i.e the CLASSPATH needs to be updated with your custom directory). Spring Beans present in those files will overwrite the Jahia default ones. (e.g. CUSTOMCLASSPATH/org/jahia/config/test/applicationcontextcustomBeans.xml file)

### 4.11.5 Licence file

Finally, this feature allows the externalization of the License file. The lookup sequence is listed below and it stops on the first found license file:

- **file:${jahia.license}**: This is a lookup for a file, specified in with the Java system property "jahia.license" (e.g. -Djahia.license=/opt/jahia/license-pro.xml)

- **classpath:org/jahia/config/license*.xml**: Similar to the other ones, Spring is going to fetch the following pattern in the CLASSPATH: org/jahia/config/license*.xml. (i.e. the CLASSPATH needs to be updated with your custom directory).

- **WEB-INF/etc/config/license.xml**: the standard location of the Jahia license file.

### 4.12 Error and thread dump directories

#### 4.12.1 Error file dumper server

Jahia’s error file dumper service is used to automatically create reports when an error occurs. The location of those files is by default: ${java.io.tmpdir}/jahia-errors (for Apache Tomcat this is by default under <tomcat-home>/temp/jahia-errors).
Since Jahia 6.6 SP2 it is possible to override that folder location with a system property named jahia.error.dir, e.g. by adding -Djahia.error.dir=/var/logs/jahia/errors to the JVM options (CATALINA_OPTS for Apache Tomcat).

### 4.12.2 Thread dumps

Jahia Tools Area provide a functionality (see chapter “6.4.1 System and Maintenance”) to perform single thread dumps as well as a series of thread dumps into a file. The location of those files is by default: ${java.io.tmpdir}/jahia-threads (for Apache Tomcat this is by default under <tomcat-home>/temp/jahia-threads).

Since Jahia 6.6 SP2 it is possible to override that folder location with a system property named jahia.thread.dir, e.g. by adding -Djahia.thread.dir=/var/logs/jahia/threads to the JVM options (CATALINA_OPTS for Apache Tomcat).
5 Fine Tuning

After having implemented all your templates, and you are satisfied with your website, there maybe some modifications to be done in order to enhance the performance of your server.

Before changing any values on your production server, you should ask yourself the following questions:

- How many editors do you have working simultaneously on the system?
- What is the number of authenticated users that can log into your system (in general, not necessarily at the same time)?
- What is the number of pages that you have in your system, and if they contain a lot of resources (PDF files, etc)?

As a general rule, in order to test the performance of any system running Jahia, here are the issues that need to be addressed:

1. Tomcat and the amount of virtual memory (typically the Xmx part in the catalina.bat file)
2. The database and its default settings
3. Jahia properties configuration

The values given here are the high values and have been tested, but that does not mean that this corresponds to the values you should set. The way to find the proper values that will fit your system is to increase progressively, and set the values here one at a time (except for the server.xml and database pool size, they go by pair). Then run a load test (bearing in mind the answers to the questions at the beginning of this document) to see if it corresponds to your expectations.

5.1 Tomcat

5.1.1 bin/catalina.sh

We usually recommend raising the amount of virtual memory (Xmx parameter) in your bin/catalina.sh file to 2048, 4096 or even higher.
It is not necessarily true that the more virtual memory you give to your system, the faster you get, as sometimes having a lot of memory can benefit you in the beginning, but then garbage collection may take longer, which will make your server unavailable for a longer period of time.

### 5.1.2 `conf/server.xml`

Here you can increase the amount of `maxThreads` as well as the amount of `acceptCount`. These settings are the ones handling the connections to your database. `maxThreads` is the maximum number of threads processing requests in Tomcat, whether serving pages from Jahia cache or not. If this one is exceeded, then errors will be sent to the client. In case you need to modify those settings, do it in the HTTP connector, the AJP connector or both, depending how you access your application server.

On the other hand, raising this number may not bring the wanted effect. For example, if you leave `maxModulesToGenerateInParallel` at 50 in `jahia.properties`, as no more than that number will do the real work, while the other threads will queue. But we will talk about that configuration in chapter “5.3 Module generation queue”.

