Alfresco™ Enterprise Edition

Enterprise Edition 3.3 SP1

Administering an Alfresco ECM Production Environment

Contents

Preface

The purpose of this guide is to provide guidance on installing, configuring, maintaining, and administering an Alfresco production environment.

This guide contains the following chapters:

- Chapter 1 - Installing Alfresco describes how to install Alfresco and components
- Chapter 2 - Configuring Alfresco describes how to configure Alfresco
- Chapter 3 - Administering Alfresco describes how to maintain and manage the system
- Chapter 4 - Troubleshooting describes how to analyze and troubleshoot various scenarios
- Chapter 5 - Reference provides additional information on topics discussed in this guide

Audience

This guide is intended to assist administrators to install, upgrade, configure, and manage an Alfresco production environment.

No specialist knowledge is assumed to install and configure Alfresco; however, the information provided in this guide assumes that you are familiar with the environment on which you are installing. Some administrative tasks also require knowledge of your environment and configuration processes.

Conventions

This guide uses the following terms and conventions.

Typographic conventions used in this guide

The following conventions are used in this guide to indicate types of information.

Convention	Type of information
bold	Identifies user interface elements and items to select, such as menu options, command buttons, and items in a list.
`monospace`	Identifies file and path names, input text, standard output, code, and so on.
italics	Emphasizes importance and used for variable expressions, such as parameters. For example: `kill -9 <`*`process_id`*`>`
CAPITALS	Refers to specific keys on the keyboard. For example: SHIFT, CTRL, or ALT
KEY+KEY	Refers to key combinations when you must press and hold down the first key, and then press another key. For example: CTRL+P or ALT+F4
✏️	Refers to a note that provides supplemental information related to a topic.
❗	Refers to a note that provides important information to remember.
⚠	Refers to a note that warns about the danger of doing or not doing something.
💡	Refers to a note that provides helpful information or a faster way of doing something.

System paths

The following standard conventions describe common system paths:

- Explicit Windows paths use back slashes

`C:\Adirectory`

- Explicit Linux paths use forward slashes

  `/srv/adirectory`

- Back slashes also indicate the same path can apply in both Windows or Linux environments

  `\adirectory\`

`<classpathRoot>` directory (Windows)

The `<classpathRoot>` denotes a directory whose contents are automatically added to the start of your application server's classpath. The location of this directory varies depending on your application server. For example:

- (Tomcat) `C:\Alfresco\tomcat\shared\classes`
- (JBoss) `C:\jboss\server\default\conf`
- (WebLogic) `C:\bea\user_projects\domains\alf_domain`

  The path can be anywhere on the WebLogic classpath.

`<classpathRoot>` directory (Linux)

The `<classpathRoot>` denotes a directory whose contents are automatically added to the start of your application server's classpath. The location of this directory varies depending on your application server. For example:

- (Tomcat) `tomcat/shared/classes/`
- (JBoss) `/jboss/server/default/conf`
- (WebLogic) `<bea>/user_projects/domains/alf_domain`

  The path can be anywhere on the WebLogic classpath.

`alfresco-global.properties` file

The `alfresco-global.properties` file is where you store all the configuration settings for your environment. The file is in Java properties format, so backslashes must be escaped. The file should be placed in `<classpathRoot>`.

`<extension>` directory

The `<extension>` directory is where you store Spring configuration files that extend and override the system configuration. This directory can be found at `<classpathRoot>\alfresco\extension`.

`<web-extension>`

The <web-extension> directory is where you store Spring configurations that extend and override the system Share configuration. This directory can be found at `<classpathRoot>\alfresco\extension`.

`<configRoot>`

The `<configRoot>` directory is where the default configuration files are stored. For example, for Tomcat, `<configRoot>` is `<TOMCAT_HOME>\webapps\alfresco\WEB-INF`.

`<configRootShare>`

The `<configRootShare>` directory is where the default configuration files for Share are stored. For example, for Tomcat, `<configRootShare>` is `<TOMCAT_HOME>\webapps\share\WEB-INF`.

Resources

The resources in the following table provide additional information related to using Alfresco.

The guides referred to in this table are available in PDF format from Alfresco Network.

Resource	Description
Getting Started with Alfresco Share Collaboration	Introduction to using Alfresco Share for collaborative content management.
Getting Started with Explorer DM for Alfresco	Introduction to using Alfresco DM Explorer features.
Getting Started with WCM for Alfresco	Introduction to using WCM features.
Getting Started with Alfresco Records Management	Introduction to using the Records Management module.
Share End User Help	How to use the Alfresco Share user interface.
Explorer End User Help	How to use the Alfresco Explorer user interface.
MS Office Add-in End User Help	How to use the MS Office Add-in.
Managing Alfresco Content from within Microsoft Office	How to use the Alfresco Explorer user interface.
Installing and Configuring Alfresco ECM Enterprise Edition 3.3 SP1	Installing Alfresco and related components, and configuring the core and extended services.
Administering an Alfresco Enterprise Edition 3.3 SP1 Production Environment	Administration guide for installing, configuring, and managing an Alfresco production environment.
http://network.alfresco.com	Alfresco Enterprise Edition Network provides real-time access to Alfresco Support, developers, technicians, and staff. It works in conjunction with Alfresco Enterprise Edition backbone services to automate real-time, seamless integration with Alfresco Enterprise Edition server installations, and it provides reporting for Alfresco subscription customers.
Knowledge Base	Additional information on specific Enterprise Edition features and applications in white papers, Frequently Asked Questions (FAQ), and articles.
http://wiki.alfresco.com	Alfresco Enterprise Edition wiki, community-contributed information on all aspects of the Alfresco Enterprise Edition environment.
http://www.alfresco.com	Alfresco web site for all information about Alfresco, including links to various resources, such as webinars and forums.
http://www.alfresco.com/services/support/faq	Lists the most requested information from support and how to contact Alfresco Support.

Before installing

This chapter describes how to prepare your system for installing Alfresco.

Software requirements

The following table lists the required software that must be on your system before you install Alfresco.

Component	Recommendation
Java SE Development Kit (JDK)	The Sun Microsystems JDK 6 is required.
Database	Alfresco comes preconfigured with the MySQL database. If you intend to use Alfresco in a production environment, you can use the default MySQL database or one of the supported database. For the latest information on supported databases, refer to the Alfresco website.
OpenOffice.org	Alfresco uses OpenOffice for transforming documents from one format to another, for example, a text file to a PDF file. If you do not install OpenOffice, you will not have access to the transformation functionality.
Flash Player Version 10.x	Alfresco Share requires Flash Player Version 10.x to upload multiple files and view Flash previews. If you do not install Flash, you see the upload screen for single files.
SWF Tools	Alfresco Share uses the pdf2swf utility for previewing PDF files. If you do not install SWF Tools, you will not see PDF previews, but image previews will still be available.

Supported stacks

The supported stacks are the combinations of operating systems, databases, and application servers that are tested and certified for Alfresco. For the latest list, please refer to the **Supported Stacks** page on Alfresco Network at http://network.alfresco.com.

Installing a JDK

A Java SE Development Kit (JDK) must be installed on your system before you install Alfresco. Some Alfresco installation wizards will detect whether you have a JDK on your machine and, if not, install a version for you. This task explains how to install JDK manually.

1. Browse to the Sun Microsystems Java download website: http://java.sun.com
2. Select and download the Java Development Kit (JDK) 6 for your platform.
3. If prompted, specify a location in which to download.
4. Navigate to where you downloaded the JDK.
5. Install the JDK on your system.

JDK is installed on your system. Next, set the `JAVA_HOME` environment variable.

Verifying the `JAVA_HOME` environment variable location

The `JAVA_HOME` environment variable location must be set to where the JDK is installed.

1. Open a command prompt.

2. Enter the following:
   - (Windows) `echo %JAVA_HOME%`
   - (Linux) `echo $JAVA_HOME`

To add or update the variable location in Windows, see Adding folder paths to the Windows path variable on page 208

Installing MySQL

This section describes how to set up a MySQL open source relational database management system (RDBMS) for use with Alfresco.

Some of the Alfresco installation wizards install an embedded instance of MySQL that is configured with the correct settings. If you prefer to install MySQL database independently, this section describes the configuration settings that you should use.

Installing MySQL

This task describes how to install a MySQL database for use with Alfresco.

1. Browse to the MySQL download site: http://dev.mysql.com/downloads
2. Locate and select the appropriate package for your platform.

   Alfresco requires MySQL 5.1.39.
3. If prompted, specify a location on your system in which to download and install MySQL.
4. Browse to where you downloaded MySQL, and double-click the installer file.

   The MySQL Server Setup wizard guides you through the MySQL installation, followed by the Configuration wizard.

   When you are prompted, add the MySQL program files to the your system path.
5. At the Welcome window, click **Next**.
6. Select the **Typical** setup type, and click **Next**.
7. Click **Install**, and click **Next**.
8. Skip the MySQL registration.
9. In the Wizard Completed window, click **Finish**.

   The MySQL Server Setup wizard closes, and the MySQL Server Instance Configuration wizard opens.

Configuring MySQL

The MySQL configuration wizard starts immediately after the MySQL Server Setup wizard closes. This section describes how to configure MySQL to work with Alfresco.

1. In the Welcome window, click **Next**.
2. Select **Detailed Configuration**, and click **Next**.
3. Select **Server Machine**, and click **Next**.

   For production use, choose **Dedicated MySQL Server Machine**. The option selected determines the memory allocation.
4. For database use, select **Transactional Database Only**, and click **Next**.

   This creates a database that uses InnoDB as its storage engine.

5. Accept the default drive and path for the **InnoDB tablespace** settings, and click **Next**.
6. To set the approximate number of concurrent connections to the server, select **Decision Support (DSS) OLAP,** and click **Next**.
7. Accept the default networking options (**Enable TCP/IP Networking**, **Port Number 3306**), and the default server SQL mode (**Enable Strict Mode**), and click **Next**.
8. Select **Best Support for Multilingualism**, and click **Next**.

   This sets the default character set to be UTF-8.
9. (Windows) Select **Install as Windows Service** and **Include Bin Directory in Windows PATH**, and click **Next**.
10. Set the following security options:
    a. Select **Modify Security Settings**.
    b. Type the root password `admin`, then retype the password.
11. Click **Next**.
12. Click **Execute**.

    A message informs you the configuration is complete and MySQL is installed.
13. Click **Finish**.

MySQL is set up. Next, you can verify that MySQL is correctly installed.

Verifying the MySQL installation

Once you have installed MySQL, this task describes how to verify that it was installed correctly.

1. Open a command prompt.
2. At the prompt, enter `mysql -u root -p`.
3. Type the password that you set during the installation, and press ENTER.

   Information about the installed MySQL version displays. If no errors are reported, MySQL is installed and running.
4. At the `mysql>` prompt, type `exit` to exit MySQL.

You have verified the MySQL installation was successful.

Installing Alfresco

This chapter provides information for installing Alfresco and Alfresco components. Depending on your system, you can install Alfresco using a number of different methods. For example, you can install Alfresco using one of the following methods:

- Using an installation wizard, which contains the required software and components you need
- Using a bundle that includes a preconfigured Tomcat server, the Alfresco Web Archive (WAR), batch files, database setup scripts, and a sample extensions folder
- Using a standard WAR file to deploy on your existing application server

Installation options

There are a number of different installation files available to you, each of which you can choose depending on what is already installed on your system. By choosing an installation file that contains only the necessary components, this lets you reduce your download time.

The following table is a guide to help you determine what file to download and install.

Description	When to use	File name
Alfresco Tomcat bundle: A preconfigured Tomcat bundle	For manual installation of Alfresco bundled in a Tomcat application server. Use this bundle if you have some components, for example, JDK and OpenOffice, installed on your system.	`alfresco-enterprise-tomcat-3.3.1.zip` `alfresco-enterprise-tomcat-3.3.1.tar.gz`
Alfresco WAR: Includes Alfresco WAR files for deployment in existing application servers	For manual installation or for upgrading. Use this file if you have all the prerequisite software and an existing application server installed on your system.	`alfresco-enterprise-war-3.3.1.tar.gz` `alfresco-enterprise-war-3.3.1.tar.gz`
Alfresco WCM: Includes WCM functionality for adding to an Alfresco install	If you have Alfresco installed on your system from either the Tomcat bundle or the Alfresco WAR.	`alfresco-enterprise-wcm-3.3.1.zip` `alfresco-enterprise-wcm-3.3.1.tar.gz`
Deployment Receiver: Includes installer for WCM deployment.	If you have Alfresco and WCM installed on your system.	`Alfresco-DeploymentEnterprise-3.3.1-Setup.exe` `Alfresco-DeploymentEnterprise-3.3.1-Linux-x86-Install`

Downloading Enterprise installation files

1. Browse to Alfresco Enterprise Network.
2. At the Login prompt, enter your user name and password.
3. Click **Downloads**.

4. Select the version from the left panel or search for the required file using the **Advanced Filters**.
5. Select the installation file.

   A summary of the file displays.
6. Click **Download Now** to download the file to your system.

   Refer to the relevant section in this guide for installation instructions.

Installing Alfresco on Windows

This section describes how to install Alfresco on Windows using the following methods:

- Tomcat bundle installation
- Installation as a Windows service

Installing Alfresco Tomcat bundle on Windows

This section describes how to install Alfresco using the Tomcat bundle on a Windows platform.

Before you start, ensure that you have a JDK installed. Refer to Installing a JDK on page 13. Alfresco also requires Flash Player, SWF Tools, and OpenOffice.org. For more information on installing these components, refer to Installing Alfresco components on page 29.

1. Browse to the Enterprise download area, and download the following installation file:

   `alfresco-enterprise-tomcat-3.3.1.zip`
2. Specify `C:\Alfresco` as the location for the download.
3. Extract the downloaded file into the location you specified.
4. Ensure that the `JAVA_HOME` environment variable points to the location of your JDK install.
5. Open the `alfresco-global.properties` file.
6. Locate the property `dir.root`.
7. Change the property to show an absolute path for the `alf_data` directory. Replace backslashes with slashes. For example:

   `dir.root=C:/Alfresco/alf_data`

   This directory is created for you when you start the server.
8. Add the property settings for your preferred database.
9. Comment out the settings for the remaining databases.
10. Save the `alfresco-global.properties` file.

For further details on databases, refer to Configuring databases on page 67.

Configuring Alfresco as a Windows service

This section describes how to configure Alfresco as a Windows service in a standard Tomcat installation.

The Alfresco default installation is bundled as a web application that launches within the Tomcat application server. To configure Alfresco to run as a Windows service, you need to set up Tomcat to run as a Windows service.

Before you start, Alfresco and JDK 5 or 6 must be installed on your system.

1. Open a command prompt.

2. To install Alfresco as a Windows service, enter the following commands:

   For Tomcat 6:

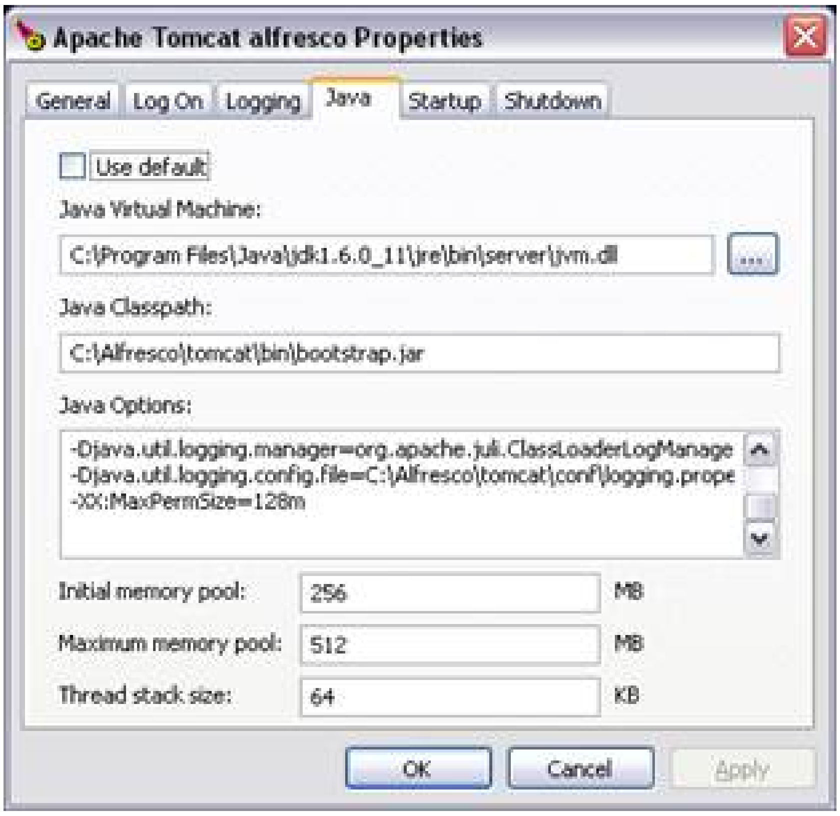
```
cd c:\alfresco\tomcat\bin
service.bat install alfresco
tomcat6 //IS//Tomcat6 --DisplayName="Alfresco Server" \
--Install="C:\Program Files\Tomcat\bin\tomcat6.exe" --Jvm=auto \
--StartMode=jvm --StopMode=jvm \
--StartClass=org.apache.catalina.startup.Bootstrap --StartParams=start \
--StopClass=org.apache.catalina.startup.Bootstrap --StopParams=stop
```

   It is important to use the full path. The commands in this task assume an Alfresco installation at `c:\alfresco`.

3. To edit your service settings, enter the following commands:

   For Tomcat 6:

```
cd c:\alfresco\tomcat\bin
tomcat6w.exe //ES//alfresco
```



Use default: ☐

Java Virtual Machine: C:\Program Files\Java\jdk1.6.0_11\jre\bin\server\jvm.dll

Java Classpath: C:\Alfresco\tomcat\bin\bootstrap.jar

Java Options:

```
-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManage
-Djava.util.logging.config.file=C:\Alfresco\tomcat\conf\logging.prope
-XX:MaxPermSize=128m
```

Initial memory pool: 256 MB

Maximum memory pool: 512 MB

Thread stack size: 64 KB

4. Locate the service named **Alfresco Server** in the Services panel.
5. Start the **Alfresco Server** service.

You can uninstall the service using the following commands:

```
cd c:\alfresco\tomcat\bin
service.bat uninstall alfresco
```

Installing the Alfresco Tomcat bundle on Linux

This section describes how to install the Alfresco Tomcat bundle on Linux.

Before you start, ensure that you have a JDK installed. Refer to Installing a JDK on page 13. Alfresco also requires Flash Player, SWF Tools, and OpenOffice.org. For more information on installing these components, refer to Installing Alfresco components on page 29.

1. Browse to the Alfresco Enterprise Edition downloads area, and download the following installation file:

   `alfresco-enterprise-tomcat-3.3.1.tar.gz`
2. Specify `/opt/alfresco/` as the location for the download.

   If you change this location from`/opt/alfresco/`, edit `alfresco.sh` to point the `APPSERVER` variable to the Tomcat bundle location `/opt/alfresco/tomcat`.
3. Extract the downloaded file into the location you previously specified.
4. Ensure that the `JAVA_HOME` environment variable points to the location of your JDK install.
5. Open the `alfresco-global.properties` file.
6. Locate the property `dir.root`.
7. Change the property to show an absolute path for the `alf_data` directory. For example:

   `dir.root=/opt/alfresco/alf_data`

   This directory is created for you when you start the server.
8. Add the property settings for your preferred database.
9. Comment out the settings for the remaining databases.
10. Save the `alfresco-global.properties` file.
11. (Optional) If you deployed previous versions of Alfresco, you can remove any temporary files created by your application server.

For further details on databases, refer to Configuring databases on page 67.

Installing the Alfresco WAR on any platform

Use the Web Archive (WAR) file to install Alfresco on any platform into an existing application server. A WAR file is a JAR file used to distribute a collection of files (JavaServer Pages, servlets, Java classes, XML files, tag libraries, and static Web pages) that together constitute a web application.

Use this method of installing Alfresco if you already have installed a JDK, a supported database, an application server, and the additional Alfresco components.

1. Browse to the Alfresco Enterprise Edition download area.
2. Select and download one of the following files:
   - (Windows) `alfresco-enterprise-war-3.3.1.zip`
   - (Linux) `alfresco-enterprise-war-3.3.1.tar.gz`
3. Specify a location for the download.
4. Extract the downloaded file.
5. Copy the `alfresco.war` file and `share.war` file to the appropriate location for your application server, for example: `<TOMCAT_HOME>/webapps`.
6. Ensure that you download the sample extension files. Refer to Download the extension samples on page 20.
7. Copy the `alfresco-global.properties` file to `<classpathRoot>`.
8. Edit the `alfresco-global.properties` file with your configuration settings. Refer to .

If you deployed previous versions of Alfresco, you must remove any temporary files created by your application server.

Download the extension samples

Each Alfresco distribution includes a download containing sample extension files, such as the global properties file (alfresco-global.properties). The sample extension files also include the Spring configuration extensions, which can be used for advanced Alfresco customizations.

1. Browse to the Alfresco Enterprise Edition downloads area.
2. Select and download one of the following files:
   - (Windows) `alfresco-enterprise-sample-extensions-3.3.1.zip`
   - (Linux) `alfresco-enterprise-sample-extensions-3.3.1.tar.gz`
3. Specify a location for the download.

Once you have downloaded the sample files, copy them into `<classpathRoot>`.

Modifying the directory paths for Tomcat 6.x

If you install Tomcat 6.x separately, some of the directories that were present in Tomcat 5.x are not present. For example, Tomcat 6.x does not contain the `shared/classes` and `shared/lib` directories. Alfresco uses these directories to locate some of the configuration override files.

This section describes how to configure Tomcat 6.x to use the correct directory structure and files for Alfresco.

1. Locate `<TOMCAT_HOME>`.
2. Create the `shared/classes` directory.
3. Open the `<TOMCAT-HOME>/conf/catalina.properties` file.
4. Change the value `shared.loader=` to the following:

   ```
   shared.loader=${catalina.base}/shared/classes,${catalina.base}/shared/
   lib/*.jar
   ```
5. Copy the JDBC drivers for the database you are using to:

   `lib/`
6. Ensure that a copy of the `commons-el.jar` file is in the `lib` directory.
7. If you are using Java SE 6, copy any jar files that needed to go into the Tomcat `common/endorsed` directory into the following directory:

   `...jdk6/jre/lib/endorsed`

Deploying Share into a separate Tomcat instance

This task provides information for running Share in a separate Tomcat instance. These instructions are for Windows deployments, but Linux-based deployments can use the same methods.

1. Install a new Tomcat instance.
2. Modify the `/conf/server.xml` file for the new Tomcat instance as follows:
   a. Change the port number in the line (for example, to 8006):

   ```
   <Server port="8005" shutdown="SHUTDOWN">
   ```
   b. Change the port number in the section (for example, to 8180):

   ```
   <!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
   <Connector port="8080" ....
   ```
3. Move the `share.war` file from the original Tomcat `\webapps` directory to the new Tomcat `/webapps` directory.

4. (Optional) Configure the original Alfresco Tomcat deployment.
5. Start the original Tomcat. You can use Alfresco supplied batch files.
6. Ensure that a copy of the `commons-el.jar` file is in the Share Tomcat `lib` directory.
7. If you are running the Share Tomcat on a separate machine, you must modify the override file in the Share Tomcat `web-extension` directory, as follows:
   a. Open the `webscript-framework-config-custom.xml` file.
   b. Change any instance of the server and port to the correct name or IP address of the Alfresco server.

      `http://`*`yourserver`*`:8080`
8. Start the new Share Tomcat. You can use copies of the Alfresco supplied batch files, or your own methods.

Installing Alfresco on JBoss

You can install and deploy the Alfresco WAR on the JBoss application server. This deployment uses the supported versions of Red Hat Enterprise Linux (RHEL), JBoss, and the PostgreSQL database.

Before you install Alfresco on JBoss, ensure that you have added the following channel subscriptions relevant to your architecture in Red Hat Network (RHN):

- RHEL Supplementary
- Red Hat Application Stack v2

1. Run the following command:

   `yum install java-1.6.0-sun-devel`
2. Repeat the above command for the following packages:
   a. `yum install postgresql-server`
   b. `yum install postgresql-jdbc`
   c. `yum install jbossas`
   d. `yum install ImageMagick`
   e. `yum install Xvfb`
   f. `yum install giflib`
   g. `yum install xorg-x11-fonts-misc`
   h. `yum install wget`
   i. `yum install jakarta-commons-el`
   j. `yum install httpd`
   k. `yum install mod_jk-ap20`
3. Run the following command to install the SWF Tools:

   ```
   rpm -ivh http://www.fonteaperta.com/scripts/alfresco3/
   swftools-0.8.5-1.i386.rpm
   ```
4. Use the following commands to download and install the Linux RPM version of OpenOffice 3.0. These instructions are for the i386 platform.

   ```
   cd /tmp
   wget http://openoffice.mirror.wrpn.net/stable/3.0.0/
   OOo_3.0.0_LinuxIntel_install_wJRE_en-US.tar.gz
   tar xvfz OOo_3.0.0_LinuxIntel_install_wJRE_en-US.tar.gz
   cd OOO300_m9_native_packed-1_en-US.9358/RPMS
   ```

```
rm jre-6u7-linux-i586.rpm
rm ooobasis3.0-gnome-integration-3.0.0-9358.i586.rpm
rpm -ivh *.rpm
Instal
```

5. Browse to the Enterprise Edition download area.
6. Download the following file to your local machine:

   `alfresco-enterprise-war-3.3.1.tar.gz`
7. Use the following commands to deploy Alfresco to JBoss:

```
mkdir /usr/share/alfresco
cp /root/alfresco-enterprise-war-3.1.tar.gz /usr/share/alfresco
cd /usr/share/alfresco
tar xvfz alfresco-enterprise-war-3.1.tar.gz
rm -rf alf_data
rm alfresco-enterprise-war-3.1.tar.gz
mv alfresco.war alfresco.jar
mv share.war share.jar
mkdir alfresco.war
mkdir share.war
mv alfresco.jar alfresco.war
mv share.jar share.war
cd alfresco.war
jar xvf alfresco.jar
rm alfresco.jar
cd ../share.war
jar xvf share.jar
rm share.jar
mkdir -p /var/lib/alfresco/alf_data
chown -R jboss:jboss /var/lib/alfresco
mkdir -p /var/logs/alfresco
chown -R jboss:jboss /var/logs/alfresco
chown -R jboss:jboss /usr/share/alfresco
```

Configuring JBoss for Alfresco

This section describes the configuration steps to deploy Alfresco to JBoss.

1. Create the symbolic links for `alfresco.war` and `share.war`:

```
cd /usr/share/jbossas/server/default/deploy
ln -s /usr/share/alfresco/alfresco.war
ln -s /usr/share/alfresco/share.war
```

2. Create the symbolic link for the extension folder:

```
cd /usr/share/jbossas/server/default/conf
mkdir alfresco
cd alfresco
ln -s /usr/share/alfresco/extensions/extension
```

3. Create the symbolic link for the JDBC connector. For example, for PostgreSQL, use the following commands:

```
cd /usr/share/jbossas/server/default/lib
ln -s /usr/share/java/postgresql-8.3-604.jdbc4.jar
```

4. Create the symbolic link for for jakarta commons el library:

   `ln -s /usr/share/java/jakarta-commons-el.jar`
5. Edit the `usr/share/jbossas/server/default/run.conf` file.
6. Modify the `JAVA_OPTS` to read:

```
JAVA_OPTS="-Xms256m -Xmx1024m -XX:PermSize=128m
-Dorg.apache.catalina.STRICT_SERVLET_COMPLIANCE=false
-Dsun.rmi.dgc.client.gcInterval=3600000
-Dsun.rmi.dgc.server.gcInterval=3600000"
```

7. Edit the `/usr/share/jbossas/server/default/deploy/jboss-web.deployer/conf/web.xml` file.
8. Add two new `<init-param>` entries to the `<servlet-name>jsp</servlet-name>`.

   ```
   <init-param>
   <param-name>compilerSourceVM</param-name>
   <param-value>1.5</param-value>
   </init-param>
   <init-param>
   <param-name>compilerTargetVM</param-name>
   <param-value>1.5</param-value>
   </init-param>
   ```
9. Edit the `/usr/share/jbossas/server/default/deploy/ejb3.deployer/META-INF/persistence.properties` file.
10. Change the `hibernate.bytecode.provider` to be `cglib` instead of `javassist`:

    ```
    hibernate.bytecode.provider=cglib
    ```
11. Ensure that JBoss starts by default using the following command:

    ```
    chkconfig jbossas on
    ```

Ensure that you have created the Alfresco database for your supported database and modified the database configuration files.

Setting the URIEncoding

When running Alfresco with JBoss, you must add the `URIEncoding="UTF-8"` setting to the `server.xml` file. This sets the default character set to be UTF-8.

1. Ensure that the Alfresco server is not running.
2. Edit the `server.xml` file.

   The `server.xml` file is located in `<jboss>/server/default/deploy/jboss-web.deployer` and `<jboss>/server/all/deploy/jboss-web.deployer`.
3. Locate the following section:

   ```
   <Connector port="8080" address="${jboss.bind.address}"
            maxThreads="250" maxHttpHeaderSize="8192"
   ```
4. Add `URIEncoding="UTF-8"`.

   For example:

   ```
    <Connector port="8080" URIEncoding="UTF-8"
    address="${jboss.bind.address}"
            maxThreads="250" maxHttpHeaderSize="8192"
   ```
5. Restart the Alfresco server.

Installing Alfresco on WebLogic

This section describes how to install Alfresco as an Enterprise ARchive format (EAR) into Oracle WebLogic 10.3.

Certain components of Alfresco require access to the EAR file contents as files. These instructions require expanding the `.ear` into exploded format, as described in the WebLogic documentation. On other application servers, this may not be necessary.

Before you start:

- Install OpenOffice and set the `ooo.exe` property in the the `alfresco-global.properties` file

- Install MySQL and create an `alfresco` database and user with appropriate permissions
- Install WebLogic 10.3 without creating any domains or servers

The Alfresco WebLogic deployment solution uses a Filter Classloader. The Classloader is configured in the `weblogic-application.xml` file, which ensures that the unmodified contents of the Alfresco web module will run in WebLogic.

1. Browse to the Enterprise Edition download area.
2. Download the `alfresco-enterprise-3.3.1.ear` file.
3. Download the license file.
4. Create a directory in the WebLogic user's home directory to host the exploded EAR file and copy the `alfresco-enterprise-3.3.1.ear` file to that directory.
5. Run the following commands in the new directory to explode the EAR file:

   ```
   mkdir alfresco
   cd alfresco
   jar xvf ../alfresco-enterprise-3.3.1.ear
   mv alfresco.war alfresco.war.tmp
   mv share.war share.war.tmp
   mkdir alfresco.war
   mkdir share.war
   cd alfresco.war
   jar xvf ../alfresco.war.tmp
   cd ../share.war
   jar xvf ../share.war.tmp
   ```
6. Open the WebLogic Configuration Wizard. For example, on Unix, use the following command:

   `<Weblogic_HOME>/common/bin/config.sh`
7. Create a new domain.

   For example, `alf_domain`.
8. Create a directory for the license file.

   For example, in Linux: `mkdir -p <Weblogic_HOME>/user_projects/domains/alf_domain/alfresco/extension/license`
9. Move the license `.lic` file into the `license` directory.
10. In the `<Weblogic_HOME>/user_projects/domains/alf_domain` directory, create the `alfresco-global.properties` file.

    Modify the file in the same way you would for repository configuration. For more information on creating this file, refer to Download the extension samples on page 20.
11. In the `alfresco-global.properties` file, add the following line:

    `db.pool.statements.enable=false`

    This is required to make the DBCP connection pool work on WebLogic.
12. Move the database JDBC driver `.jar` to the `<Weblogic_HOME>/user_projects/domains/alf_domain/lib` directory.
13. Edit the `MEM_MAX_PERM_SIZE` line to increase the PermGen space:

    - (Linux) In the `<Weblogic Home>/user_projects/domains/alfresco_domain/bin/setDomainEnv.sh` file, set the following property:

      ```
      MEM_MAX_PERM_SIZE="-XX:MaxPermSize=256m"
      ```
    - (Windows) In `<Weblogic Home>\user_projects\domains\alfresco_domain\bin\setDomainEnv.bat` file, set the following property:

      ```
      MEM_MAX_PERM_SIZE=-XX:MaxPermSize=256m
      ```

This setting may need to be increased further, depending on the number of deployed applications.

14. To enable WebDAV, feed service, proxy service, and other services that need Basic HTTP authentication:
    a. Edit the `config.xml` file in the `alf_domain` directory.
    b. Navigate to the `<Weblogic Home>/user_projects/domains/alf_domain/config` folder.
    c. In the `config.xml` file, add the following tag before the end of the `</security-configuration>` section:

       ```
       <enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-credentials>
       ```
15. Start the domain admin server. For example:

    ```
    <Weblogic_HOME>/user_projects/domains/alf_domain/startWebLogic.sh
    ```
16. Open a web browser and log in to the admin server (for example, at `http://localhost:7001/console`). The default user name and password is `weblogic`.
17. To enable automatic control of your Alfresco server:
    a. Create a machine with the details of the machine running the domain. This will allow the node manager to control servers on that machine.
    b. Create a server called `AlfrescoServer`, within the new machine.

       Note that you have to choose a unique port number. A good port number to choose is 8080 because it is preconfigured in Share. You can leave the host name blank if you want it to listen on all network adapters.
    c. Ensure that the node manager is running (for example, `<Weblogic_HOME>/wlserver_10.3/server/bin/startNodeManager.sh`).

       You will be able to use the admin server **Change Center** panel to start and stop the Alfresco server.

    Refer to the WebLogic documentation to find out how to create a machine and new server within the machine.
18. In the left pane of the Administration Console, click **Deployments**.
19. In the right pane, click **Install**.
20. Using the **Install Application Assistant**, locate the directory of your exploded EAR file (containing the `alfresco.war` and `share.war` directories).
21. Locate the file or directory to install, and then click **Next**.
22. Check **Install this deployment as an application** radio button, and then click **Next**.
23. Click **Finish**.
24. Click **Activate Changes**.
25. Using the **Change Center** panel, restart `AlfrescoServer`.
26. Log in to Alfresco:
    - Alfresco Share at `http://localhost:8080/share`
    - Alfresco Explorer at `http://localhost:8080/alfresco`

If the Alfresco finds a JDBC data source with JNDI path (`java:comp/env/jdbc/dataSource`), it will use that rather than the embedded data source. To set that up in WebLogic you need to define a new global data source, for example, `AlfrescoDataSource`. See the WebLogic documentation for more information. Then, map `AlfrescoDataSource` in to Alfresco by adding the `WEB-INF/weblogic.xml` file into `alfresco.war` containing the following:

```
<! DOCTYPE weblogic-web-app PUBLIC "-//BEA Systems, Inc.//DTD Web
Application 8.1//EN"

"http://www.bea.com/servers/wls810/dtd/weblogic810-web-jar.dtd" >
< weblogic-web-app >
< reference-descriptor >
< resource-description >
< res-ref-name > jdbc /dataSource </ res-ref-name >
< jndi-name > AlfrescoDataSource </ jndi-name >
</ resource-description >
</ reference-descriptor >
</ weblogic-web-app >
```

Installing Alfresco on WebSphere

This section describes how to install Alfresco on WebSphere 7.0. These instructions are valid for installing on Windows 2008.

Before you start:

- Install OpenOffice and set the `ooo.exe` property in the the `alfresco-global.properties` file
- Install MySQL and create an `alfresco` database and user with appropriate permissions
- Install WebSphere 7.0 and ensure that it is updates with all available service packs, including WASSDK (embedded JDK) patches. This is because the Alfresco optimized I/O code requires the very latest patches available from IBM and will not function on an unpatched Websphere 7 installation

1. Add the database driver `jar` file to support your database connection to `/lib` within the WebSphere installation directory.
2. Create the `/SharedLib` directory within the Alfresco root directory.
3. Configure the Java Server Faces implementation.
   a. Copy the following files from the Alfresco distribution to the `/SharedLib` directory:
   ```
   myfaces-api-1.1.5.jar
   myfaces-impl-1.1.5.jar
   commons-digester-1.6.jar
   commons-beanutils-1.7.0.jar
   commons-codec-1.3.jar
   commons-collections-3.1.jar
   commons-el.jar
   commons-fileupload-1.1.1.jar
   commons-lang-2.1.jar
   commons-logging-1.1.jar
   ```
   b. Delete the following files from the `alfresco.war` file:
   ```
   api-1.1.5.jar
   myfaces-impl-1.1.5.jar
   ```
   c. Open the WebSphere console navigation tree.
   d. Click **Environment > Shared libraries**.
   e. Select **Server scope**.
   f. Click **New**.
   g. Enter a name.

      For example, `alfJsf`.

h. Enter the following in the **Classpath** field:

```
${USER_INSTALL_ROOT}/sharedLib/myfaces-api-1.1.5.jar
${USER_INSTALL_ROOT}/sharedLib/myfaces-impl-1.1.5.jar
${USER_INSTALL_ROOT}/sharedLib/commons-beanutils-1.7.0.jar
${USER_INSTALL_ROOT}/sharedLib/commons-codec-1.3.jar
${USER_INSTALL_ROOT}/sharedLib/commons-collections-3.1.jar
${USER_INSTALL_ROOT}/sharedLib/commons-digester-1.6.jar
${USER_INSTALL_ROOT}/sharedLib/commons-el.jar
${USER_INSTALL_ROOT}/sharedLib/commons-fileupload-1.1.1.jar
${USER_INSTALL_ROOT}/sharedLib/commons-lang-2.1.jar
${USER_INSTALL_ROOT}/sharedLib/commons-logging-1.1.jar
```

i. Select **Use an isolated class loader for this shared library**.

j. Click **OK**.

k. Click **Save directly to the master configuration**.

4. Upgrade the `xstream-1.1.3.jar` file to `xstream-1.2.2.jar` in the `alfresco.war` file.
5. Create the `WEB-INF\classes\alfresco\extension\license` directory in the `alfresco.war`.
6. In the `<configRoot>\classes\` directory, create the `alfresco-global.properties` file.

   Modify the file in the same way you would for repository configuration. For more information on creating this file, refer to Download the extension samples on page 20.
7. Deploy the `alfresco.war` file using the WebSphere Administrative console.

   a. Select **Application > Application Types > WebSphere enterprise applications space**.

   b. Click **Install**.

   c. In Step 1 of the wizard, enter the name of an installation folder.

   For example, `alfresco`.

   d. Click **Next**.

   e. In Step 2 and Step 3, accept all the settings, and click **Next**.

   f. In Step 4 **Map context roots for Web Modules**, enter the context of the Alfresco module as `/alfresco`.

   g. Click **Finish**.

   Wait until the application is deployed.

   h. Save the configuration.