### 5.2 Database

As we have increased the amount of threads in Tomcat, we have to increase the pool of connections to the database. We usually have the following values for MySQL:

```plaintext
key_buffer = 384M
max_allowed_packet = 512M
table_cache = 512
sort_buffer_size = 2M
read_buffer_size = 2M
read_rnd_buffer_size = 8M
myisam_sort_buffer_size = 64M
thread_cache_size = 8
query_cache_size = 128M
thread_concurrency = 8
#lower_case_table_names=1
max_connections=3000
```
5.3 Module generation queue

The queue can be configured in:

```java
webapps/ROOT/WEB-INF/etc/config/jahia.properties
```

or in your externalized `jahia.properties` file.

Here you should increase the following value for your server:

```java
maxModulesToGenerateInParallel = 100
```

This value controls how many parallel threads will be allowed to start rendering modules not coming from cache, meaning that they will open JCR and DB connections to obtain the content from there.

`maxModulesToGenerateInParallel` in `jahia.properties` should not be bigger than the `maxThreads` value in `server.xml`

The factor between `maxModulesToGenerateInParallel` and `maxThreads` (HTTP or/and AJP) should be around 2-3, meaning:

`maxThreads = maxModulesToGenerateInParallel * (2-3)`

For example:

```
maxModulesToGenerateInParallel = 100, maxThreads = 300
maxModulesToGenerateInParallel = 200, maxThreads = 600
```
5.4 Operating mode

Setting the operating mode to “production” enhances the performance of your server as when set to “development”, we check more often, which resources (templates, rules) on the server changed in order to redeploy or reinitialize them. The Development Mode will also write more debug information or not compress certain data in order to have it readable.

The “RemoteServer” Mode provides similar performances as the Production Mode, but deactivates some authoring features, as you are not supposed to perform authoring actions directly on this server.

This mode is configured in WEB-INF/etc/config/jahia.properties:

```
# This setting can be used to activate particular profile:
#  - development
#  - production
#  - distantPublicationServer
operatingMode = development
```

5.5 JCR DataStore garbage collector

In Jahia releases prior 6.6.1.5 the Maintenance.DataStoreGarbageCollectorJob was automatically scheduled to run every day. This job is deleting binary property data (e.g. uploaded files) bigger than 1 kB, which is no longer referenced by any node in any workspace (live, default and versioned).

The current implementation of this job is very resource intensive as it actually loads all nodes in the system and sets the last modified date on the binary data. On large installations this has led to performance penalties and even OutOfMemory exceptions. As a better implementation is not yet available, we decided to deactivate automatic scheduling of this job from Jahia 6.6.1.5 onwards.