   Do not start the application yet.
8. Configure an application class loader.

   a. Select **Application > Application Types > WebSphere enterprise applications > alfresco_war > Class loading and update detection**. (application configuration page)

   b. Select the following options: **Classes loaded with local class loader first (parent last)** and **Class loader for each WAR file in application**.
9. Configure the application shared libraries to use the `alfJsf` libraries.

   a. Select **Application > Application Types > WebSphere enterprise applications > alfresco_war > Shared library references** (Application Configuration page).

   b. Select **Alfresco Web Client** module and click **Reference shared libraries**.

   c. Navigate to the `alfJsf` libraries from the **Available** to **Selected** box.

   d. Click **OK**.

10. Click **Save configuration**.
11. Prepare `share.war`.
    a. Create an empty file.

       For example, `_fake.txt`.
    b. Move the file to `\WEB-INF\classes\alfresco\site-data\page-associations` and `\WEB-INF\classes\alfresco\site-data\content-associations` to `share.war`.
    c. Configure all URLs in the Share configuration files to your required values.

       For example, `http://localhost:9080/alfresco`.
12. Use WebSphere Administrative console to deploy the `share.war`.
    a. Select **Application > Application Types > WebSphere enterprise applications space**.
    b. Click **Install**.
    c. In Step 1 of the wizard, enter the name of an installation folder.

       For example, `share`.
    d. Click **Next**.
    e. In Step 2 and Step 3, accept all the settings, and click **Next**.
    f. In Step 4 **Map context roots for Web Modules**, enter the context of the Alfresco module as `/share`.
    g. Click **Finish**.

       Wait until the application is deployed.
    h. Save the configuration.
13. Start `alfresco_war`.
14. Start `share_war`.

If configured correctly, the `alfresco.log` file will contain a warning that both the MyFaces and RI implementations of JSF have been found on the classpath.

Installing a new license

The Alfresco license file must be installed before you can use Alfresco.

You must have installed Alfresco before you can install the license. This is because you use a license directory within the installed product.

1. Browse to Alfresco Enterprise Network.
2. Log in using your Network user name and password.
3. Click **Downloads**.
4. Search for and download the Enterprise license file.

   The license file has a file extension of `.lic`.
5. Ensure that the Alfresco server is not running.
6. From your Alfresco installation directory, browse to the `<extension>` directory, for example for Tomcat on Windows, this is:

   `C:\Alfresco\tomcat\shared\classes\alfresco\extension`

   Refer to System paths on page 10 for information on the default paths for the `<extension>` directory.
7. Create the `license` directory.

8. Move the `.lic` file into the `license` directory.

You have installed the Alfresco license file.

When you run Alfresco, the server detects the existence of the `.lic` file and installs your license. If the license is valid, Alfresco renames the file to `<license-name>.lic.INSTALLED` and you can begin to use the terms of your license immediately.

Installing Alfresco components

This section describes how to install components that integrate with Alfresco. Some of these components can be installed any time before or after installing Alfresco.

Installing Alfresco WCM

This section describes how to set up Alfresco Web Content Management (WCM).

Installing WCM to an existing instance of Alfresco

This task describes how to install WCM to an existing instance of Alfresco.

1. Browse to the Alfresco Enterprise Edition download area.
2. Select one of the following files:
   - (Windows) `alfresco-enterprise-wcm-3.3.1.zip`
   - (Linux) `alfresco-enterprise-wcm-3.3.1.tar.gz`
3. Download the file into the Alfresco home directory. For example:
   - (Windows) `C:\Alfresco`
   - (Linux) `/opt/alfresco`
4. Browse to the Alfresco home directory, and unzip the downloaded file.

   If your unzip program asks about existing directories, allow this because no existing files will be overwritten.
5. In the root of the Alfresco home directory, copy the file `wcm-bootstrap-context.xml` to the `<extension>` directory.
6. Restart the Alfresco server.

   This ensures that the Alfresco server starts to use the installed WCM components. To restart the Alfresco server, see Starting the Alfresco server on page 53

WCM is installed and configured.

Verifying the WCM installation

Verify the WCM installation after you have installed, configured, and started the Alfresco server.

1. In the Alfresco home directory, open `alfresco.log`.
2. In `alfresco.log`, search for the following text:

   ```
   The Web Forms folder was successfully created: and The Web Projects
   folder was successfully created:
   ```
3. Check that the following additional spaces are in your Alfresco repository:
   - **Web Projects** in **Company Home**
   - **Web Forms** in **Data Dictionary**

WCM has been installed, configured, started, and verified. To use the Website Preview feature, start the Alfresco virtualization server (see Starting the Alfresco virtualization server on page 54).

Installing the WCM standalone deployment receiver

The standalone deployment receiver allows a web project from WCM to be deployed to a remote file server, typically a web or application server.

The published files are then typically published by a web server such as Apache for static content, or an application server, such as Tomcat or JBoss for dynamic content.

1. Browse to the Alfresco Enterprise Edition download area.
2. Download one of the following files:
   - (Windows) `Alfresco-DeploymentEnterprise-3.3.1-Setup.exe`
   - (Linux) `Alfresco-DeploymentEnterprise-3.3.1-Linux-x86-Install`
3. At the **Language selection** prompt, click **OK**.
4. At the Install **Alfresco Standalone Deployment Receiver** prompt, click **Yes**.
5. In the **Welcome to the Alfresco Standalone Deployment Receiver window**, click **Next**.
6. In the **Choose Destination Location** window, click **Next** to accept the default location for Alfresco or choose another location.

   For example, `C:\alfresco\deployment` on Windows or `/opt/alfresco/deployment` on Linux.
7. In the **Deployment Settings** window, enter the following settings:

Deployment setting	Description
Temporary Data Location	The directory in which to store temporary data files.
Log Location	The directory in which to store log data.
Metadata Location	The directory in which to store metadata.
Target Location	The directory in which to store deployment files.
Name of default file system target	The default name of the file system target is `default`.

   For Windows directory locations, the backslashes need to be escaped. For example, use `C:\\directory1\\directory2`. Alternatively, you can use the slash character as a separator, for example, `C:/directory1/directory2`.
8. Click **Next**.
9. Enter a user name and password for the user account that will administer the deployment receiver.
10. If you are using RMI as your transport protocol, enter the port numbers for the following:

Deployment	Description
RMI Registry Port Number	The port number for the RMI registry. Choose the default of 44101 to avoid conflict the other services.
RMI Service Port Number	The port number to use for the RMI service. Choose this so that there are no conflicts with other services.

11. In the **Start Copying Files** window, click **Next**.

12. In the **InstallJammer Wizard Complete** window, click **Finish**.

The deployment receiver, out of the box, is configured with a single file system deployment target.

To change the settings, or to add further deployment targets, see Configuring WCM deployment receiver properties on page 120.

Installing OpenOffice

Within Alfresco, you can transform a document from one format to another, for example, a text file to a PDF file. To have access to these transformation facilities in Alfresco, you must install OpenOffice. This is optional, and can be done any time after Alfresco is installed.

1. Browse to the OpenOffice.org download site: http://download.openoffice.org
2. Download the latest (stable) version of OpenOffice for your platform.
3. When prompted, specify a download destination.
4. Browse to the location of your downloaded file, and install the application.

   A wizard guides you through the installation.
5. Accept the license agreement, and then click **Next**.
6. Enter Customer information as appropriate, and then click **Next**.
7. Select the set up type and **Custom**, and then click **Next**.
8. Change the installation directory to:
   - (Windows) `c:\Alfresco\OpenOffice`
   - (Linux) `/opt/alfresco/OpenOffice`
9. Optionally, select the files for which you want OpenOffice to be the default application, and then click **Next**.
10. Start one of the OpenOffice programs for the initial registration, and then close the program.
11. Modify the `ooo.exe=` property in the `<classpathRoot>/alfresco-global.properties` file to point to the OpenOffice binary `soffice.exe`.

    For Windows, set the path using the `\\` separator or use the forward slash `/` Unix path separator. For example: `c:\\Alfresco\\OpenOffice\\soffice.exe` or `c:/Alfresco/OpenOffice/soffice.exe`.

    For Solaris, ensure that the `<configRoot>/classes/alfresco/subsystems/OOoDirect/default/openoffice-transform-context.xml` file can start OpenOffice. Remove the quotes from the connection string values:

    ```
      <value>-
    accept=socket,host=localhost,port=8100;urp;StarOffice.ServiceManager</
    value>
      <value>-env:UserInstallation=file:///${ooo.user}</value>
    ```
12. If the Alfresco server is running, stop and restart the server.

You can configure the OpenOffice transformation settings using one of the OpenOffice subsystems. Refer to Configuring OpenOffice on page 105.

Installing ImageMagick

To enable image manipulation in Alfresco, you must install and configure ImageMagick. Alfresco uses ImageMagick to manipulate images for previewing.

1. Verify if ImageMagick is already installed on your system.

You can run the `convert` command, which is part of ImageMagick and usually located in `/usr/bin`.

2. If ImageMagick is not on your system, browse to the ImageMagick download site and install the appropriate package for your platform.
3. Modify the `img.root=` and `img.exe=` properties in the `<classpathRoot>/alfresco-global.properties` file to point to the ImageMagick root directory.

   For example, for Windows:

   a. Set the `img.root=` property to `img.root=C:/Alfresco/ImageMagick`.
   b. Set the img.exe= property to `img.exe=C:/Alfresco/ImageMagick/bin/convert.exe`.

   For Linux:

   a. Set the `img.root=` property to `img.root=/ImageMagick`.
   b. Set the img.exe= property to `img.exe=/ImageMagick/bin/convert.exe`.

   Ensure that you do not include a slash (`/`) at the end of the path. For example, `/ImageMagick/`

Installing Microsoft Office Add-ins

This task describes how to install Alfresco Add-ins for Microsoft® Office applications, such as Word, Excel, and PowerPoint. The Alfresco Add-Ins have been designed for Microsoft Office 2003.

Before you start, make sure that:

- The .NET Programmability Support option is installed for each of the Office applications that you are installing the Add-ins (such as Word, Excel, and PowerPoint). To find these options, run the Office Setup program and expand the list of available options for each application. You may require your original Office 2003 install media to add these required components.
- The installing user has Windows administrator privileges.
- Any Microsoft Office applications on your system are NOT running, including Outlook if you use Word as an email editor.

If you are running Office 2007 on Windows Vista, note that Microsoft have rewritten the WebDAV parts of Vista which means you will experience READ ONLY access to the Alfresco repository over WebDAV. This is a known problem with Vista and affects many applications, including Microsoft's own SharePoint Server. There is no known workaround at the time of writing. CIFS access is unaffected and works as it does with Windows XP. Therefore, use CIFS to obtain read/write access to Alfresco using Windows Vista.

1. Browse to the Alfresco Enterprise Edition download area.
2. Run the Microsoft setup file:

   `alfresco-enterprise-office2003-addins-3.3.1.zip`

   `alfresco-enterprise-office2003-addins-3.3.1.zip`

   This example refers to the Office installer that installs all three Add-ins. You can also use individual installers to add Alfresco to one of the three Office applications.

   These individual installers are:

   - `alfresco-enterprise-word2003-addin-3.3.1.zip`
   - `alfresco-enterprise-excel2003-addin-3.3.1.zip`

- `alfresco-enterprise-powerpoint2003-addin-3.3.1.zip`
- `alfresco-enterprise-word2003-addin-3.3.1.zip`
- `alfresco-enterprise-excel2003-addin-3.3.1.zip`
- `alfresco-enterprise-powerpoint2003-addin-3.3.1.zip`

3. Run `setup.exe`.

   If required, the setup program will download the required components from the Microsoft website. These components are .NET 2.0 framework, and Visual Studio 2005 Tools for Office Second Edition runtime.

   The setup is complete.
4. Run the Office application, for example, run Word.

   A Welcome window with configuration options displays. You can return to the configuration options at any time using the link at the bottom of the Add-In window.
5. In the **Web Client URL** field, enter the location of Alfresco Explorer.

   For example: `http://server:8080/alfresco/`
6. In the **WebDAV URL** field, append `webdav/` to the Alfresco Explorer URL.

   For example: `http://server:8080/alfresco/webdav/`
7. In the **CIFS Server** field, enter the path to the CIFS server. The Add-in will try to auto-complete this value, but you should verify for the correct address.

   For example: `\\server_a\alfresco\` or `\\servera\alfresco\`

   If you intend to use the CIFS interface to save documents via the Add-in, it is very important that you are authenticated automatically. Limitations in the Microsoft Office APIs mean that an error is shown instead of an authentication dialog box if the Alfresco CIFS interface rejects your connection attempt. If you are not in an Enterprise environment, where it may not be possible to configure automatic authentication, you can map a network drive to the Alfresco CIFS interface instead.
8. In the **Authentication** area, enter your Alfresco user name and password.

   The Add-In will always try to automatically log you in to Alfresco in order to present your checked-out documents and your assigned tasks. If you are using the CIFS interface, authentication is usually automatic. However, sometimes the Add-In needs to present your Alfresco user name and password for authentication. It is recommended that you enter and save your Alfresco credentials. All values are stored in the Windows registry and your password is encrypted against casual hacking attempts.
9. Click **Save Settings**.

Installing Flash Player

Alfresco Share uses the Flash Player for viewing Flash previews and also when you use the multi-file upload facility.

This is optional and may be installed after you have installed Alfresco.

1. Browse to the Adobe Flash Player download website: http://www.adobe.com/products/flashplayer.
2. Download the latest (stable) version of Flash Player for your platform.
3. Browse to the location of your downloaded file and install the application.

   A wizard guides you through the installation.
4. When the installation is complete, click **Close**.

Installing SWF Tools

Alfresco Share uses the pdf2swf utility of the SWF Tools for previewing PDF files. The pdf2swf utility generates one frame per page of fully formatted text inside a Flash movie. To install the pdf2swf utility, you must install the complete SWF Tools.

This is optional and may be installed after you have installed Alfresco.

Installing SWFTools on Windows

1. Browse to the SWF Tools website.
2. Download the latest (stable) version of the SWF Tools for your platform. The Windows version is designated with the suffix `.exe`.

   Download a version post 0.8.1 from 2007-02-28 because it does not support some functionalities Alfresco needs to render the preview.
3. Browse to the location of your downloaded file and and install the application.

   A wizard guides you through the installation.
4. Accept the license agreement and click **Next**.
5. Select the installation directory.
6. Select whether you want to install the SWF Tools for all users or only for the current user.
7. Click **Next** to begin the install process.

   By default, the options to **Create start menu** and **Create desktop shortcut** are selected.
8. Click **Finish**.
9. Modify the `swf.exe=` property in the `alfresco-global.properties` file to point to the SWF Tools root directory, for example: `swf.exe=C:/Alfresco/bin/pdf2swf`

   Ensure that you do not include a slash (`/`) at the end of the path. For example, `/usr/bin/`

The SWF Tools are installed. For the most up-to-date instructions on installing the SWF Tools, refer to the SWF Tools website.

Installing SWF Tools on Linux

Alfresco Share uses the features provided in the development snapshots of the tools. For Linux, there is no binary version, so you need to compile a development snapshot.

(Linux) Before you compile, ensure that the following packages are installed on your machine:

- `zlib-devel`
- `libjpeg-devel`
- `giflib-devel`
- `freetype-devel`
- `gcc`
- `gcc-c++`

You can download and install all of these packages using the following command:

```
yum install zlib-devel libjpeg-devel giflib-devel freetype-devel gcc gcc-c++
```

1. Browse to the SWF Tools website.
2. Download the latest version of the SWF Tools for your platform. The Unix version is designated with the suffix .tar.gz.

> Download a version post 0.8.1 from 2007-02-28 because it does not support some functionalities Alfresco needs to render the preview. The following version has been tested and verified by Alfresco as being fully functional: `http://www.swftools.org/swftools-2008-10-08-0802.tar.gz` (you may have to copy this URL and paste it into a download manager).

3. Unpack the tar.gz file.

   The install file contains detailed instructions on how to compile and install the SWF Tools.
4. Change to the directory containing the source code.
5. Type the following command to configure the package for your system:

   ```
   ./configure
   ```

   If you see a message on Red Hat Linux that states your operating system is unknown, then use the following setting: `./configure–build=x86_64-pc-linux-gnu`

   If you have an issue on Solaris with the lame libs, you can disable the making of portions of SWF Tools that use lame by using the following setting: `./configure -disable-lame`
6. Type the following command to compile the package:

   ```
   make
   ```

   Optionally, you can run the `make check` command to run any self-tests that come with the package.
7. Type the following command to install the programs, data files, and documentation:

   ```
   make install
   ```

   By default, the files are installed to the `/usr/local/bin` directory.
8. Modify the `swf.exe=` property in the `alfreso-global.properties` file to point to the SWF Tools root directory, for example: `swf.exe=/usr/bin/pdf2swf`

   > Ensure that you do not include a slash (`/`) at the end of the path. For example, `/usr/bin/`

The SWF Tools are installed. For the most up-to-date instructions on installing the SWF Tools, refer to the SWF Tools website.

Installing TinyMCE language packs

Translations in Alfresco use the language pack supplied in the default bundle. This default bundle includes English (en), French (fr), German (de), Japanese (jp), Spanish (es), and Italian (it).

If you have a translation that is not supplied with Alfresco, then you must add the appropriate TinyMCE language pack for the translation to work correctly.

If you installed Alfresco using one of the installation wizards or bundles, the default language packs are already installed.

1. Browse to the TinyMCE website: http://tinymce.moxiecode.com/download_i18n.php
2. Download the required TinyMCE language pack.
3. Unpack the language file:
   - For Share, unpack to: `<TOMCAT_HOME>/webapps/share/modules/editors/tiny_mce`
   - For Explorer, unpack to: `<TOMCAT_HOME>/webapps/alfresco/scripts/tiny_mce`
4. Ensure that the browser cache is cleared or refresh the page.

Installing an Alfresco Module Package

An Alfresco Module Package (AMP) is a bundle of code, content model, content, and the directory structure that is used to distribute additional functionality for Alfresco. This section describes how to install an AMP in an Alfresco WAR using the Module Management Tool (MMT). The MMT is used to install and manage AMP files, and it is included in the Alfresco installers.

The MMT is also available as a separately downloadable JAR file from the Alfresco release download area (`alfresco-mmt-3.3.1.jar`).

The MMT is also available as a separately downloadable JAR file from the Alfresco release download area (`alfresco-mmt-3.3.1.jar`).

For Tomcat, alternatively, run the `apply_amps` command in the root Alfresco directory, which applies all the AMP files that are located in the `amps` directory. Refer to Changing the default shell (Unix/Linux/Solaris) for shell scripts on page 208 for more information on running this command on Unix.

1. Browse to the `/bin` directory:
   - (Windows) `C:\Alfresco\bin`
   - (Linux) `/opt/alfresco/bin`
2. Run the following command:

   ```
   java -jar alfresco-mmt.jar install <AMPFileLocation> <WARFileLocation>
   [options]
   ```

   Where:

Option	Description
`<AMPFileLocation>`	The location of the AMP file that you want to install.
`<WARFileLocation>`	The location of the WAR file for your Alfresco installation.
`-verbose`	Install command [options]. Enables detailed output containing what is being updated and to where it is being copied.
`-directory`	Install command [options]. Indicates that the AMP file location specified is a directory. All AMP files found in the directory and its sub directories are installed.
`-force`	Install command [options]. Forces installation of AMP regardless of currently installed module version.
`-preview`	Install command [options]. Previews installation of AMP without modifying WAR file. It reports the modifications that will occur on the WAR without making any physical changes, for example, the changes that will update existing files. It is good practice to use this option before installing the AMP.
`-nobackup`	Indicates that the WAR will not be backed up before the AMP is installed.

   This command installs the files found in the AMP into the Alfresco WAR. If the module represented by the AMP is already installed and the installing AMP is of a higher release version, then the files for the older version are removed from the WAR and replaced with the newer files.

The following commands show examples of how to install the `example-amp.amp`, and assumes that the AMP file is in the same directory as the WAR file:

```
java -jar alfresco-mmt.jar install example-amp.amp alfresco.war -preview
```

Review the modification to check the changes that will update any existing files.

The following example will install the AMP file:

```
java -jar alfresco-mmt.jar install example-amp.amp alfresco.war -verbose
```

The modified Alfresco WAR can then be redeployed back into your application server.

On restarting the application server, the console will show that the custom class was initialized during startup.

3. Verify that the AMP is installed using the MMT `list` command. For example:

```
java -jar alfresco-mmt.jar list <WARFileLocation>
```

This command provides a detailed listing of all the modules currently installed in the WAR file specified.

When the repository is next started, the installed module configuration will be detected, and the repository will be bootstrapped to include the new module functionality and data.

It is not recommended that you overwrite an existing file in an AMP, however it is sometimes necessary. The MMT makes a backup copy of the updated file and stores it in the WAR. When an update of the module occurs and the old files are removed, this backup will be restored prior to the installation of the new files. Problems may occur if multiple installed modules modify the same existing file. In these cases, a manual restore may be necessary if recovery to an existing state is required.

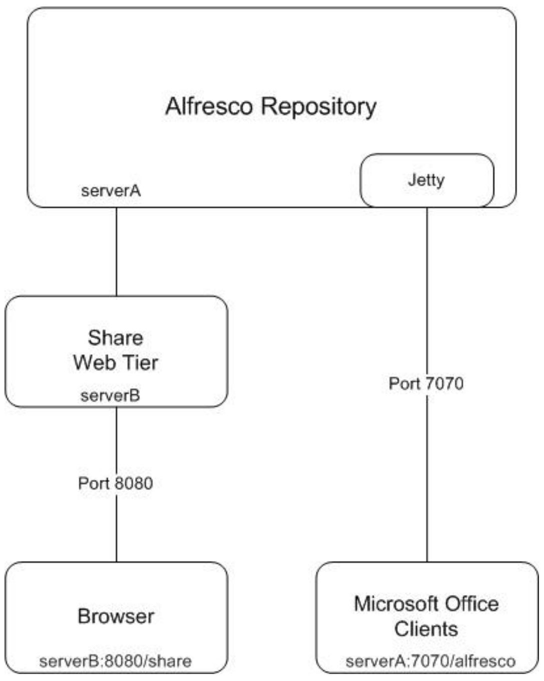
Some application servers (notably Tomcat) do not always fully clean up their temporary working files, and this can interfere with successful installation of an AMP file. To remedy this situation, it is recommended that you delete (or move) the Tomcat `work` and `temp` directories while Tomcat is shut down.

Installing Microsoft Office SharePoint Protocol Support

The Microsoft Office SharePoint Protocol Support offers Microsoft users greater choice by providing a fully-compatible SharePoint repository that allows the Microsoft Office Suite applications (for example, Word, PowerPoint, Excel) to interact with Alfresco as if it was SharePoint. This enables your users to leverage the Alfresco repository directly from Microsoft Office.

You can also use Microsoft Office SharePoint Protocol Support to enable online editing for Office documents within Alfresco Share. It enables your users to modify Office files without checking them in and out. Alfresco locks the file while it is being modified and releases the lock when the file is saved and closed.

The following diagram shows the architecture of the SharePoint Protocol Support in relation to an Alfresco installation.



The SharePoint Protocol Support architecture embeds a Jetty web server within the Alfresco repository. The Microsoft Office clients communicate directly with the Jetty server using WebDAV-like calls with proprietary extensions and on different port number from Alfresco Share. This port number can be configured in the SharePoint Protocol Support properties files.

Installing the SharePoint Protocol Support AMP

The SharePoint Protocol support functionality is installed from an Alfresco AMP. If you use the Windows or Linux installers to install Alfresco, the SharePoint Protocol Support is installed by default. These instructions describe how to install the SharePoint Protocol Support into the Alfresco WAR. When you install this file, it responds to the SharePoint requests from Office, and therefore allows Alfresco to appear as the SharePoint server.

1. Shut down your Alfresco server.
2. Browse to the Enterprise download area.
3. Download the `vti-module.amp` file.
4. Move the file to the `amps` directory.
5. Install the `vti-module.amp` file into the `alfresco.war` file using the Module Management Tool (MMT).

   The `vti-module.amp` file holds the functionality for the SharePoint connector.

   For Tomcat, alternatively, run the `apply_amps` command in the root Alfresco directory, which applies all the amps that are located in the `amps` directory. Refer to Changing the default shell (Unix/Linux/Solaris) for shell scripts on page 208 for more information on running this command on Unix.
6. Start your Alfresco server to expand the directory structure.
7. Verify that you have applied the SharePoint AMP to Alfresco by checking that you have the following directory:

   ```
   /webapps/alfresco/WEB-INF/classes/alfresco/module/
   org.alfresco.module.vti/context
   ```

Configuring SharePoint Protocol Support

The SharePoint Protocol Support functionality uses the properties in the default configuration file called `vti.properties` in `<TOMCAT_HOME>/webapps/alfresco/WEB-INF/classes/alfresco/module/org.alfresco.module.vti/context`. Custom properties and overrides can be set in the `alfresco-global.properties` file.

Ensure that you have applied the SharePoint Protocol Support AMP, following the instructions in Installing the SharePoint Protocol Support AMP on page 38.

1. Open the `alfresco-global.properties` file.
2. Add the following properties:

   ```
   vti.server.port=7070
   vti.alfresco.deployment.context=/alfresco
   vti.alfresco.alfresoHostWithPort=http://your-host:8080
   vti.share.shareHostWithPort=http://your-share-host:8080
   ```

   The following table describes the properties.

Property	Description
`vti-server.port`	Use this property to configure on which port the SharePoint server listens. The default port number is 7070.
`vti.alfresco.deployment.context`	Use this property to specify the URL that you enter in the Microsoft Office Shared Workspace. The default is set to `/alfresco`. For example, the default `/alfresco` sets the URL to `http://your-share-host:7070/alfresco`.
`vti.alfresco.alfresoHostWithPort`	Use this property to specify the IP address or host name of the Alfresco server. Replace `your-host` with the location of your Alfresco server.
`vti.share.shareHostWithPort`	Use this property to specify the Share instance. Replace `your-share-host` with the location of your Share instance. This property includes the prefix `http://` which allows for HTTPS support.

   The `vti.properties` file in `<TOMCAT_HOME>/webapps/alfresco/WEB-INF/classes/alfresco/module/org.alfresco.module.vti/context` contains the full list of properties. Do not change the properties in this file; use only the `alfresco-global.properties` file.
3. Save the `alfresco-global.properties` file.
4. Restart your Alfresco server.

The Microsoft SharePoint Protocol Support functionality is installed and configured.

Configuring SharePoint Protocol for Online Editing

There is a known issue with Office 2003 and 2007 Office documents opening as read-only in Internet Explorer for all versions before Vista SP1.

Please refer to the knowledge base article 870853 on the Microsoft website to enable the Online Editing functionality.

Installing the Alfresco Records Management module

The Alfresco Records Management module is designed to work with an Alfresco installation, applying new functionality to Alfresco Share.

Records Management installation procedure

This section describes the general procedure for installing Records Management.

Ensure that you have Alfresco Enterprise Edition 3.3 SP1 installed on your machine.

1. Download and apply the Records Management AMP files to your existing Alfresco installation.
2. Restart the Alfresco server.
3. Log in to Share to add the Records Management dashlet.
4. Create the Records Management site.

Applying the Records Management AMP files

To install Records Management, you need to apply the AMP files to an existing Alfresco installation.

The installation procedure uses the following Records Management AMP files:

`alfresco-dod5015-share.amp`	This AMP file contains the additional Records Management functionality that is applied to an existing Alfresco Share user interface. The functionality should be applied to the `tomcat/webapps/share` directory.
`alfresco-dod5015.amp`	This AMP contains the additional Records Management functionality that is applied to an existing Alfresco installation.

The following procedure applies the Records Management AMP files to an existing Alfresco Enterprise Edition 3.3 SP1 running in a Tomcat application server. If you are using an application server other than Tomcat, use the Module Management Tool (MMT) to install the AMP files. If you are using the MMT, you must apply the `alfresco-dod5015-share.amp` to the `share.war` file.

1. Move the `alfresco-dod5015.amp` file to the `amps` directory.
2. Move the `alfresco-dod5015-share.amp` file to the `amps-share` directory.
3. Stop the Alfresco server.
4. Run the `apply_amps` command, which is in the root Alfresco directory.

   This command applies all the AMP files that are located in the `amps` and `amps-share` directories.

   For Linux, edit the `CATALINA_HOME` variable, as needed.
5. Start the Alfresco server.
6. Start Alfresco Share by browsing to:

   `http://<your-server-name>:8080/share`

Adding the Records Management dashlet

When you have installed the Records Management module and started your server, you need to add the Records Management dashlet into your personal dashboard.

This dashlet allows you to create the pre-defined Share site for Records Management functionality.

1. Log in to Alfresco Share.
2. Click **Customize Dashboard**.
3. Click **Add Dashlets**.
4. Locate the **Records Management config** dashlet in the **Add Dashlets** list.
5. Drag the **Records Management config** dashlet to the desired column.
6. Click **OK**.

The Records Management dashlet displays in the selected column in your Records Management site dashboard.

The Records Management dashlet provides actions that allow you to:

- Create the Records Management site
- Access the Records Management site once it is created
- Load sample test data for an example of how to structure a File Plan
- Access the management console

Creating the Records Management site

This task assumes that you have added the Records Management dashlet into your dashboard.

1. On the Records Management dashlet, click **Create Records Management Site**.

   A message displays to confirm that the site had been created.
2. Refresh the Share dashboard, either:
   - Refresh the browser, or
   - Logout of Share, and then logon again using the same user name and password.

The Records Management site now displays in the **My Sites** dashlet.

Kofax Release script

This section provides details on installing, configuring, and using the Alfresco Kofax Release script.

Integrating Kofax and Alfresco provides complete content management support including the capture, management, and publishing of content. Kofax Capture captures content from various sources, typically through scanning and OCR. The captured information is then released to Alfresco to be managed in an ad-hoc manner or using pre-defined business processes.

The Kofax architecture provides a plug-in architecture for deploying a Kofax Release script that is responsible for mapping and transferring the information captured by Kofax to the destination application or database.

The Alfresco Kofax Release script comprises a Release script plug-in that is installed within the Kofax Capture application and a set of Alfresco web scripts installed on the Alfresco server.

The Alfresco Kofax Release script provides the following capabilities:

- Alfresco server connection (connection URL, user name, password)
- Destination folder in which to store the captured documents (folders may be automatically created based on index field values)
- Mapping of Kofax Capture indexing information and files to Alfresco properties
  - Support for Alfresco types, sub-types, and aspects, and their associated properties
  - Mapping of Kofax Image (TIFF), Text (OCR), or PDF files to Alfresco content properties
- Automatic versioning, overwrite, and error handling for existing documents

System requirements and prerequisites

The Alfresco Kofax Release script has the following prerequisites:

- Alfresco Version 3.1 or higher
- Kofax Capture 8.x

You need to have a working knowledge of Kofax Capture and Alfresco.

Installation and advanced configuration requires experience with Alfresco Module Packages (AMPs) and defining Alfresco models. For more information Kofax, refer to the Kofax Capture documentation.

Installing Kofax Release script

Installing the Alfresco Kofax Release script is a two-part process that involves:

1. Installation of the Alfresco Kofax Release script Alfresco Module Package (AMP) file using the Alfresco Module Management Tool.
2. Installation of the Alfresco Kofax Release script binaries in your Kofax Capture installation.

Installing the Alfresco Kofax Release script AMP

The following describes how to install the Alfresco Kofax Release script AMP file (`alfresco-kofax-beta.amp`) on your Alfresco server.

1. Shut down your Alfresco server.
2. Move or copy the `alfresco-kofax-beta.amp` file to the `amps` directory in your Alfresco installation.
   - (Windows) `c:\Alfresco\amps`
   - (Linux) `/opt/alfresco/amps`
3. From the command line, browse to the Alfresco `bin` directory.
   - (Windows) `c:\Alfresco\bin`
   - (Linux) `/opt/alfresco/bin`
4. Install the Alfresco Kofax AMP using the Module Management Tool.
   - (Windows) `java -jar alfresco-mmt.jar install c:\Alfresco\bin\amps\alfresco-kofax-beta.amp c:\Alfresco\tomcat\webapps\alfresco.war`
   - (Linux) `java -jar alfresco-mmt.jar install /opt/alfresco/amps/alfresco-kofax-beta.amp /opt/alfresco/tomcat/webapps/alfresco.war`

   Alternatively for Tomcat, you can run the `apply_amps.bat` command in the root Alfresco directory to install the `alfresco-kofax-beta.amp`. This batch file applies all the AMPs that are located in the amps directory.
5. Remove your existing expanded Alfresco web application directory to allow updates to be picked up when the server restarts.
   - (Windows) `c:\Alfresco\tomcat\webapps\alfresco.war`
   - (Linux) `/opt/alfresco/tomcat/webapps/alfresco`

Installing the Alfresco Kofax Capture Release script binaries

The following steps describe how to install the binaries required to set up and configure the Kofax Release script in your Kofax Capture installation.

You must have Windows administrator privileges to install Kofax Capture Release script binaries. If you do not have administrator rights, you may encounter errors and the script may fail to install.

1. Unzip the `alfresco-kofax-client-binaries-3.2-beta.zip` file to your Kofax Capture `bin` directory.

   For example, (Windows) `c:\Program Files\Kofax\Capture\bin`
2. Start the Kofax Capture Administration Module.
3. In the Kofax Administration module, click **Tools > Release Script Manager**.
4. From the **Release Script Manager** dialog box, click **Add** and then browse to the directory of the unzipped files.
5. Select `Alfresco.Kofax.Release.inf`, and click **Open**.
6. Click **OK** to register the release script.
7. Close the open dialog boxes to complete the process.

Configuring the Alfresco Kofax Release script

This section provides instructions on setting up the Alfresco Kofax Release script. These instructions assume you are familiar with Kofax Capture and have created a Kofax Capture batch class. For information on setting up batch classes in Kofax Capture, refer to the Kofax Capture documentation.

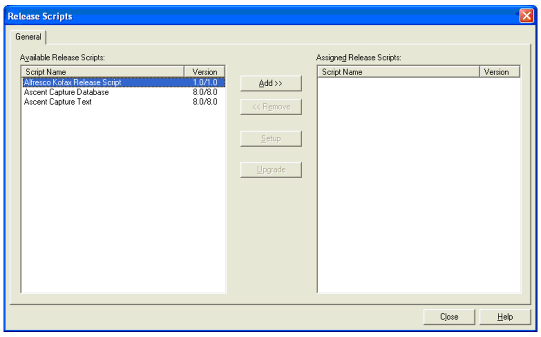
In Kofax Capture, release scripts are associated with document classes. The script is configured to define where and how the documents will be released, including:

- URL to connect to your Alfresco server
- Alfresco user name and password used to create the documents in Alfresco
- Location in the Alfresco repository where documents will be released
- Options for handing existing documents, such as Overwrite, Version, Release to Default Folder, or Report an Error
- Alfresco document type
- Mapping between the Alfresco properties (including those based on type and configured aspects), and the Kofax indexing fields to be populated by the release script

Associating the Alfresco Kofax Release script with a document class

Once you have set up a batch class with an associated document class in Kofax Capture, you can associate a Release script with the batches document class. As part of this process, you are prompted to enter the connection details for your Alfresco server.

1. Start the Kofax Capture Administration Module.
2. Select the **Batch class** tab from the **Definitions** panel, and right-click the applicable document class. (Expand the Batch class item to select associated document classes.)
3. From the **Context** menu, select **Release Scripts**.



Available Release Scripts:

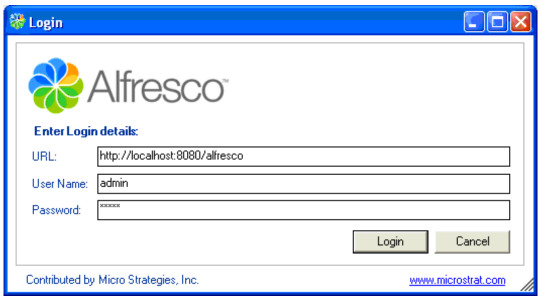
Script Name	Version
Alfresco Kofax Release Script	1.0/1.0
Ascent Capture Database	8.0/8.0
Ascent Capture Text	8.0/8.0

Assigned Release Scripts:

Script Name	Version

The **Release Scripts** dialog box displays, listing all available release scripts. Available release scripts are those that are registered with Kofax Capture.

4. From the **Release Scripts** dialog box, select the Alfresco Kofax Release Script, and click **Add**.



Enter Login details:
URL: http://localhost:8080/alfresco
User Name: admin
Password: *****

Contributed by Micro Strategies, Inc. www.microstrat.com

The **Login** dialog box displays.

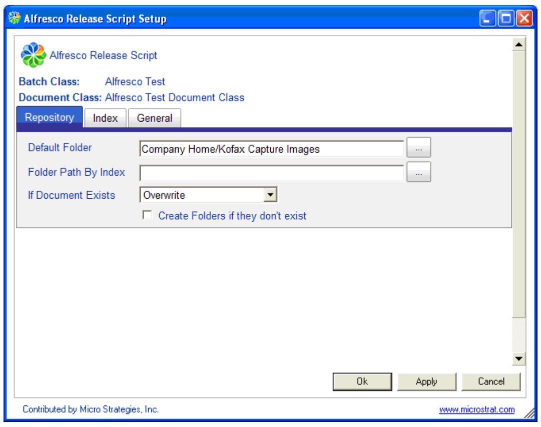
5. Enter your Alfresco server URL, user name, and password.
6. Click **Login**.

Configuring the Alfresco Kofax Release script

The Kofax Release script is configured using three main tabs. The following sections describe each of these and the options available.

Repository tab

The **Repository** tab is used to configure where documents are stored in the Alfresco repository and how existing documents are handled.



Batch Class: Alfresco Test
Document Class: Alfresco Test Document Class
Default Folder: Company Home/Kofax Capture Images
Folder Path By Index:
If Document Exists: Overwrite
☐ Create Folders if they don't exist

Contributed by Micro Strategies, Inc. www.microstrat.com

The Repository tab has the following options:

Default Folder

Defines the root Alfresco space in which documents will be created.

The user that connects to Alfresco must have permission to create documents in this space.

Folder Path by Index

Allows the folder path to be dynamically generated based on indexing values. Substitute Alfresco property name(s) to be used as part of the folder path.

For example, the following will store all documents with the same `Invoice Date` property in folders according to the invoice date:

```
Company Home/Invoices/[Invoice Date]
```

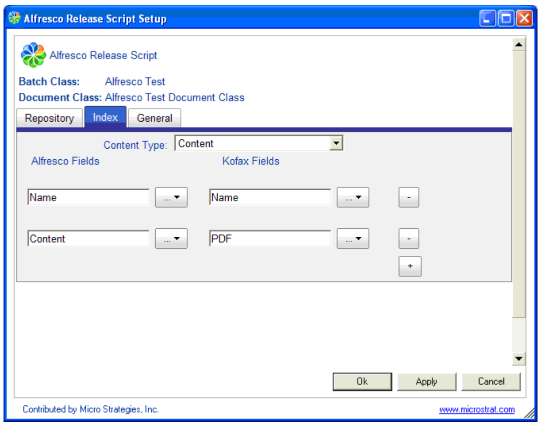
If Document Exists

A document already exists if a document of the same name already exists in the folder in which the document is being released. The following defines how the Release script will handle existing documents.