Also we recommend customers running on earlier Jahia versions to deactivate the job. This can be done in the tools area under Background job administration (http://localhost:8080/tools/jobadmin.jsp). You need to select the checkbox Show actions in the top-right box and then click on the red X icon in the row of the Maintenance.DataStoreGarbageCollectorJob. Furthermore you need to create on all cluster nodes a
file under .../WEB-INF/etc/spring/ for instance called applicationcontext-scheduler-jobs-disableDGC.xml with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:util="http://www.springframework.org/schema/util"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
   http://www.springframework.org/schema/util
   http://www.springframework.org/schema/util/spring-util-3.0.xsd">
   <bean id="DataStoreGarbageCollectorJob" parent="jobSchedulingBean" lazy-init="true">
      <property name="jobDetail">
         <bean class="org.springframework.scheduling.quartz.JobDetailBean">
            <property name="name" value="DataStoreGarbageCollectorJob" />
            <property name="jobClass" value="org.jahia.services.scheduler.DataStoreGarbageCollectorJob" />
            <property name="group" value="Maintenance" />
            <property name="description" value="Runs JCR DataStore Garbage Collector that deletes no longer used entries" />
         </bean>
      </property>
   </bean>
</beans>
```

As this job is no longer triggered automatically now you need to plan running a maintenance task manually for instance every month or even just once per quarter at times when the processing node is not under stress. The job can be triggered manually from the tools area: JCR DataStore garbage collection (http://localhost:8080/tools/jcrGc.jsp).

5.6 Storing binary files

During the installation process when setting the database connection settings an option allows you to either check or uncheck the box “Store binary data in the database”.

According to the Apache Jackrabbit wiki (http://wiki.apache.org/jackrabbit/DataStore#File_Data_Store), “FileDataStore is usually faster than the DbDataStore, and the preferred choice unless you have strict operational reasons to put everything into a database.”

We recommend you leave the “Store binary data in the database” checkbox unchecked.
You cannot switch between the store implementation at a later time, unless one makes an export-import of the repository data. When using a FileDataStore in cluster, a shared file system needs to be used, where all cluster-nodes point to.

The path to the FileDataStore should not be located under WEB-INF as this can very much increase server startup and JSP compilation times. While the installer lets configure that path only for cluster installation, you can manually modify it for standalone servers.

To do that:

1.) Stop the server
2.) Move `/WEB-INF/var/repository/datastore` to a different location not under WEB-INF
3.) Edit `/WEB-INF/etc/repository/jackrabbit/repository.xml` and change the path value within the DataStore element on the bottom of the file to point the new location
4.) Start the server

### 5.7 Increasing bundleCacheSize

Another recommendation is to increase the value of the bundleCacheSize settings. There are three PersistenceManagers using bundle caches: one for default workspace, one for live workspace and one for the version space. Each is on default just 8MB small. For large production systems you should increase the values, so that they together occupy around 1/10\(^\text{th}\) of the JVM maximum heap space.

More information about this, can be found at this link: [https://www.jahia.com/community/extend/technical-blog/performance-sizing-the-jackrabbit-bundle-cache-properly](https://www.jahia.com/community/extend/technical-blog/performance-sizing-the-jackrabbit-bundle-cache-properly)

At that linked article you get some information how to read the bundleCache related log output in the console. Based on the miss to access ratio in your environment you can decide whether you should dedicate more or less memory to either default, live or the version bundle cache. In cluster it also depends whether a cluster node is used for authoring/processing content or just for serving the published live content. So you should adapt the setting to the cluster node role, and if for instance it is just used to serve live content, then the live bundle cache should get most of the 1/10\(^\text{th}\) of heap.
Usually the versioning bundleCache can be 2-4 times smaller than the default/live bundleCache, but it depends on the environment and usage, so you can decide on your own by checking the bundleCache lines in the console output.

Let's take as example that we have a system using 3GB of heap, we may set the bundleCache to the following values: default: 128MB, live: 128MB, version: 64MB.

To increase the bundleCacheSize parameter of the different PersistenceManagers you have to do this in the following files:

For the version bundleCache open WEB-INF/etc/repository/jackrabbit/repository.xml and in the section Versioning/PersistenceManager add the following configuration:

```
<param name="bundleCacheSize" value="64" />
```

For the default bundleCache open WEB-INF/var/repository/workspaces/default/workspace.xml and in the Workspace/PersistenceManager edit and uncomment the following configuration:

```
<param name="bundleCacheSize" value="128" />
```

For the live bundleCache open WEB-INF/var/repository/workspaces/live/workspace.xml and in the Workspace/PersistenceManager edit and uncomment the following configuration:

```
<param name="bundleCacheSize" value="128" />
```
6 Monitoring

There are multiple ways of monitoring a Jahia installation's behavior in real-time; we will present it in this chapter.

Also, if you have identified an issue with a Jahia installation and want to communicate it back to us, we have a section below that describes what is required to efficiently provide us with the data that will help us assist you in a timely manner.

6.1 Stack trace dumps

Stack trace dumps are a very useful way of figuring out exactly what the JVM is executing at a specific point in time. Basically the JVM has a way of dumping onto the console output a list of all the threads currently executing with, for each thread, a detailed stack trace of where in the code each thread is currently. More information on stack traces is here:

http://java.sun.com/developer/technicalArticles/Programming/Stacktrace/

If errors occur, Jahia automatically generates thread dumps. To create thread dumps on demand you can also use the Thread State Information Tool delivered by Jahia (see chapter 6.4 Tools), which can also automatically create multiple thread dumps in an interval. If you want to analyze the thread dumps created by Jahia with a tool, you may have to switch the useJstackForThreadDumps in jahia.properties to true, provided that the jstack command (from Oracle Java Platform SE package) is available in your PATH. That allows you to generate more accurate thread dumps (although the generation process is slower), and it is guaranteed that in this case a dump can be read by any thread dump analyzer tool available on the market.

You may also trigger such standard thread dumps manually in a Java standard way. Performing a stack trace dump is different on various platforms:

6.1.1 UNIX

On UNIX platforms you can send a signal to a program by using the kill command. This is the quit signal, which is handled by the JVM. For example, on Linux you can use the kill command -QUIT process_id, where
process_id is the process number of your JVM. Don't be alarmed by the fact that the command is called "kill", despite the name, all this command will do is perform a stack trace dump and the JVM will continue executing. Alternatively you can enter the key sequence <ctrl>\ in the window where the JVM was started (this works only if the java process is running in foreground in this window, not if you are doing a tail on the log file). Sending this signal instructs a signal handler in the JVM to recursively print out all the information on the threads and monitors inside the JVM.

6.1.2 Windows

To generate a stack trace on Windows 95, or Windows NT platforms, enter the key sequence <ctrl><break> in the window where the Java program is running, or click the Close button on the window.

The output of the stack trace will go to the console output, so under Windows it will be displayed in the JVM window, and under UNIX it will be in tomcat/logs/catalina.out.

Once the dump has been performed, you can look for threads that are blocked, or see the amount of threads that are performing some operations, which might not be expected.

6.1.3 Tools

You can also use the JVM "jstack <pid>" tools if you are using an Oracle SUN JVM. This will render the thread dump in your console or in a file, if you want.

6.2 Memory dumps

In order to analyze the memory usage of a JVM, it is possible to perform memory dumps that can then later be analyzed to determine if the application is behaving as expected, or if a data structure is eating up too many resources.

There are two ways of performing memory dumps with the JVM:

- via Java VM parameters:
  - -XX:+HeapDumpOnOutOfMemoryError writes heap dump on OutOfMemoryError (recommended)
  - -XX:+HeapDumpOnCtrlBreak writes heap dump together with thread dump on CTRL+BREAK (recommended)
6.3 Java profilers

The most powerful tool to analyze in real-time what is going on inside a Jahia installation is a JVM profiler. There are multiple tools that exist, but we recommend YourKit Java Profiler, which is a commercial tool that can be used even in production with lesser performance impacts.

You can find a more extensive list of profilers here:

- Free Profilers
- Commercial Profilers

6.4 Tools

Enterprise Jahia v6.6 provides several tools as JSP's files that you can call to run certain commands on your server (activate Maintenance Mode, get information about the system, display thread dump, view the cache, etc.)

Those tools are password protected by a security realm with the Jahia Tool Manager user. Its username and password are configured during the installation wizard (defaults are: jahia/password).

The list of tools can be found after Jahia installation at http://localhost:8080/tools (adapt the URL, if you use other domains, ports or server contexts):
6.4.1 System and Maintenance

The tools under system and maintenance allow you to see the status of your platform. They also allow you to put your system under maintenance. This mode will display a nice page of information while you update your server (Jahia needs to be running, otherwise use a HTTP server in front to deliver a static maintenance page). The JSP pre-compiler should be run after deploying new releases of modules in order to pre-compile the JSPs, so that this will not happen once the server is already under load.