- **Overwrite**: Replaces the document with the one being currently released.
- **Version**: Creates a new version of the document.
- **Release To Default Folder**: If the folder path specified in the **Folder Path By Index** field has an existing document with the same name, the document will be put into the location specified in the **Default Folder** field.
- **Throw Error**: The release fails with the error `Duplicate child name not allowed`.
- **Create Folders if they don't exist**: If selected, this will automatically create folders that do not exist as defined by the previous **Folder Path by Index** settings. If this is not selected, and the folder path(s) do not exist, an error will occur and the document will fail.

Index tab

The **Index** tab defines the Alfresco document type used for released documents, and the mappings between Kofax index fields and Alfresco properties.



Batch Class: Alfresco Test
Document Class: Alfresco Test Document Class

Content Type: Content

Alfresco Fields	Kofax Fields
Name	Name
Content	PDF

Each row defines the mapping between an Alfresco property and a Kofax indexing field. The **Content Type** and **Alfresco Fields** values available can be controlled through configuration.

Content Type

The Alfresco content type that will be used for documents created by the Release script. It can be a custom content type or content.

Alfresco Fields

Use the drop-down list to pick Alfresco properties based on the available types and aspects that will be populated with Kofax Capture index data.

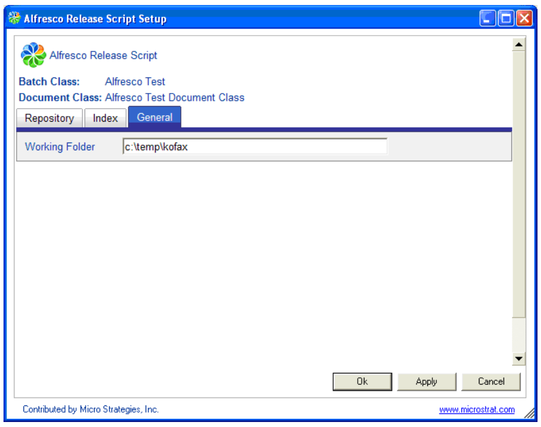
Kofax Fields

Use the drop-down list to pick the Kofax Capture field to map to the Alfresco property. The **Text Constant** field can provide a fixed text value for the field.

You must define an Alfresco **Name** field and an Alfresco **Content** field, as shown in the previous figure. The **Content** field is used to store the image file, such as Image (TIF), PDF, or Text (OCR).

General tab

The **General** tab defines the working folder used by Kofax Capture for temporary file storage during the release process.


Alfresco Release Script Setup

Alfresco Release Script

Batch Class: Alfresco Test

Document Class: Alfresco Test Document Class

Working Folder: c:\temp\kofax

Contributed by Micro Strategies, Inc.

www.microstrat.com

Working Folder
Set this to a folder where the user running the script has write access on the local Kofax Capture machine.

Publishing a batch class

After you select all your batch class settings, you must publish your batch class before you can use it.

The publishing process checks the integrity of the settings in your batch class and makes the batch class available for use. If problems are found with any of the settings, error and warning messages will display, along with the recommended actions for fixing the problems.

If you edit your batch class, you must publish your batch class again before your changes can be used. Your changes will not be applied to batches created before the new publication date.

1. Start the Kofax Capture Administration module to display the main screen.
2. Select the **Batch class** tab from the **Definitions** panel, and right-click the applicable batch class.
3. From the **Context** menu, select **Publish**.
4. From the **Publish** window, select your batch class and click **Publish**.

   Kofax Capture will check all of your batch class settings and display the results in the **Results** box.

   If no problems are detected, the message “Publishing successful” displays. If problems are detected, warning/error messages will display along with recommended actions to resolve the problems. Perform the recommended actions, and then try to publish the batch class again.
5. Run some sample batches through the system to test the operation of the release script.

After successfully publishing, you can create batches based on your batch class. As your batches flow through your Kofax Capture system, they will be routed from module to module. The modules that are used to process a batch, and the order that processing occurs, are specified as part of the batch class definition for the batch.

Refer to the Kofax Capture Help for more information about batch classes.

Releasing batches

The Kofax Capture Release module will process batches based on the settings of the associated batch classes. This module is responsible for releasing documents, as well as index data using the attributes defined during release setup.

The Kofax Capture Release module usually runs as an unattended module on a Windows workstation, periodically polling the module for available batches. It may be configured to run during off-hours to avoid any impact to the throughput of Kofax Capture and/or the network system.

1. Start the Kofax Capture Release module by selecting **Start > Programs > Kofax Capture > Release**.

   All batches queued for release will be processed after initiation of the module.

   Once your batch is released, it will be removed from Kofax Capture. If any documents or pages are rejected, the batch will be routed to the Kofax Capture Quality Control module.
2. To exit the Kofax Capture Release module, select **Batch > Exit** from the module menu bar.

Refer to the Kofax Capture Help for more information about releasing batches.

Advanced configuration: custom types, aspects, and properties

By default, the Release Setup web script (`\service\kofax\releasesetup`) displays all types, aspects, and their associated properties available in your Alfresco repository.

The Release script can be configured to limit this list to only show only those values that are applicable to your use case. A web script configuration file is used to define the items to be displayed.

The Release script configuration file uses a structure similar to that used by the model definitions themselves. Add the types and/or aspects and the relevant properties to the `releasescript.get.config.xml` file to define the options you want available. See the sample configuration provided for examples.

For information on defining your own model for types and aspects, refer to the Alfresco Wiki page **Data Dictionary Guide**.

1. Locate the `releasesetup.get.config.xml.sample` file. For Tomcat this will be located at:

   ```
   tomcat\WEBINF\classes\alfresco\templates\webscripts\com\microstrat\kofax
   \releasesetup.get.config.xml.sample
   ```

   This is the default location used by the Tomcat application server. The location of the file may vary depending on the application server used by your Alfresco installation.
2. Rename `releasesetup.get.config.xml.sample` to `releasesetup.get.config.xml`.
3. Reload your web script using the Web Script Index page as follows:
   a. Go to `http://YOURHOST:8080/alfresco/service/index`.
   b. Click **Refresh Web Scripts**.
4. Open the Release Script **Index** tab.

   This will now only allow selection of types, aspects, and properties as defined in the configuration file.

Removing the Alfresco Kofax Release script

The following steps describe how to remove the Alfresco Kofax Release script from your Kofax installation.

1. Start the Kofax Capture Administration module.
2. Remove the Alfresco Kofax Release script from any document classes using the script:
   a. Right-click the applicable document class. (Expand the batch class item to select associated document classes.)
   b. From the **Context** menu, select **Release Scripts**.

    c. From the **Release Scripts** dialog box, select the Alfresco Kofax Release Script from the list of **Assigned Release Scripts**, and click **Remove**.

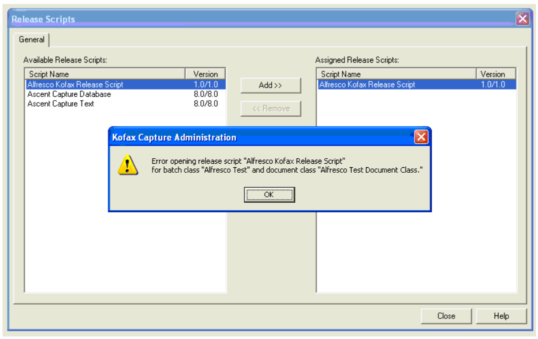
3. Repeat step 2 for all document classes using the Alfresco Kofax Release script.
4. In the Kofax Administration module, click **Tools > Release Script Manager**.
5. Select **Alfresco Kofax Release Script**, and click **Remove**.
6. To remove the installation files, manually delete the following files from your Kofax Capture `bin` directory.
    - `Alfresco.Kofax.Release.Core.dll`
    - `Alfresco.Kofax.Release.Core.Logging.xml`
    - `Alfresco.Kofax.Release.Core.xml`
    - `Alfresco.Kofax.Release.inf`
    - `Alfresco.Kofax.Release.WebScripts.dll`
    - `Antlr.runtime.dll`
    - `Common.Logging.dll`
    - `Jayrock.Json.dll`
    - `log4net.dll`
    - `Spring.Core.dll`

Troubleshooting the Kofax Release script

The following section describes how to troubleshoot the Kofax Release script.

Error adding the Alfresco Kofax Release script to a document class

If you see and error message “Error opening release script “Alfresco Kofax Release Script" when adding the script to a document class, it may be an indication that you have not copied the binaries to your Kofax Capture `bin` directory.



Release Scripts

Available Release Scripts:

Script Name	Version
Alfresco Kofax Release Script	1.0/1.0
Ascent Capture Database	8.0/8.0
Ascent Capture Text	8.0/8.0

Assigned Release Scripts:

Script Name	Version
Alfresco Kofax Release Script	1.0/1.0

Kofax Capture Administration

Error opening release script "Alfresco Kofax Release Script"
for batch class "Alfresco Test" and document class "Alfresco Test Document Class."

Ensure that the following files are in the `bin` directory:

- `Alfresco.Kofax.Release.Core.dll`
- `Alfresco.Kofax.Release.Core.Logging.xml`
- `Alfresco.Kofax.Release.Core.xml`
- `Alfresco.Kofax.Release.inf`
- `Alfresco.Kofax.Release.WebScripts.dll`
- `Antlr.runtime.dll`
- `Common.Logging.dll`
- `Jayrock.Json.dll`
- `log4net.dll`

- `Spring.Core.dll`

Release Error: [Release Script Returned -1. Your release script may need to be re-installed.]

This is a generic Kofax error. The most likely cause is that an invalid working folder has been specified when setting up the release.

Ensure that you have entered a valid folder path in the **Working Folder** field on the **General** tab.

Other causes of this error include missing dependencies in the installation. Check that you have installed all the required files the `bin` directory.

IBM Lotus Quickr integration with Alfresco

This section provides information on integrating Alfresco with the IBM Lotus software suite of products (Lotus Quckr, Lotus Connections, and Lotus Notes).

Installing the Lotus Quickr AMP

This section describes how to install and configure the AMP that integrates Lotus Quickr with Alfresco.

To integrate Lotus Quickr with Alfresco, you must have already installed Alfresco Enterprise Edition 3.3 and Lotus Quickr 8.1.1. To use Quickr Connectors with Alfresco, you must also have the following:

- Lotus Notes 8.5
- Microsoft Office 2003 or Microsoft Office 2007
- Windows XP

1. Download alfresco-quickr.amp from Alfresco Enterprise Edition download area.
2. Stop the Alfresco server.
3. Copy the `alfresco-quickr.amp` file to the `amps` directory.
4. Apply the `alfresco-quickr.amp` file into the `alfresco.war` file using the Module Management Tool (MMT). Refer to Installing an Alfresco Module Package on page 36 for more information.

   Alternatively, for Tomcat, run the `apply_amps` command, which is in the root Alfresco directory. This batch file applies all the AMP files that are located in the `amps` directory.
5. Start your Alfresco server to deploy the newly installed AMP.
6. Move and rename the following files:
   a. `<configRoot>\classes\alfresco\module\ org.alfresco.module.quickr \context\custom-lotus-ws-context.xml.sample` to `<configRoot>\classes \alfresco\extension\custom-lotus-ws-context.xml`
   b. `<configRoot>\classes\alfresco\module\org.alfresco.module.quickr \context\custom-lotus.properties.sample` to `<configRoot>\classes \alfresco\extension\custom-lotus.properties`
7. Open the `custom-lotus.properties` file, and then edit the following settings:

Setting	Description
`lotus.ws.version`	Add the version of Quickr Protocol implemented by Alfresco. The default is 8.0.1. You do not need to change this version for the Technical Preview version of Alfresco AMP file.

Setting	Description
`lotus.server.host`	Add the server host name where Alfresco is installed. For example, `localhost`.
`lotus.server.port`	Add the port number for Quickr to talk with Alfresco. For example, the default is 6060.
`lotus.share.document.url`	`http://${lotus.server.host}:8080/share/page/site/{0}/document-details?nodeRef={1}`
`lotus.share.folder.url`	`http://${lotus.server.host}:8080/share/page/site/{0}/documentlibrary\#path\={1}`
`lotus.share.site.url`	`http://${lotus.server.host}:8080/share/page/site/{0}/dashboard`

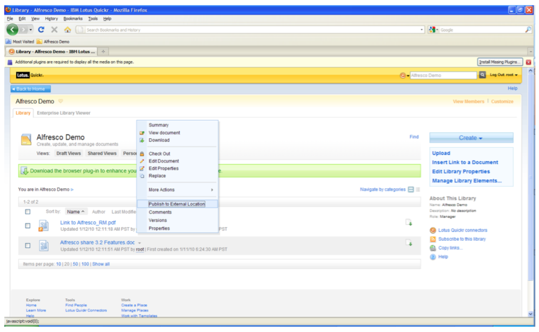
8. Start the Alfresco server.

   Lotus Quickr can now communicate with Alfresco as an ECM server.

Publishing content from Quickr to Alfresco

You can publish documents to Alfresco from the Lotus Quickr Library.

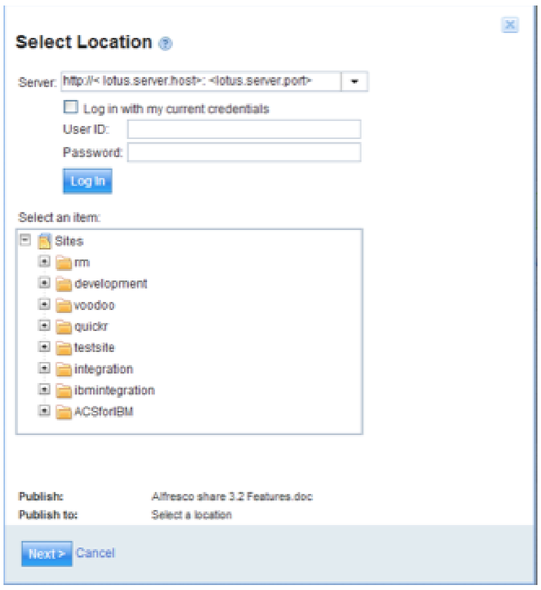
1. In Lotus Quickr, right-click on a document, and then select **Publish to External Location**.

   You see the location of Alfresco sites.

2. During the publish process, enter the server details and the Alfresco Share login credentials.

   For example, the server details are `http://< lotus.server.host>:<lotus.server.port>` as configured.


Select Location

Server: http://< lotus.server.host>: <lotus.server.port>

☐ Log in with my current credentials

User ID:

Password:

Log In

Select an item:

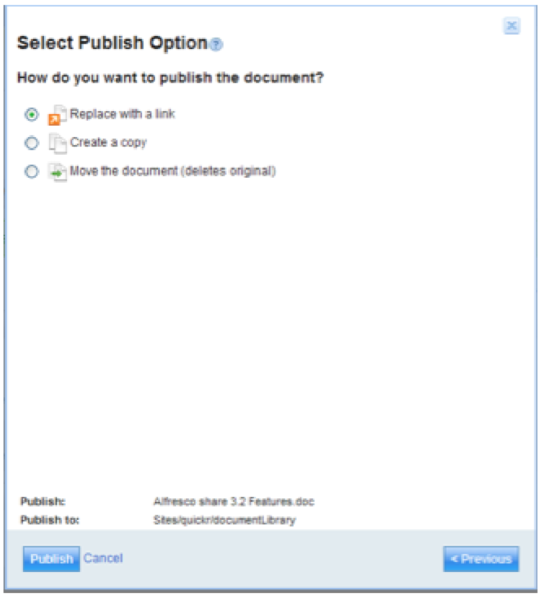
- Sites
  - rm
  - development
  - voodoo
  - quickr
  - testsite
  - integration
  - ibmintegration
  - ACSforIBM

Publish:	Alfresco share 3.2 Features.doc
Publish to:	Select a location

Next > Cancel

3. Alfresco supports all the three publishing options that Quickr provides:

   - **Replace with a Link**: the document is removed from Quickr library and added to Alfresco. A link pointing to the document in Alfresco is added to the Quickr Library. Click the link in Quickr to open the document in Alfresco.
   - **Create a Copy**: the document remains in Quickr library, and a copy is added to Alfresco.
   - **Move the Document**: the document is deleted from the Quickr library and is added to Alfresco.


Select Publish Option

How do you want to publish the document?

[✓ Replace with a link]
Create a copy
Move the document (deletes original)

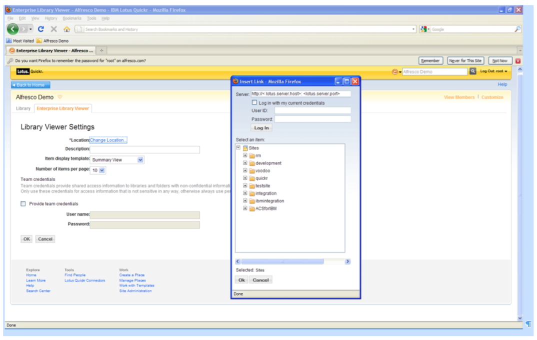
Publish:	Alfresco share 3.2 Features.doc
Publish to:	Sites/quickr/documentLibrary

Publish Cancel < Previous

Configuring Quickr to browse Alfresco as the Enterprise library

In Quickr, you can use the Enterprise Library Viewer to browse Alfresco as an enterprise library.

1. In Quickr, select the **Enterprise Library Viewer** tab.
2. Browse to Alfresco as an enterprise library.



3. To configure the settings, select the **Enterprise Library Viewer** tab, and then select Alfresco as the server location.
4. Browse the sites in Quickr and select content from Alfresco in Quickr.

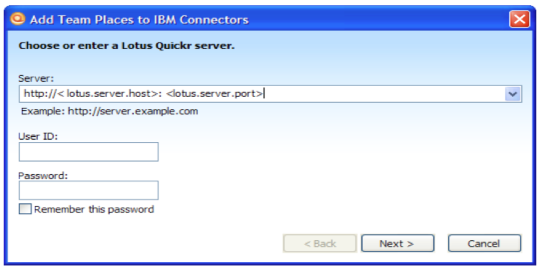
   The content opens in Alfresco.

Accessing Alfresco as a Team Place using Quickr connectors

You can also add Alfresco as a Team Place to provide access to Alfresco from the standard Quickr connectors. This enables you to perform Alfresco content management operations using the following:

- Lotus Notes: Lotus Notes IBM Lotus Notes 8.5
- Microsoft Office 2003 or Microsoft Office 2007
- Microsoft Windows Explorer for Windows XP

1. Access the Connector for which you want to add Alfresco as a Team Place.
2. Add a Team Place, specifying the Alfresco URL (as configured during installation) as the server.



Add Team Places to IBM Connectors

Choose or enter a Lotus Quickr server.

Server:
http://< lotus.server.host>: <lotus.server.port>

Example: http://server.example.com

User ID:

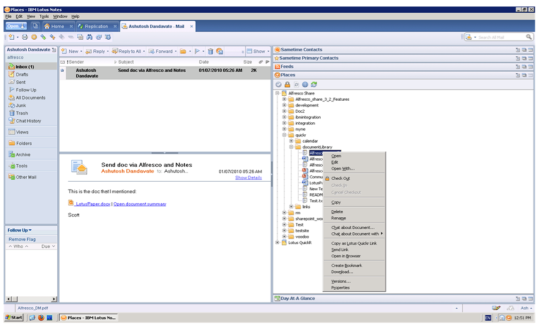
Password:

Remember this password

< Back | Next > | Cancel

3. From the Team Place, you can perform operations such as check in, check out, versioning, and sending documents as a link using email.

   The following example shows Alfresco Share as Team place in a Lotus Notes client.



Running Alfresco

This section describes how to start and stop the following:

- Alfresco server
- Alfresco Explorer
- Alfresco Share
- Virtualization server
- Standalone deployment receiver

Before running Alfresco, you may need to modify the configuration settings. If you have installed Alfresco using one of the installation wizards, the configuration is set for you. Refer to the Configuring Alfresco on page 56 chapter for more information on how to set up Alfresco for your system.

Starting the Alfresco server

Once you have installed Alfresco, you can test the installation by starting the server.

- (Windows)
  - Browse to `C:\alfresco`, and double-click `alf_start.bat`, or
  - If you installed Alfresco using the installer, click the **Start** menu, and select **All Programs > Alfresco Enterprise > Start Alfresco Server.**

  A command prompt opens with a message indicating the server has started.

  ```
  INFO: Server startup in nnnn ms
  ```
- (Linux) Browse to `/opt/alfresco/` and run `alfresco.sh start`.

  The default shell for this script is `sh`. You can edit the `alfresco.sh` file to change to your preferred shell. For example, change the `#!/bin/sh` line to `#!/bin/bash`.

Starting Alfresco Share

Once you have installed Alfresco, you can start Alfresco Share using a browser.

1. Browse to the location of your Alfresco installation. If you installed Alfresco on your local machine, browse to `http://localhost:8080/share`.

   In Windows, alternatively, you can click the **Start** menu, and select **All Programs > Alfresco Enterprise Edition > Alfresco Share**.

   Alfresco Share opens.
2. Log in using `admin` as the default user name and password.

Starting Alfresco Explorer

Once you have installed Alfresco, you can start Alfresco Explorer using a browser.

1. Browse to the location of your Alfresco installation. If you installed Alfresco on your local machine, browse to `http://localhost:8080/alfresco`.

   In Windows, alternatively, you can click the **Start** menu, and select **All Programs > Alfresco Enterprise Edition > Alfresco Explorer**.

   Alfresco Explorer opens.
2. Log in using `admin` as the default user name and password.

Stopping the Alfresco server

- (Windows)
    - Browse to `C:\alfresco`, and double-click `alf.stop.bat`, or
    - Click the **Start** menu, and select **All Programs > Alfresco Enterprise Edition > Stop Alfresco Server**.

  The command prompt that opened during startup closes. Alfresco has now stopped.
- (Linux) Browse to `/opt/alfresco/`, and run `alfresco.sh stop`.

Starting the Alfresco virtualization server

If you have installed Alfresco WCM, you can use the Website preview feature by starting the Alfresco virtualization server.

- (Windows)
    - Browse to `C:\alfresco`, and double-click `virtual.start.bat`, or
    - Click the **Start** menu, and select **All Programs > Alfresco Enterprise Edition > Start Virtual Server**.
- (Linux) Browse to `/opt/alfresco/` and run `virtual_alf.sh start`.

  The default shell for this script is `sh`. You can edit the `virtual_alf.sh` file to change to your preferred shell. For example, change the `#!/bin/sh` line to `#!/bin/bash`.

Stopping the Alfresco virtualization server

- (Windows)
    - Browse to `C:\alfresco`, and double-click `virtual.stop.bat`, or
    - Click the **Start** menu, and select **Programs >Alfresco Enterprise Edition > Stop Virtual Server**.
- (Linux) Browse to `/opt/alfresco/`, and run `sh virtual_alf.sh stop`.

Starting the deployment engine

The standalone deployment engine is implemented as a set of Java libraries and is multi-platform.

Bourne shell scripts are provided for UNIX and Windows batch files are provided for Windows.

- (Windows) To start the standalone deployment engine:
    - Open a command prompt, and run the `deploy_start` script, or
    - Select **Start Menu > All Programs > Alfresco Standalone Deployment Receiver > Start Alfresco Standalone Deployment Receiver**.

  The Start menu action is available if you have used the deployment installer to install the Standalone Deployment Engine. This action is calling the `deploy_start.bat` script.

  It is also possible to install the standalone deployment engine as a Windows service, which can automatically start when Windows starts.
- (Linux) To start the standalone deployment engine, open a command prompt and run the `deploy_start.sh` script.

The default shell for this script is `sh`. You can edit the `alfresco.sh` file to change to your preferred shell. For example, change the `#!/bin/sh` line to `#!/bin/bash`.

When deploying to a deployment engine running on a multi-NIC system, it may be necessary to bind the RMI registry to a particular IP address. To do this, add the following to the Java command in `deploy_start.sh` or `deploy_start.bat`:

```
-Djava.rmi.server.hostname=x.x.x.x
```

Where `x.x.x.x` is the IP address assigned to the NIC to which you want to bind.

Stopping the deployment engine

The standalone deployment engine is implemented as a set of Java libraries and is multi-platform.

Bourne shell scripts are provided for UNIX and Windows batch files are provided for Windows.

- (Windows) To stop the standalone deployment engine:
  - Open a command prompt, and run `deploy_stop.bat`, or
  - Select **Start Menu > All Programs > Alfresco Standalone Deployment Receiver > Stop Alfresco Standalone Deployment Receiver**.
- (Linux) To stop the standalone deployment engine, open a command prompt, and run the `deploy_stop.sh` script.

Configuring Alfresco

This chapter provides information on configuring Alfresco, including databases, subsystems, and the Alfresco core and extended services.

Configuration overview

Alfresco is preconfigured with a set of system configuration parameters.

The system property settings are found in the following files:

- `<configRoot>/alfresco/repository.properties`
- `<configRoot>/alfresco/hibernate-cfg.properties`
- `<configRoot>/alfresco/subsystems/<category>/<type>/*.properties`

Many of the system configuration parameters are completely exposed as properties, which you can extend or override.

There are two methods of extending the properties:

- Editing the global properties (`alfresco-global.properties`) file
- Using a JMX client, such as JConsole

The global properties file is used by Alfresco to detect the extended properties. For example, when you install Alfresco, many of the installation settings are saved in the global properties file. You can continue to use the global properties to do all your property extensions; however, whenever you make a change, you must restart the Alfresco server.

The JMX client allows you to edit the settings while the system is running. The settings you change are automatically persisted in the database and synchronized across a cluster. When you start up Alfresco, the system initially uses the `alfresco-global.properties` file to set the properties within the JMX client, but then any changes you make in the JMX client persist in the database but are not reflected back into the `alfresco-global.properties` file.

There are two types of property that you may need to edit:

Type 1: Properties specified directly in XML files

For example:

```
<bean id="wcm_deployment_receiver"
class="org.alfresco.repo.management.subsystems.ChildApplicationContextFactory"
        <parent="abstractPropertyBackedBean">
        <property name="autoStart">
                <value>true</value>
        </property>
</bean>
```

The value for the property `autoStart` is set to true directly in the `wcm-bootstrap-context.xml` file.

Type 2: Properties set by variables in XML files

For example:

```
<bean id="userInstallationURI" class="org.alfresco.util.OpenOfficeURI">
      <constructor-arg>
         <value>${ooo.user}</value>
      </constructor-arg>
   </bean>
```

The value for the property `constructor-arg` is replaced with a variable `${ooo.user}`.

When Alfresco starts up, type 1 properties are read from the XML file; type 2 properties get their values read from all the various property files. Then, the database is checked to see if there are any property values set there, and if any property has been changed, this value is used instead.

Some of the type 2 properties can be viewed and changed by the JMX console, some cannot. For example. `ooo.exe` can be viewed and changed using the JMX client; `index.recovery.mode` cannot be viewed or changed using the JMX client.

In a new Alfresco installation, none of these properties are stored in the database. If you set a property using the JMX interface, Alfresco stores the value of the property in the database. If you never use JMX to set the value of a property, you can continue using the `alfresco-global.properties` file to set the value of the property. Once you change the property setting using JMX, and it is therefore stored in the DB, you cannot use the properties files to change the value of that property.

For advanced configuration, you can also extend or override the Spring bean definitions that control Alfresco's Java classes. To do so, add or copy a Spring bean file named `*-context.xml` to the `<extension>` directory, or `<web-extension>` directory to extend Share. For examples of the Spring bean extensions, download the sample extension files.

Runtime administration with a JMX client

By default, you can reconfigure Alfresco by shutting down the server, editing the relevant property in the configuration files, and then restarting the server. However, there are some support operations that can be performed on-demand at runtime without needing to restart the server.

The Java Management Extension (JMX) interface allows you to access Alfresco through a standard JMX console that supports JMX Remoting (JSR-160). This lets you:

- Manage Alfresco subsystems
- Change log levels
- Enable or disable file servers (FTP/CIFS/NFS)
- Set server read-only mode
- Set server single-user mode
- Set server maximum user limit - including ability to prevent further logins
- Count user sessions/tickets
- User session/ticket invalidation

Example consoles include:

- JConsole (supplied with Java SE 5.0 and higher)
- MC4J
- JManage

Some of these consoles also provide basic graphs and/or alerts for monitoring JMX-managed attributes.

Connecting to Alfresco through JMX client

You can connect to Alfresco through a JMX client that supports JSR-160.

1. Open a JMX cilent that supports JMX Remoting (JSR-160).
2. Enter the JMX URL:

   ```
   service:jmx:rmi:///jndi/rmi://<hostname>:50500/alfresco/jmxrmi
   ```

Where `<hostname>` is the name of your host or IP address.

3. Enter the default JMX user name: `controlRole`
4. Enter the default JMX password: `change_asap`

   You must change the default JMX password as soon as possible.

5. Change the JMX password in the following files:
   - `<configRoot>/alfresco/alfresco-jmxrmi.access`
   - `<configRoot>/alfresco/alfresco-jmxrmi.password`

Disabling JMX

The JMX functionality is enabled using the `RMIRegistryFactoryBean` in the `core-services-context.xml` file.

1. Open the `<configRoot>\classes\alfresco\core-service-context.xml` file.
2. Comment out the `RMIRegistryFactoryBean` section.
3. Save the file.

Configuring Alfresco with JConsole

This section describes how to use the JMX client, JConsole for Alfresco runtime administration. JConsole is a JMX client available from Sun Microsystems Java SE Development Kit (JDK).

The initial configuration that displays in JConsole is set from the `alfresco-global.properties` file.

1. Open a command console.
2. Locate your JDK installation directory.

   For example, the JDK directory may be `java/bin`.
3. Enter the following command:

   ```
   jconsole
   ```

   The **JConsole New Connection** window displays.
4. Double-click on the Alfresco Java process.

   For Tomcat, this the Java process is usually labelled as **org.apache.catalina.startup.Bootstrap start**.

   JConsole connects to the managed bean (or MBean) server hosting the Alfresco subsystems.
5. Select the **MBeans** tab.

   The available managed beans display in JConsole.
6. Navigate to **Alfresco > Configuration**.

   The available Alfresco subsystems display in an expandable tree structure. When you select a subsystem, the **Attributes** and **Operations** display below it in the tree.
7. Select **Attributes** and set the required Alfresco subsystem attribute values.

   Values that can be edited are shown with blue text.

   When you change a configuration setting, the subsystem automatically stops.
8. Restart the Alfresco subsystem:
   a. Navigate to the subsystem.
   b. Select **Operations**.

   c. Click **Start**.
9. To stop the Alfresco subsystem without editing any properties:
   a. Navigate to the subsystem.
   b. Select **Operations**.
   c. Click **Stop**.
10. To revert back to all the previous edits of the Alfresco subsystem and restores the default settings:
    a. Navigate to the subsystem.
    b. Select **Operations**.
    c. Click **Revert**.
11. Click **Connection > Close**.

The settings that you change in a JMX client, like JConsole, are persisted in the Alfresco database. When you make a dynamic edit to a subsystem:

1. When a subsystem, that is currently running, is stopped, its resources are released and it stops actively listening for events. This action is like a sub-part of the server being brought down. This 'stop' event is broadcast across the cluster so that the subsystem is brought down simultaneously in all nodes.
2. The new value for the property is persisted to the Alfresco database.

There are two ways to trigger a subsystem to start:

- The start operation
- An event that requires the subsystem

Global properties file

The global properties `alfresco-global.properties` file contains the customizations for extending Alfresco.

If you install Alfresco using one of the installation wizards, the `alfresco-global.properties` file is modified with the settings that you chose during installation. If you install Alfresco using one of the Tomcat bundles or the WAR file, you can modify the `alfresco-global.properties` file manually.

A standard global properties file is supplied with the Alfresco installers and bundles. By default, the file contains sample settings for running Alfresco, for example, the location of the content and index data, the database connection properties, the location of third-party software, and database driver, and Hibernate properties.

Modifying global properties

For edits to the `alfresco-global.properties` file, when specifying paths for Windows systems, you must replace the Windows path separator characters with either the `\\` separator or the forward slash `/` Unix path separator. Also, when using folder names like `User Homes`, you must manually escape the space. For example, change the value to `User_x0020_Homes`.

1. Browse to the `<classpathRoot>` directory.

   For example, for Tomcat 6, browse to the `$TOMCAT_HOME/shared/classes/` directory.
2. Open the `alfresco-global.properties` file.

3. Add a root location for the storage of content binaries and index files in the `dir.root=` property.

   For example, `dir.root=C:/Alfresco/alf_data`.

4. Change the database connection properties.

Property	Description
`db.name=alfresco`	Specifies the name of the database.
`db.username=alfresco`	Specifies the name of the main Alfresco database user.
`db.password=alfresco`	Specifies the password for the Alfresco database user.
`db.host=localhost`	Specifies the host where the database is located.
`db.port=3306`	Specifies the port that the database uses.

5. Specify the location of the following external software:

Property	Desctiption
`ooo.exe=`	Specifies the location of the OpenOffice installation.
`ooo.user`	Specifies the user account for OpenOffice.
`jodconverter.officeHome=`	Specifies the location of the OpenOffice installation for JODConverter transformations. To use the JODConverter, uncomment the `ooo.enabled=false` and jodconverter.enabled=true properties.
`jodconverter.portNumbers=`	Specifies the port numbers used by each JODConverter processing thread. The number of process will match the number of ports.
`img.root=`	Specifies the location of the ImageMagick installation.
`swf.exe=`	Specifies the location of the SWF tools installation.

6. Uncomment the section for the database that you require.
7. Select a JDBC driver used with each connection type.
8. (Optional) Set the Hibernate dialect for your database.

   If you do not specify a dialect, Hibernate will attempt to detect the dialect of your database automatically from the JDBC connection.

9. Add your global custom configurations.
10. Save your file.

Setting composite properties in the global properties file

This section uses the the `imap.server.mountPoints` property as an example. The `ImapConfigMountPointsBean` class that holds the component beans has four properties of its own:

- `beanName`
- `store`

- `rootPath`
- `mode`

1. Open the `<classpathRoot>/alfresco-global.properties` file.
2. To set some overall defaults for all component instances, use the format:

   ```
   <property>.default.<component property>
   ```

   These values would show up, for example, when you added a new component instance but did not specify its properties.

   For example:

   ```
   imap.server.mountPoints.default.store=${spaces.store}
   imap.server.mountPoints.default.rootPath=/
   ${spaces.company_home.childname}
   imap.server.mountPoints.default.mode=virtual
   ```

   This example does not define a default for `beanName` because there is a way of populating it for each instance.
3. To set up the `imap.server.mountPoints` with a composite value, set the master composite property using a comma-separated list.

   For example:

   ```
   imap.server.mountPoints=Repository_virtual,Repository_archive
   ```

   This defines that the property contains two `ImapConfigMountPointsBean` instances, named `Repository_virtual` and `Repository_archive`. Because `ImapConfigMountPointsBean` implements the `BeanNameAware` Spring interface and has a `beanName` property, these instance names are automatically set as the bean names.
4. To define component properties specific to each component instance, use the format:

   ```
   <property>.value.<component instance name>.<component property>
   ```

   For example:

   ```
   imap.server.mountPoints.value.Repository_virtual.mode=virtual
   imap.server.mountPoints.value.Repository_archive.mode=archive
   ```

Java command line

All the Alfresco properties can be set using using the standard `alfresco-global.properties` configuration file.

There may be circumstances where it is more convenient to change properties on the fly. The Java command line provides an alternative method of setting the properties.

The most common use of the Java command line is in a multiple-machine environment where the basic, common customizations are set using standard properties and the machine-specific values are set using command line options. For example, an administrator is likely to configure all Alfresco installs to behave similarly by setting properties in the configuration files, but will use the Java command line to vary settings like the database connection, contentstore locations, and CIFS domain name.

You can set any property that is specified in the `alfresco-global.properties` file, including CIFS, JGroups, and Hibernate.

Setting properties on the Java command line

- Add a `-Dprop=value` to `JAVA_OPTS` or for anything that is sent to the Java command line.

  For example, `-Ddir.root=/alfresco/data -Ddb.url=xxxx`.

Upgrading configurations

This page describes the important information for upgrading to Alfresco Enterprise Edition Edition 3.3 SP1.

Alfresco Enterprise Edition 3.3 SP1 includes the concept of subsystems. Overall defaults for subsystem properties are set in the `alfresco-global.properties` file.

When you upgrade to Enterprise Edition Edition 3.3 SP1 from releases prior to Version 3.2, the recommendation is that you move all your repository and database configurations from the `<extension>custom-repository.properties` file to the `alfresco-global.properties` file.

For example, you should move the configuration settings for the following properties:

Sample custom content and index data location property:

- `dir.root=`

Sample database connection properties:

- `db.name=`
- `db.username=`
- `db.password=`
- `db.host=`
- `db.port=`

External locations for third-party products:

- `ooo.exe=soffice`
- `img.root=./ImageMagick`
- `swf.exe=./bin/pdf2swf`

Database connection properties:

- `db.driver=`
- `db.url=`
- `hibernate.dialect=`

Also, move any Hibernate settings from the `custom-hibernate-dialect.properties` file to the `alfresco-global.properties` file.

When you have moved your configurations, delete the `custom-repository.properties` file and the associated Spring context file `custom-repository-context.xml`, then restart the server for the settings to take effect.

If you continue to use the `custom-repository.properties` file to set your configurations, the settings may override those set in the `alfresco-global.properties` file requiring more complex ongoing administration and maintenance and possibly leading to unexpected results.

For more information on subsystems, refer to Alfresco subsystems on page 81.

If you currently have configuration using any of these services, it is recommend that you move or reconfigure them using the new `alfresco-global.properties` configuration. This new method simplifies the setup and maintenance of these systems and it also simplifies future upgrades.

Modifying Spring bean definition files

For advanced configuration, you can also extend or override the Spring bean definitions that control the Alfresco Java classes in the following directories:

- The `<extension>` directory contains the configuration files for extending Alfresco.
- The `<web-extension>` directory contains the configuration files for extending Alfresco Share.

For more information on the `<extension>` and `<web-extension>` paths, refer to System paths on page 10.

1. Browse to the `<extension>` directory. For example, for Tomcat 6:
   - (Windows) `C:\Alfresco\tomcat\shared\classes\alfresco\extension`
   - (Linux) `tomcat/shared/classes/alfresco/extension`

   Each file has a copy with a `.sample` extension.
2. Open the configuration file with the `.sample` extension.
3. Add your configurations to the file.
4. Save the file without the `.sample` extension.

Modifying system configuration files

The system configuration files that are read by Alfresco are contained in the `<configRoot>` and `<configRootShare>` directories. The preferred method of configuration is to extend the system files using the global properties file (`alfresco-global.properties`). If you choose to modify the system files directly, there is a risk that you will lose your changes when you next upgrade. To minimize the risk, use the following approach.

For more information on the `<configRoot>` and `<configRootShare>` paths, refer to System paths on page 10.

Before you start, back up all your configuration files for security.