- **System information** – gather system information to analyze all the current settings.
- **Thread state information** – create one or multiple thread dumps.
- **Memory information** – show the current memory status.
- **System maintenance** – set system in Maintenance code to block access.
- **JSP pre-compilation** – trigger the precompiling of JSPs after deployment.

6.4.2 Logging

These tools are here to manage your log4j configuration (change the log level for certain categories) over a user interface. Notice that these settings then only apply to the current runtime – they are not persistent, so on the next server startup the settings will be taken from log4j.xml.

You can also control the activation of the error file dumper.

- **Log4j administration** – tool to change log levels immediately.
- **Error file dumper** – ability to switch on/off error dumping to files.

6.4.3 Administration and Guidance

The tools in this section give you an overview of the currently running or scheduled background jobs in Jahia, allow you to re-index the spell-checking dictionary and run SQL statements or Groovy scripts. As you may update a runtime database with that, you have to be very cautious and do backups before manipulations.

- **Background job administration** – view active or scheduled background jobs.
- **Search engine management** – trigger the update of the spell checker dictionary index.
- **DB query tool** – run SQL queries/updates using a connection from the configured Jahia data source,
6.4.4 Data

The data tools contain a JCR repository browser that can be really helpful to browse your JCR content and have all data displayed in a particular node. You can also run JCR queries and Groovy scripts within a JCRTemplate.

Furthermore, there are housekeeping tools to clean-up the version history and to run the data store garbage collector.

Some tools display information about the installed modules, definitions and render filters.

- **JCR repository browser** – browse the JCR content tree in a simple UI.
- **JCR query tool** – run JCR queries with SQL-2, XPath or SQL syntax.
- **JCR console** – paste Groovy code to execute in a JCRTemplate against the JCR repository.
- **JCR DataStore garbage collection** – run the JCR DataStore garbage collector.
- **JCR version history management** – perform cleanup tasks on the version store.
- **Installed modules browser** – display details of installed modules.
- **Installed definitions browser** – display details of installed node/property definitions.
- **Render filters** – display details of installed render filters.

6.4.5 Cache

You can also access the content of the HTML caches if needed by accessing the following tools.

- **Cache management** – flush certain caches.
- **Modules** – view and search module cache elements.
- **Dependencies** – view and search module dependency cache elements.

6.4.6 Miscellaneous Tools

These are various tools that could not be classified into the other categories.

- **Password encryption** – tool to encrypt passwords.
- **Document converter** – convert documents into other formats if the conversion service is active.
6.5 Other Issues

The best way to get support for your issues is to contact us for a support agreement. Please see this page for more information.

If you have a commercial support contract, you will get your own space to submit issues that will be handled according to our SLA. Otherwise, you can report issues to the general JIRA projects, but here there will be no guarantee as to how and when the issue will be handled. When submitting an issue to our JIRA Issue tracker, make sure you include as much information as possible, including:

- A detailed description of your environment with the version number and patches (J2EE server, JDK, OS) as well as memory and architecture (32-bit, 64-bit).
- A detailed (or complete) log file, including date and times at which the problem occurs, to be able to corroborate with log file.
- A list of steps to reproduce the problem (if not random).
- A stack trace dump or, in case of performance issues, multiple thread dumps in intervals (see chapter 6.1 Stack trace dumps).
- If dealing with an OutOfMemory issue, please include a memory dump (see chapter 6.2 Memory dumps).

As a basic rule, we also prefer to have too much information than too little.
7 Frequently asked questions (FAQ)

7.1 How to backup Jahia?

Backing up your system is useful in many cases as it minimizes the risk of losing all of your data, whether it is on the database or server side.

**Database**

A database dump contains a record of the table structure and/or the data from a database, and it is usually in the form of a list of SQL statements. A database dump is useful for backing up a database so that its contents can be restored in the event of data loss (or in our case reusing an environment). It can be performed anytime (even when the Jahia server is running), but it is usually preferable to shut down your Jahia before dumping your database.

There are many software products (proprietary or Open Source) that can perform a database dump for all types of databases. Here, we will use the example of MySQL.