1. Make a copy of the default file you want to modify, and rename it.
2. Make your customization. Make only one logical change at one time (one logical change may mean several physical changes).
3. Check any XML is well-formed. You can use any XML editor or you can open the file in a browser, such as Firefox.
4. Before making further changes, test each logical change by stopping and restarting Alfresco.
5. If you need to roll back the changes for troubleshooting, roll them back in the reverse sequence to which you applied them. Stop and restart Alfresco after each rollback.

Repository system configuration files

The Alfresco system configuration files are in the application WAR file. When the server starts, the files expand to `<configRoot>`. The path for `<configRoot>` is different depending on your application server. For example:

- Tomcat: `<TOMCAT_HOME>\webapps\alfresco\WEB-INF`
- JBoss: `<JBOSS_HOME>\server\default\tmp\deploy\tmp*alfresco-exp.war\WEB-INF`

The system configuration files are maintained by Alfresco and contained in `<configRoot>` and `<configRoot>\classes\alfresco`. The preferred method of configuring Alfresco is to extend the default files using the global properties file (`alfresco-global.properties`).

The following four files represent the core of the application configuration:

1. `<configRoot>\classes\alfresco\application-context.xml`

   This file is the starting point of the Spring configurations. This file only performs imports, including a wildcard import of all `classpath*:alfresco/extension/*-context.xml` files.
2. `<configRoot>\classes\alfresco\core-services-context.xml`

   Core Alfresco beans are defined here, including the importing of properties using the `repository-properties` bean.
3. `<configRoot>\classes\alfresco\repository.properties`

   This file is imported by the `repository-properties` bean. The file defines some of the core system properties, including:

   - `dir.root`

     This folder is where the binary content and Lucene indexes are stored. The `alf_data` folder is where they are stored by default, but you should change this to your own location. It is relative by default, but must point to a permanent, backed-up location for data storage.
   - `dir.auditcontentstore`

     This folder is where the audit's content store is stored.
   - `dir.indexes`

     This folder contains all Lucene indexes and deltas against those indexes.
   - `db.*`

     These are the default database connection properties.
   - `db.schema.update`

     This property controls whether the system bootstrap should create or upgrade the database schema automatically.
   - `hibernate.hbm2ddl.auto`

     This determines whether Hibernate updates the database schema or just validates it.

     - `update`: checks and updates the database schema as appropriate
     - `validate`: checks the database schema and fails if it is not compliant
4. `<configRoot>\classes\alfresco\domain\hibernate-cfg.properties`

   This file defines Hibernate default system properties. The file is imported by the `hibernateConfigProperties` bean, which is in `hibernate-context.xml`. The important Hibernate property is:

   - `hibernate.dialect`

     This alters the SQL dialect that Hibernate uses when issuing queries and updates to the database. Normally, this is the only Hibernate property that will need changing, depending on the database you are running against.

Customizing individual configuration items

The Alfresco configuration is implemented using three types of files:

- Properties files

- Configuration files
- Bean files

Customizing properties files

Properties files contain properties and end with the extension `.properties`. Each property is defined on one line. A `.properties` file has no header, so you do not need to take any action to preserve the header.

1. Open the file you want to customize.

   For example, open the `alfresco-global.properties` file.
2. Comment out all the properties you do not want to modify by adding the “#” character.

   For example, to override the `db.driver` property, you only require one line:

   ```
   db.driver=oracle.jdbc.OracleDriver
   ```
3. Uncomment all the properties that you want to activate by removing the “#” character.
4. Save the file.

Customizing a configuration file

Configuration files end with the extension `.xml`, and define `<config>` tags.

A typical configuration file is `<extension>/web-client-config-custom.xml`. A configuration file contains `<alfresco-config>` tags outside the `<config>` tags. You must preserve these tags in your customized file.

1. Open the configuration file that you want to customize.

   For example, the following code snippet is from the `<configRoot>/classes/alfresco/web-client-config.xml` file:

   ```
   <config evaluator="string-compare" condition="Languages">
      <languages>
         <language locale="en_US">English</language>
         <language locale="fr_FR">French</language>
      </languages>
   </config>
   ```
2. Delete each pair of `<config> </config>` tags that you do not want to modify.
3. To delete part of an item:

   a. Copy the original `<config>` statement.

   b. Delete the children you do not want.

   c. Use the replace=“true” attribute.

   For example, to delete French from the example in Step 1:

   ```
   <config evaluator="string-compare" condition="Languages" replace="true">
      <languages>
         <language locale="en_US">English</language>
      </languages>
   </config>
   ```
4. To add another item, such as another language, you only need a single `<language>` item. The other `<language>` items will remain enabled. For example, if you wanted Spanish in addition to English and French:

   ```
   <config evaluator="string-compare" condition="Languages">
     <languages>
         <language locale="es_ES">Spanish</language>
   ..</languages>
   ```

```
</config>
```

5. Customize the contents of the remaining `<config>` tags.
6. Save your customized file.

Configuration files

The configuration files contain configurations that either augment or replace the standard configuration. The system configuration files are located in:

`<configRoot>\classes\alfresco\web-client-config-*`

Replacing a configuration

To replace the configuration, add a `replace=“true”` attribute to the configuration element. For example: `<config evaluator="xx" condition=“yy” replace=“true”>`

Any configuration within a section marked this way completely replaces any configuration found in the Alfresco-maintained files.

For example, to replace the list of languages shown in the login page, add the following:

```
<config evaluator="string-compare" condition="Languages" replace="true">
 <languages>
    <language locale="fr_FR">French</language>
    <language locale="de_DE">German</language>
 </languages>
</config>
```

Modifying one property

The attribute `replace` completely replaces the configuration. To modify one property, you can add the changed piece.

For example, to add another language, you need a single `<language>` item. The other `<language>` items remain enabled. For example, if you want Spanish in addition to English and German:

```
<config evaluator="string-compare" condition="Languages">
 <languages>
   <language locale="es_ES">Spanish</language>
   ..
 </languages>
</config>
```

For example configurations, see Customizing Alfresco Explorer on page 137

Customizing a bean file

Bean files end with the extension `.xml` and contain `<bean>` tags. You can modify `<bean>` tags to define properties or point to customized files.

There are two common uses of beans:

- To define properties
- To point to one or more of your customized files

A typical bean file is `<extension>/custom-repository-context.xml`. A bean file contains `<?xml>` and `<!DOCTYPE>` headers, and `<beans>` tags outside the `<bean>` tags. You must preserve these items in your customized file.

When you override a `<bean>`, the entire effects of the original bean are lost. The effect is the same as if you had overridden a `<config>` by using `replace="true"`. Therefore, the overriding `<bean>` must contain any information from the default bean that you want to keep, as well as any additional information.

For example, if a core bean has four values, and you want to modify a single value, the resultant bean must still have four values. However, if you want to add a value, then the resultant bean must have five values - the original four values plus the added value.

1. Open the bean file that you want to customize.

   For example, the following `<bean>` is from the `<configRoot>/classes/alfresco/action-services-context.xml` file:

```
<bean id="mail"
 class="org.alfresco.repo.action.executer.MailActionExecuter"
 parent="action-executer">
   <property name="publicAction">
      <value>true</value> <!-- setting to true -->
   </property>
   <property name="mailService">
      <ref bean="mailService"></ref>
   </property>
</bean>
```

2. Delete each pair of `<bean> </bean>` tags that you do not want to modify.
3. Modify the contents of the remaining `<bean>` tags.

   For example, the following overrides the `publicAction` property from the previous example:

```
<bean id="mail"
 class="org.alfresco.repo.action.executer.MailActionExecuter"
 parent="action-executer">
   <property name="publicAction">
      <value>false</value> <!-- setting to false -->
   </property>
   <property name="mailService">
      <ref bean="mailService"></ref>
   </property>
</bean>
```

4. Save the file.

Configuring databases

This section describes how to configure supported databases for use with Alfresco.

Overriding database connection properties

This section describes how to override the database properties.

1. Open the `<classpathRoot>/alfresco-global.properties` file.

   Lines in these files are commented (begins with the “#” sign).
2. Remove the “#” sign from the beginning of each line for the database you want to activate.
3. Type a "#" at the beginning of each line for the database you want to deactivate.
4. Save the file.

Configuring a MySQL database

Setting up the MySQL database to work with Alfresco involves the following tasks:

1. Installing a MySQL database connector.

2. Creating the Alfresco database.
3. Configuring the MySQL database connection.

Some of the Alfresco installation wizards install an embedded instance of MySQL that is configured with the correct settings. If you prefer to use your own instance of MySQL, this section describes the configuration that you should use.

Installing a MySQL database connector

The MySQL database connector is required when installing Alfresco with MySQL. The database connector allows MySQL database to talk to the Alfresco server.

If you installed Alfresco using one of the installation wizards, the MySQL database connector is already in the correct directory.

1. Browse to the MySQL download site: http://dev.mysql.com/
2. Download **MySQL Connector/J x.x**, and extract the following file from the downloaded `.zip` or `.tar.gz` file:

   `mysql-connector-java-5.x.x-bin.jar`
3. Copy the JAR file into the `<TOMCAT_HOME>/lib` directory for Tomcat 6.

The MySQL connector is installed. The next step is to create the Alfresco MySQL database.

Creating the Alfresco database

This section describes how to set up a MySQL database to work with Alfresco.

If you installed Alfresco using one of the installation wizards, and chose the MySQL database, the Alfresco database is already created for you. To check that the database is created correctly, go to step 7.

1. Open a command prompt.
2. Type the following:
   - (Windows) `cd <alfresco install dir>\extras\databases\mysql`
   - (Linux) `cd <alfresco install dir>/alfresco/extras/databases/mysql`
3. Press ENTER.
4. Type the following:
   - (Windows) `db_setup.bat`
   - (Linux) `mysql –u root –p <db_setup.sql`
5. Press ENTER.
6. At the Enter Password prompt, type `admin` or the password you used to install MySQL, and press ENTER to prepare the database.
7. To verify the database, type `mysql -u root -p`.
8. At the mysql> prompt, type `show databases;`

   A list of databases in MySQL should display, including one named `alfresco`. If you receive an error message, add the `mysql` command to the Windows Path variable, and recreate the MySQL database.
9. Browse to the MySQL data directory, and verify the `Alfresco` folder exists.

   The MySQL database is created in the `Alfresco` folder.

You have now created the Alfresco database. To replace the default database configuration for the MySQL (or other) database, you must modify the values in two files.

Configuring the MySQL database connection

To replace the default database configuration, you must modify values in the `<classpathRoot>/alfresco-global.properties` file.

1. Open the `alfresco-global.properties` file.
2. Locate the following line:

   ```
   dir.root=./alf_data
   ```
3. Edit the line with an absolute path to point to the directory in which you want to store Alfresco data. For example: `dir.root=C:/Alfresco/alf_data`
4. Locate the following line:

   ```
   db.url=jdbc:mysql://${db.host}:${db.port}/${db.name}
   ```
5. Set the host, port number, and name to the location of your MySQL JDBC driver.

   ```
   db.url=jdbc:mysql://your-host:3306/alfresco
   ```
6. (Optional) Enable case sensitivity.

   The default, and ideal, database setting for Alfresco is to be case-insensitive. For example, the user name properties in the `<configRoot>\classes\alfresco\repository.properties` file are:

   ```
   # Are user names case sensitive?
   user.name.caseSensitive=false
   domain.name.caseSensitive=false
   domain.separator=
   ```

   If your preference is to set the database to be case-sensitive, add the following line to the `alfresco-global.properties` file:

   ```
   user.name.caseSensitive=true
   ```

   You also must ensure that the MySQL database is set to use UTF-8 and InnoDB. Refer to Configuring MySQL on page 14
7. Save the file.

You have now configured the MySQL database to work with Alfresco.

Configuring an Oracle database

To replace the default database configuration, you must modify values in the `<ClasspathRoot>/alfresco-global.properties` file.

This section describes how to configure an Oracle RDBMS database for use with Alfresco.

Some of the Alfresco installation wizards configure Oracle with the correct settings. If you prefer to install Alfresco manually, this section describes the configuration that you should use.

1. Create a new `alfresco` user and schema in Oracle.

   The `alfresco` user must have Connect and Resource privileges in Oracle.
2. Ensure the `alfresco` user has the required privileges to create and modify tables.

   This can be removed once the server has started, but may be required during upgrades.
3. Open the `<ClassPathRoot>\alfresco-global.properties` file.
4. Locate the following line:

   ```
   dir.root=./alf_data
   ```

5. Edit the line with an absolute path to point to the directory in which you want to store Alfresco data. For example: `dir.root=C:/Alfresco/alf_data`
6. Set the `host` and `port` number to the location of your Oracle JDBC driver.
7. Override the repository properties by removing the comments on each of the following lines:

   ```
   #oracle#db.driver=oracle.jdbc.OracleDriver
   #oracle#db.url=jdbc:oracle:thin:@${db.host}:${db.port}:${db.name}
   ```
8. (Optional) If you have multiple Alfresco instances installed on your Oracle server, add the following:

   `hibernate.default_schema=ALFRESCO`

   This forces the database metadata queries to target the schema that each database user is using.
9. Comment out the MySQL connection section.
10. Save the file.
11. Copy the Oracle `JDBC driver JAR` into `/lib`.
12. Start the Alfresco server.

    If you receive JDBC errors, ensure the location of the Oracle JDBC drivers are on the system path, or add them to the relevant `lib` directory of the application server. The Oracle JDBC drivers are located in the `<orainst>/ora<ver>/jdbc/lib` directory (for example, `c:\oracle\ora92\jdbc\lib`).

    The JDBC driver for Oracle is in the JAR file: `ojdbc14.jar`.

Configuring a SQL Server database

This section describes how to configure a Microsoft SQL Server database for use with Alfresco. To modify the default database configuration, you must edit values in the `<ClasspathRoot>\alfresco-global.properties` file.

Some of the Alfresco installation wizards configure SQL Server with the correct settings. If you prefer to install Alfresco manually, this section describes the configuration that you should use.

1. Create a new `alfresco` database and and user in SQL Server.

   Create the database with a UTF8 character set and the default collation settings.
2. Ensure the `alfresco` user has the required privileges to create and modify tables.

   This can be removed once the server has started, but may be required during upgrades.
3. Enable snapshot isolation mode with the following command:

   `ALTER DATABASE alfresco SET ALLOW_SNAPSHOT_ISOLATION ON;`
4. Ensure that the TCP connectivity is enabled on the fixed port number 1433.
5. Copy the jTDS JDBC driver to `$TOMCAT_HOME\lib`.

   This release requires the jTDS driver version 1.2.5 for compatibility with the SQL Server database.
6. Open the `alfresco-global.properties` file.
7. Locate the following line:

   `dir.root=./alf_data`
8. Edit the line with an absolute path to point to the directory in which you want to store Alfresco data. For example: `dir.root=C:/Alfresco/alf_data`.

9. Set the database connection properties.

   For example:

```
db.name=alfresco
db.username=alfresco
db.password=alfresco
db.host=localhost
db.port=1433
```

10. Locate the SQLServer connection section:

```
# SQLServer connection
# Requires jTDS driver version 1.2.5 and SNAPSHOT isolation mode
# Enable TCP protocol on fixed port db.port
# Prepare the database with:
# ALTER DATABASE db.name SET ALLOW_SNAPSHOT_ISOLATION ON;
#
#mssql#db.driver=net.sourceforge.jtds.jdbc.Driver
#mssql#db.url=jdbc:jtds:sqlserver://${db.host}:${db.port}/${db.name}
#mssql#db.txn.isolation=4096
```

11. Remove the `#mssql#` comment on the following lines:

```
#mssql#db.driver=net.sourceforge.jtds.jdbc.Driver
#mssql#db.url=jdbc:jtds:sqlserver://${db.host}:${db.port}/${db.name}
#mssql#db.txn.isolation=4096
```

12. Add the following line:

```
hibernate.default_schema=dbo
```

13. Comment out the MySQL connection section.
14. Save the file.
15. Start the Alfresco server.

Configuring a PostgreSQL database

This section describes how to configure a PostgreSQL database for use with Alfresco.

Some of the Alfresco installation wizards configure PostgreSQL with the correct settings. If you prefer to install Alfresco manually, this section describes the configuration that you should use.

1. Copy the appropriate PostgreSQL driver JAR to:
   - (Tomcat) `$TOMCAT_HOME/lib`
   - (JBoss) `/jboss/server/default/lib`
2. Create a database named `alfresco`.
3. Create a user named `alfresco`.
4. Set the new user's password to `alfresco`.
5. Ensure the Alfresco user has the required privileges to create and modify tables.
6. Open the `<classpathRoot>/alfresco-global.properties` file.
7. Locate the following line:

```
dir.root=./alf_data
```

8. Edit the line with an absolute path to point to the directory in which you want to store Alfresco data. For example: `dir.root=C:/Alfresco/alf_data`
9. Override the following properties:

```
db.driver=org.postgresql.Driver
db.name=alfresco
db.url=jdbc:postgresql://localhost/<database name>
db.username=alfresco
db.password=alfresco
hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
hibernate.query.substitutions=true TRUE, false FALSE
```

10. Comment out the MySQL database connection properties.
11. Save the file.
12. To allow password-authenticated connections through TCP/IP, ensure the PostgreSQL configuration file contains the following:

    ```
    host all all 127.0.0.1/32 password
    ```

    For PostgreSQL version 8.1.3, the file name is `pg_hba.conf`. For every other version, the file name is `postgresql.conf`.
13. Change the ownership of the directory tree specified in the `custom-data-location.properties` file to be owned by the user running the alfresco server. .

    The tree must be owned by the user running the Alfresco server.
14. Browse to the Alfresco Enterprise Edition download area, and download the following AMP file:

    `postresqlamp`
15. Apply the file using the Module Management Tool. Refer to Installing an Alfresco Module Package on page 36 for more information.
16. Restart the Alfresco server.

Configuring core services

This section describes how to configure Alfresco core services.

Tuning the JVM

The hardware requirements for the Alfresco repository, and Explorer and Share, are variable and depend on the number of concurrent users that access the system. You can tune the memory and garbage collection parameters for the JVM to be appropriate for your situation. This section suggests metrics and estimates, but your system may vary.

Concurrent users are users who are constantly accessing the system through Alfresco Explorer with only a small pause between requests (3-10 seconds maximum) with continuous access 24/7. Casual users are users occasionally accessing the system through Alfresco Explorer or WebDAV/CIFS interfaces with a large gap between requests (for example, occasional document access during the working day).

Disk space usage

The size of your Alfresco repository defines how much disk space you will need; it is a very simple calculation. Content in Alfresco is, by default, stored directly on the disk. Therefore, to hold 1000 documents of 1 MB will require 1000 MB of disk space. You should also make sure there is sufficient space overhead for temporary files and versions. Each version of a file (whether in DM or WCM) is stored on disk as a separate copy of that file, so make sure you allow for that in your disk size calculations (for DM, use versioning judiciously).

Use a server class machine with SCSI Raid disk array. The performance of reading/writing content is almost solely dependent on the speed of your network and the speed of your disk array. The overhead of the Alfresco server itself for reading content is very low as content is streamed directly from the disks to the output stream. The overhead of writing content is also low but depending on the indexing options (for example, atomic or background indexing), there may be some additional overhead as the content is indexed or metadata is extracted from the content in each file.

JVM memory and CPU hardware for multiple users

The repository L2 Cache with initial VM overhead and basic Alfresco system memory is set up with a default installation to require approximately 512 MB (maximum).

This means you can run the Alfresco repository and Explorer with many users accessing the system with a basic single CPU server and only 512 MB of memory assigned to the Alfresco JVM. However, you must add additional memory as your user base grows, and add CPUs depending on the complexity of the tasks you expect your users to perform, and how many concurrent users are accessing the client.