```
mysqldump -urootUser -p jahia6_v1 > jahia6_v1.sql
```

Your database configuration is located under `webapps/META-INF/context.xml` respectively if using Tomcat after the first start, this file is copied to `conf/Catalina/localhost/ROOT.xml` (file name depends on the chosen context name). For consistency reasons you should keep these two files in sync and do modifications on both sides.

**Jahia core data files**

If you have chosen the filesystem storage for binary content in your configuration, then you should also back up all the folders under the following one:

```
webapps/ROOT/WEB-INF/var/repository/
```
If during the configuration wizard you’ve chosen filesystem-based binary storage and alternative location for the datastore folder (in a clustered installation), you should backup also that folder.

**Templates**

Your module templates are all situated under the folder:

```
webapps/ROOT/modules
```

and are shared among all of your virtual sites. This folder should also be backed up if your templates are not already saved in a version control system (in this case, if the deployed templates are not always the last development version, you just need to keep which version of your templates you deployed in your server).

**Web applications/portlets**

If you have no web additional applications (or portlets) used inside your Jahia server, you can skip this part. All default Jahia webapps are embedded inside the Jahia application and do not require any backup.

All the additional webapps you may have deployed will be located here:

```
webapps/
```

Each webapp directory name contains the name of the application. For example, if you have added a “Time Reporting” the directory name will be “TimeReporting”.

Webapps are accessible across any virtual sites.

You can backup all web applications or only the one you use. If you installed some third party portlets, be sure to check on their respective documentation. Depending on wether or not the webapp is storing information, the way you backup the webapp will be different. If the webapp stores nothing, you can either backup the .war file you had used to deploy the portlet, or the subfolder of “webapps/” in which the webapp has been deployed. If the webapp stores some data, you will also have to backup it.

**Configuration files**

All major configuration files are situated under the folder below:
or in the META-INF\spring folders of the modules. These folders contain all of the information regarding your LDAP, SSO database storage, and more specifically the jahia.properties and jahia.advanced.properties files.

If you have chosen to externalize your configurations, the backup will be easier as you will only need to backup your additional files. Refer to the “4.11 Configuration files externalization” section if you want to know how to externalize your configurations.

If you are under UNIX, for regular backup of your Jahia data, you can create a script file and run it through a Cron job. A typical example of this script could be:

```bash
DAY=`date +%u`
/home/backup/tomcat_$DAY.tar.gz /home/jahia/tomcat/ #list of folders to copy
```

### 7.2 How to restore an environment from a backup?

#### 7.2.1 Restore your database dump

Please refer your database documentation for specific instructions of how to perform this.

#### 7.2.2 Reinstall Jahia

During the configuration wizard, instead of connecting to a new empty database, connect to your newly restored database. Uncheck the option to create the tables inside this database.

Take care to specify the same value as you did for your former installation regarding the storage of the binaries (inside the database or on the filesystem).
If you do not remember, open `webapps/ROOT/WEB-INF/etc/repository/jackrabbit/repository.xml` and check the `DataStore` element, which could either be a `DbDataStore` or a `FileDataStore`.

Do not start the application server at the end of the install process.

### 7.2.3 Apply your specific configurations on your new installation.

- If you had externalized your configurations, you just need to reapply your additional configuration files.

- Otherwise, the need to edit the different configuration files you had customized and reapply your specific configurations. When going from one service pack to another, some configuration files may have changed (see the change log for this). This means that when migrating from an old version of Jahia to a newer one, simply copying and pasting the configuration files may not work, or may lead to startup errors. You will save time if you document your specific configurations, so that you can easily apply it on a new installation.

### 7.2.4 Deploy your templates and modules

Deploy your templates set(s) and modules. Do not forget that deploying simply your custom modules is not enough; you also need to deploy the Jahia standard modules you are using, as those modules are deployed automatically only when installing Jahia in the Development Mode.

### 7.2.5 Restore the binaries stored on the filesystem

If you have chosen to store the binaries in your database, just skip this step.

Copy your `WEB_INF/var/repository/` folder from your backup to your new installation. You will have the following structure:

```
repository
  |_________datastore
  |_________index
  |_________version
  |_________workspaces
  |           |___default
  |           |     |____index
```
If you have chosen an alternative location of the datastore folder during the Jahia configuration wizard (cluster installation), please restore it at the appropriate location.

Remove the 2 “lock” files. If possible, we also recommend you to also remove the 3 “index” folders. Those folders store the jackrabbit indexes, which will be regenerated at first startup if missing. Regenerating it will improve the performances, but this operation will take a variable amount of time, depending on the amount of data you have. If you are doing an emergency restore of a production server, you can keep the former indexes to save time.

### 7.2.6 Restart the Jahia server

For the last step you must restart your reinstalled Jahia application.

### 7.3 Modifying the Logging Level

The following instructions apply to modify logging levels permanently. If you want to only change the level for a short time, you can use the runtime tool, described in chapter 6.4.2 Logging.

When you install a release of Jahia, the logging level is set to the minimum to avoid slowing down Jahia. If you need to increase it for debugging purpose, you need to modify the file log4j.xml which is in the following directory:

```
TOMCAT_HOME/webapps/ROOT/WEB-INF/etc/config
```

Log4J defines the logging levels as follows (from the more to the less verbose):

```
ALL < DEBUG < INFO < WARN < ERROR < FATAL < OFF
```
At the bottom of the file, you have the `<root>...</root>` part. Change the `<priority value="info"/>` to `<priority value="debug"/>`, for example, to have more debugging information in the console. You can also change this parameter for some specific part of Jahia like Slide or pdfbox. You can even add your own logger on a specific set of classes, for example:

```xml
<category name="org.quartz">
  <priority value="info"/>
</category>
```

By default logs are outputted to the standard out, which is normally the console. Under Windows, logs will be displayed in the DOS window where Tomcat is running. On Linux, logs will be outputted to the catalina.out file. As Jahia uses Apache log4J for its logging system, you can use tools like Chainsaw (part of the log4J project) to better work with logging messages.

You can change the log-level of Jahia “on-the-fly” without having to shutdown and restart it. This is very useful when you need to have extra logs on a production server, but do not want to restart it just for this. Jahia watches for changes in the log4j.xml file every 60 seconds, so once you have changed the log level, you will need to wait a few seconds before the changes will be effective.

Do not forget to change the values of INFO back, as the DEBUG log level has a pretty important impact on performance.

### 7.4 How to handle module generation timeouts?

As mentioned in chapter 4.2.4 “Object caches”, you may sometimes get exceptions saying, ”Module generation takes too long due to module not generated fast enough (>10000 ms).” This happens when two requests try to get the same module output at the same time. To save resources, Jahia decides to let just one request render the output and the other request wait for it. The maximum wait time is configured in `jahia.properties` with the parameter `moduleGenerationWaitTime`. If rendering the module takes longer than this time, the waiting request gets cancelled with the exception.

The reasons for this exception are various. It could either be an indication that sufficient configured resources are lacking (number of database connections, heap memory, maximum number of file handles, etc.), bottlenecks (slow disk, locks, unnecessary synchronization, etc.), problems with modules (JSPs...
getting compiled, modules opening sockets and waiting for response without timeout, etc.) or bugs/performance issues in the code.

The best way to identify the issue is to analyze thread dumps. Along with the exception, Jahia should have automatically created a thread dump (unless the server load is too high), which already is a good start. If the scenario is reproducible, it would also be good to create multiple thread dumps in short intervals of a few seconds (see Thread dump Management tool mentioned in chapter 6.4.1 “System and Maintenance”, which is able to create multiple thread dumps).

The thread dump may, for instance, show that the JSP compilation is the cause of the problem. In this case you have to ensure that JSPs are getting precompiled after deployment (see JSP Pre-Compilation tool in 6.4.1 “System and Maintenance”) before the server is exposed to public requests (e.g. keep it in the Maintenance Mode). In the error log you should be able to see the URL of the request leading to the timeout, and you should see the cache-key of the module, that is not getting rendered quickly enough. You can also watch out for the other thread, which is rendering the same module and see whether, for instance, it is stuck in some slow or non-responding methods, locks etc.

You should also analyze the error log file from that time to see if there are other exceptions before or after the incident that indicate that the server is running out of resources. In such a case, you may have to utilize or configure more resources for the server.

It could also be an indication that the server is overloaded and not able to serve the number of requests. In such a case, you should think of running Jahia in cluster or add more cluster nodes to handle the expected load.
7.5 How to upgrade Tomcat for an existing Jahia installation?

With Jahia 6.6.2.0 the embedded Tomcat has been upgraded from 6.0.35 to 6.0.37. If you would like to upgrade Tomcat for your existing Jahia installation to this version, you have two options:

- Install a new Jahia 6.6.2.0 and update the Jahia Web application with your existing one
- Install a new Tomcat and use the correct related files from your existing Jahia installation

Install a new Jahia 6.6.2.0 and update the Jahia Web application with your existing one:

1. Overwrite the Jahia Web application directory (under the webapps/ROOT directory in this guide) in your new Jahia 6.6.2.0 installation with the existing Jahia Web application from your previous installation.
2. Check the database connections in \conf\Catalina\localhost\<yourcontext>.xml (e.g.: for ROOT context: \conf\Catalina\localhost\ROOT.xml), and put correct connections as in your previous installation.
3. If you changed some startup or SSL configurations apply the same modifications to this new installation.
4. You can copy Tomcat to the same path as the old one, or adjust the serverHome variable value in the <tomcat>/webapps/ROOT/WEB-INF/etc/config/jahia.properties file to point to your <tomcat> folder path.

Install a new Tomcat and use the correct related files from existing Jahia installation:

Install a new Tomcat and follow the instructions from Chapter 3.7.1 in this documentation, except for these two steps of the section 3.7.1.2:

1. Instead of the ROOT.war you have to use the Jahia Web application from your existing installation.
2. Instead of using the generated shared-libraries.zip archive, copy the following files from your existing Jahia installation in the tomcat/lib directory:
   o ccpp-*.jar
   o jahia-server-utils-*.jar
   o pluto-container-api-*.jar
   o pluto-container-driver-api-*.jar
7.6 How to clean referencesKeeper nodes?

The /referencesKeeper node is used during the import of content/sites and was used also prior to Jahia 6.6.2.3 during the deployment of templates/modules or when doing a copy and paste on content. Whenever there is a reference property in the imported content, where the value cannot be resolved immediately, because e.g. the path or UUID does not exist yet, we create(d) a jnt:reference entry under /referencesKeeper in order to resolve the reference at a later time, when this path or UUID gets available (e.g. after importing other related content). After the path gets available, the reference is correctly set and the node from referencesKeeper gets removed. Jahia can’t know whether these references will be resolvable in future, that’s why we do not delete them. On the other side the problem is that this list can grow and grow.

If the number of referencesKeeper nodes is growing in your environment, you need to look at the nodes and identify from the j:node reference, the j:propertyName and j:originalUuid if the reason is an unresolvable reference found in one of your import files. In that case you need to fix the repository.xml (or live-repository.xml) in the import file and delete the corresponding jnt:reference nodes manually.

Like mentioned since Jahia 6.6.2.3 we have reduced the cases, where we make use of the referencesKeeper node, as we saw that on customer’s sites the number of sub-nodes could grew to hundred thousands, causing performance degradation on import and module deployment. We now also started to log a warning when the number of sub-nodes exceeds 5000. In that case it is necessary to clean the nodes manually.

For that please go to the JCR query tool (see 6.4Tools ), set limit to 10000 and use the SQL-2 request

```sql
SELECT * FROM [jnt:reference]
```

You could also add a where clause if you want to delete just specific nodes, for which you know that they are unresolvable, but most of the time it will be seen that all of them are unresolvable.
After entering the query and the limit activate the checkbox: "Show actions". After fetching the first 10000 results, select the link: "Delete ALL", which will remove all these 10000 entries. You will have to run the query multiple times until you get rid of all entries. You should do that at low-peak times. To run it overnight you could also raise the limit to e.g. 50000 (modify it in the URL: ...&limit=50000&offset=0&displayLimit=100 ) in order to remove 50000 references in one attempt.