For these metrics, **N** concurrent users is considered equivalent to **10xN** casual users that the server could support.

Number of users	Recommended memory / CPU settings per server
For 50 concurrent or up to 500 casual users	1 GB JVM RAM 2x server CPU (or 1 x Dual-core)
For 100 concurrent users or up to 1000 casual users	1 GB JVM RAM 4x server CPU (or 2 x Dual-core)
For 200 concurrent users or up to 2000 casual users	2 GB JVM RAM 8x server CPU (or 4 x Dual-core)

Concurrent users are considered to be users constantly accessing the system through the Alfresco web-client with only a small pause between requests (3-10 seconds maximum) - with continuous access 24/7. Casual users are considered to be users occasionally accessing the system through the Alfresco web-client or webdav/CIFS interfaces with a large gap between requests (for example, occasional document access during the working day).

Permanent Generation (PermGen) Size

The default PermGen size in Sun JVMs is 64M, which is very close to the total size of permanent objects (Spring beans, caches, and so on) that Alfresco creates. For this reason it is easy to overflow the PermGen as the result of configuration changes or with the addition of custom extensions. It is recommended that you increase the PermGen to avoid OutOfMemory errors, for example, -XX:MaxPermSize=128M is a good starting point.

The size of the PermGen is now increased in the Alfresco startup scripts, so provided you are using those scripts, no changes should be necessary.

Maximum JVM Heap Size 32/64bit

An important calculation to keep in mind is:

```
(Managed Heap + native heap + (thread stack size * number of threads)) cannot
exceed 2GB on 32bit x86 Windows or Linux systems
```

This is a limitation of the Sun Java VM. It means that even if you install 4GB of RAM into your server, a single instance of the JVM cannot grow beyond 2GB on a 32bit server machine.

A 64 bit OS/JVM has much bigger values. It is recommended that a 64bit OS with large memory hardware (>2GB assigned to the JVM) is used for deployments of >250 concurrent or >2500 casual users.

You can also set up your machine to cluster if you prefer to solve multi-user access performance issues with additional machines rather than a single powerful server.

JVM settings

Alfresco generates a high proportion of temporary objects, both in client threads as well as in the background processes. To reduce the number of temporary objects that spill into the OldGen portion of the heap, you need to set the NewSize option as large as possible.

The following settings reduce the garbage collections and reveal (with GC printing and JMX tracing) that the OldGen was not growing noticeably over and above the permanent space allocated for caches. Cache sizes are still estimated top out around 520M. So, for a typical 32 bit installation with at least 2GB available for the VM, you can use the following settings:

```
JAVA_OPTS=
-server
-Xss1024K
-Xms1G
-Xmx2G
-XX:MaxPermSize=128M
-XX:NewSize=512m
```

If you wish to have Hotspot precompile the classes, you can add the following options. This will, however, significantly increase the server startup time, but will highlight missing dependencies that may occur later.

```
-Xcomp
-Xbatch
```

The following can also be adjusted to control the garbage collection behaviour:

```
-XX:+UseConcMarkSweepGC
-XX:+CMSIncrementalMode
-XX:CMSInitiatingOccupancyFraction=80
```

Low end machines

This section applies if you have less than 2GB available.

The stack size of 1024K (-Xss1024K) is generous. Some installations may require a little over 512K on occasion. Many may only use 256K. If the per-thread memory consumption is too high for your installation, reduce the stack size to 512K and then to 256K and note any memory-related errors in the logs.

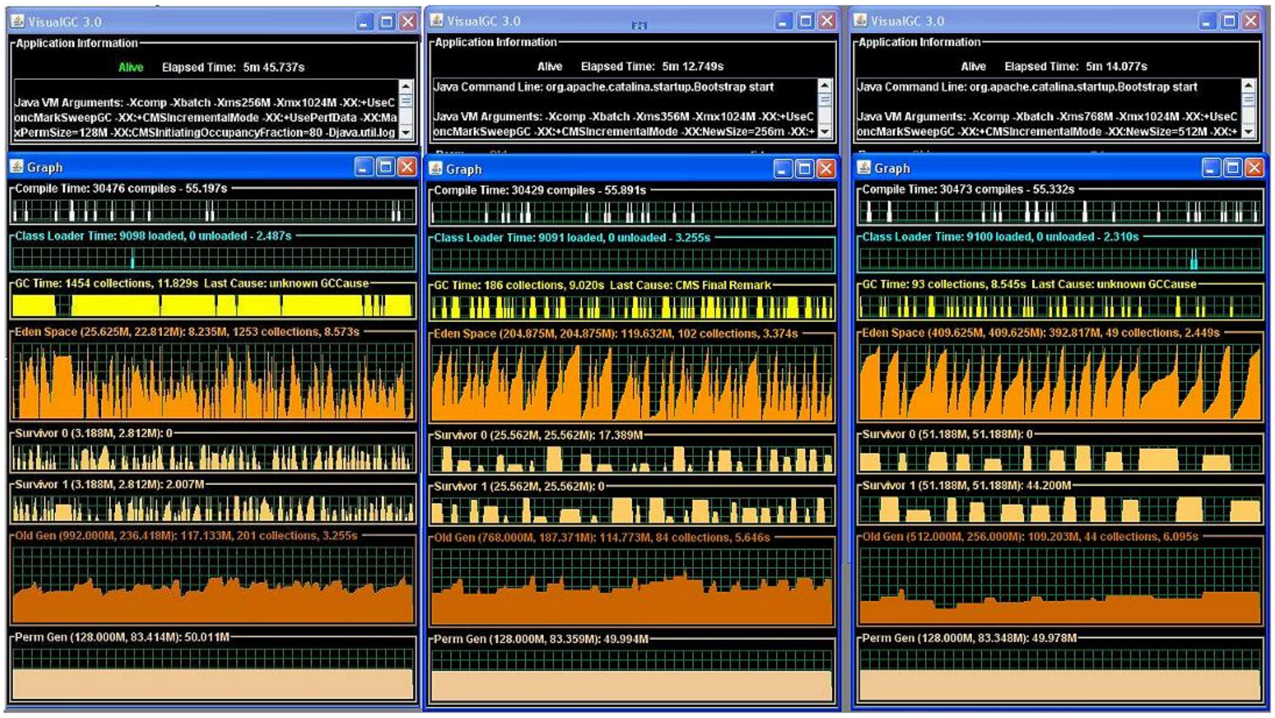
The `NewSize` should be kept as large as possible. It can be reduced, but the memory consumption should be watched on a monitoring tool, for example, JConsole, to ensure that the rate of spillover of temporary objects is kept down. If the machine is supporting 500 simultaneous operations, for instance, then the spillover of temporary objects (from `NewSize` being too small) will cause hold-ups on memory assignment as the garbage collector does sweeps.

Effects of NewSize

Given that the OldGen is composed primarily of cache data of up to about 520M, at least 1GB should be reserved for OldGen. Once -Xmx increases, the OldGen can be increased to 2G. 512M should be left as a buffer to account for miscellaneous (PermGen, and so on). So the following variations might be applied:

```
-Xmx2G -Xms1G -XX:NewSIze=512M (OldGen at least 1G)
-Xmx3G -Xms1G -XX:NewSize=512M (OldGen at least 2G)
-Xmx4G -Xms2G -XX:NewSize=1G (OldGen at least 2.5G)
-Xmx6G -Xms3G -XX:NewSize=2G (OldGen at least 3.5G)
-Xmx8G -Xms4G -XX:NewSize=3G (OldGen at least 4.5G)
```

If you need these levels, you will need to run JConsole (and Java 6) to observe the rate of spillover from Eden space to Survivor to OldGen. If, after the system has been running for a while, the OldGen size stabilizes, then the NewSize can be increased appropriately. The following diagram (using VisualGC) shows how varying the NewSize value affects overall garbage collection activity:



```
Alive    Elapsed Time:  5m 45.737s
Java VM Arguments: -Xcomp -Xbatch -Xms256M -Xmx1024M -XX:+UseC
oncMarkSweepGC -XX:+CMSIncrementalMode -XX:+UsePerfData -XX:Ma
xPermSize=128M -XX:CMSInitiatingOccupancyFraction=80 -Djava.util.log
Compile Time: 30476 compiles - 55.197s
Class Loader Time: 9098 loaded, 0 unloaded - 2.487s
GC Time: 1454 collections, 11.829s  Last Cause: unknown GCCause
Eden Space (25.625M, 22.812M): 8.235M, 1253 collections, 8.573s
Survivor 0 (3.188M, 2.812M): 0
Survivor 1 (3.188M, 2.812M): 2.007M
Old Gen (992.000M, 236.418M): 117.133M, 201 collections, 3.255s
Perm Gen (128.000M, 83.414M): 50.011M
```

```
Alive    Elapsed Time:  5m 12.749s
Java Command Line: org.apache.catalina.startup.Bootstrap start
Java VM Arguments: -Xcomp -Xbatch -Xms356M -Xmx1024M -XX:+UseC
oncMarkSweepGC -XX:+CMSIncrementalMode -XX:NewSize=256m -XX:+
Compile Time: 30429 compiles - 55.891s
Class Loader Time: 9091 loaded, 0 unloaded - 3.255s
GC Time: 186 collections, 9.020s  Last Cause: CMS Final Remark
Eden Space (204.875M, 204.875M): 119.632M, 102 collections, 3.374s
Survivor 0 (25.562M, 25.562M): 17.389M
Survivor 1 (25.562M, 25.562M): 0
Old Gen (768.000M, 187.371M): 114.773M, 84 collections, 5.646s
Perm Gen (128.000M, 83.359M): 49.994M
```

```
Alive    Elapsed Time:  5m 14.077s
Java Command Line: org.apache.catalina.startup.Bootstrap start
Java VM Arguments: -Xcomp -Xbatch -Xms768M -Xmx1024M -XX:+UseC
oncMarkSweepGC -XX:+CMSIncrementalMode -XX:NewSize=512M -XX:+
Compile Time: 30473 compiles - 55.332s
Class Loader Time: 9100 loaded, 0 unloaded - 2.310s
GC Time: 93 collections, 8.545s  Last Cause: unknown GCCause
Eden Space (409.625M, 409.625M): 392.817M, 49 collections, 2.449s
Survivor 0 (51.188M, 51.188M): 0
Survivor 1 (51.188M, 51.188M): 44.200M
Old Gen (512.000M, 256.000M): 109.203M, 44 collections, 6.095s
Perm Gen (128.000M, 83.348M): 49.978M
```

Command line configuration

The beans that load the `alfresco-global.properties` will give preferential treatment to any JVM-set properties.

Setting properties on the JVM

- (Windows) At a command prompt, enter the following:

  ```
  Set JAVA_OPTS=-Ddir.root=e:/alfresco/data
  ```

- (Linux) At a command prompt, enter the following:

  ```
  export JAVA_OPTS==Ddir.root=/srv/alfresco/data
  ```

Mixing global properties and system property settings

You can use a combination of global properties and system properties for certain customizations. For example, if you wish to distribute a system that has a core set of properties overridden but need to customize the last few for each installation.

1. Activate the properties in the `<classpathRoot>/alfresco-global.properties` file.
2. Set all common defaults for your system.
3. On each installation, add the final configuration values. For example:

   ```
   -Ddb.username=alfresco
   -Ddb.password=alfresco
   -Dhibernate.dialect=org.alfresco.repo.domain.hibernate.dialect.
   AlfrescoOracle9Dialect
   -Dhibernate.default_schema=ALFRESCO_DEV
   -Dindex.tracking.cronExpression='0/5 * * * * ?'
   -Dindex.recovery.mode=AUTO
   -Dalfresco.cluster.name=ALFRESCO_DEV
   ```

Configuring the repository cache

The Alfresco repository uses *Ehcache* in-memory caches. These caches are transaction safe and can be clustered. Caches greatly improve repository performance but they use Java heap memory.

Cache settings depend on the amount of memory available to your Alfresco server. The default `ehcache-default.xml` file is enough for most systems and is currently set for 512MB of cache heap memory. This is the recommended default for a Java heap size of 1GB.

For information on Ehcache, refer to: http://ehcache.sourceforge.net/

Individual cache settings

All cache settings are in: `<configRoot>\alfresco\ehcache-default.xml`.

Do not directly modify this file.

Each cache is configured in an XML block similar to the following:

```
<!-- approx 50MB memory required -->
<cache name="org.alfresco.repo.domain.hibernate.NodeImpl.childAssocs"
 maxElementsInMemory="25000"
eternal="true" timeToIdleSeconds="0" timeToLiveSeconds="0"
 overflowToDisk="false" />
```

The comment shows the approximate amount of Java heap memory required for the specific example. Some object references are shared by the caches, so the amount of memory used is not as high as the approximate value may suggest. It is best to err on the side of caution. Cache tracing can show which caches fill quickly for your specific server usage pattern.

`name`
The `name` attribute is the name of the cache and generally indicates the type of objects being cached.

`maxElementsInMemory`
The `maxElementsInMemory` controls the maximum size of the cache. This value can be changed to tune the size of the cache for your system. Ehcache caches are implemented using a linked-map system, which means that memory is only required for objects that are actually in memory. If you set the `maxElementsInMemory` to a high value, it will not automatically allocate that number of slots. Instead, they are added to the linked list as required. When `maxElementsInMemory` is reached, the cache discards the oldest objects before adding new objects.

`timeToIdleSeconds` - `timeToLiveSeconds`
`timeToIdleSeconds` and `timeToLiveSeconds` control the automatic timeout of cached objects.

`overflowToDisk`
`overflowToDisk` controls whether the cache should overflow to disk rather than discarding old objects.

A typical trace is as follows:

The criteria are:

- (`MissCount - CurrentCount`) must be as low as possible.
- (`HitCount/MissCount`) must be as high as possible.

`Estimated maximum size` affects the permanent memory taken up by the cache. If the caches grow too large, they may crowd out transient session memory and slow down the system. It is useful to have this running, on occasion, to identify the caches with a low `HitCount/MissCount` ratio.

Tracing the caches

This task describes how to trace the repository caches.

1. To output detailed Ehcache usage information, set the following logging category to `DEBUG`:

   `org.alfresco.repo.cache.EhCacheTracerJob`
2. Browse to the file `<configRoot>\alfresco\scheduled-jobs-context.xml`.
3. Override the following bean to change the trigger schedule:

   `ehCacheTracerJob`
4. Override the following property to activate the trigger:

   `schedulerFactory`

   When triggered, the job will collect detailed cache usage statistics and output them to the `log/console`, depending on how logging has been configured for the server.

Configuring extended services

This section describes how to configure extended services in the Alfresco server.

Adding a MIME type

To add a MIME type, you create two new files in the `<extension>` directory:

1. A file containing the added or modified MIME type
2. A file containing a `<bean>` that points to your new MIME type file

In both cases, the new files are created by copying default files and modifying them.

1. Browse to the directory: `<configRoot>\alfresco\mimetype\`
2. Open the following files:
   a. `mimetype-map.xml`
   b. `mimetype-map-openoffice.xml`
3. Refer the contents of both files to view the MIME types.
4. Copy the pair of `<mimetype> </mimetype>` tags that is most similar to what you want to add.
5. Paste these tags in the `<extension>\mimetypes-extension.xml.sample` file.
6. Modify the inserted MIME type to match your requirements.
7. Save the file without the `.sample` extension.
8. Ensure you have created and added the new file to the `<extension>` directory so that you can point to your new file.
9. To point to your new file:
   a. Browse to the file: `mimetype-map-extension-context.xml.sample`

      This file contains the following line, which points to your modified file: `<value>classpath:alfresco/extension/mimetypes-extension.xml</value>`
   b. Save the file without the `.sample` extension.

Configuring metadata extraction

Metadata extractors offer server-side extraction of values from added or updated content. Definitions of the default set of extractors are in `<configRoot>/alfresco/content-services-context.xml`.

1. Declare a new extractor in `<extension>/custom-repository-context.xml`.

   The following example shows a new extractor written in `class com.company.MyExtracter`:

   ```
   <bean id="com.company.MyExtracter" class="com.company.MyExtracter"
   parent="baseMetadataExtracter" />
   ```

Versioning

Versioning allows you to track content history. By default, content that is created in the repository is not versionable. When creating content, users must specify *versionable* on a case-by-case basis.

When content is versionable, the version history is started. The first version of the content is the content that exists at the time of versioning. If you want all content to be versionable at the instant of creation, you can modify the definition of that content type in the data dictionary. The definition must include the mandatory aspect *versionable*.

By default, all versionable content has auto-version *on*. As a result, when content is updated, the version number is updated. The auto-version capability can be turned off on a content-by-content basis in the user interface. If you want auto-versioning to be *off* for all content, you can modify the definition of that content type in the data dictionary.

Making all content versionable

1. Open the data dictionary `<configRoot>\alfresco\model\contentModel.xml`.
2. Search for the `<type>`: `<type name="cm:content">`
3. Immediately after the closing `</properties>` tag, insert the following lines:

   ```
   <type name="cm:content">
      <properties>
      ...
      </properties>
      <mandatory-aspects>
         <aspect>cm:versionable</aspect>
      </mandatory-aspects>
   </type>
   ```
4. Save the file.
5. Restart the Alfresco server.

Disabling the auto-versioning feature

1. Open the data dictionary file: `<configRoot>\alfresco\model\contentModel.xml`
2. Search for the following `<aspect>`: `<aspect name="cm:versionable">`
3. Change the boolean default to `false`, as follows:

   ```
   <property name="cm:autoVersion">
      <title>Auto Version</title>
      <type>d:boolean</type>
      <default>false</default>
   </property>
   ```
4. Save the file.
5. Restart the Alfresco server.

Setting up replication

Replication allows you to continuously copy a database to a different server.

To enable replication you set one server (the slave) to take all its updates from the other server (the master). During replication, no *data* is actually copied. It is the SQL *statements* that manipulate the data that is copied.

All statements that change the master database are stored in the master's binary logs. The slave reads these logs and repeats the statements on its own database. The databases will not necessarily be exactly synchronized. Even with identical hardware, if the database is actually in use, the slave will always be behind the master. The amount by which the slave is behind the master depends on factors such as network bandwidth and geographic location. The other server can be on the same computer or on a different computer. The effect of replication is to allow you to have a nearly current standby server.

Using more than one server allows you to share the read load. You can use two slaves. If one of the three servers fails, you can use one server for service while another server can copy to the failed server. The slaves need not be running continuously. When they are restarted, they catch up. With one or more slaves you can stop the slave server to use a traditional backup method on its data files.

Each slave uses as much space as the master (unless you choose not to replicate some tables) and must do as much write work as the master does to keep up with the write rate. Do not be without at least one slave or comparable solution if high reliability matters to you.

Replication is not another form of back up. You must do normal backups as well as replication. If a user mistypes a `DELETE` statement on the master, the deletion is faithfully reproduced on the slave.

Setting up MySQL replication

1. Open a MySQL command prompt on the master server.
2. Grant the slave permission to replicate:

   ```
   GRANT REPLICATION SLAVE ON *.* TO <slave_user> IDENTIFIED BY
    '<slave_password>'
   ```
3. If the master is not using the `binary update` log, add the following lines to `my.cnf` (Linux) or `my.ini` (Windows) configuration file on the master, and restart the server:

   ```
   [mysqld]
   log-bin
   server-id=1
   ```

   By convention, server-id for the master is usually `server-id 1`, and any slaves from 2 onwards, although you can change this. If the master is already using the binary update log, either note the offset at the moment of the backup (the next step), or use the `RESET MASTER` statement to clear all binary logs and immediately begin the backup. You may want to make a copy of the binary logs before doing this if you need to use the binary logs to restore from backup.
4. Make a backup of the database.

   This will be used to start the slave server. You can skip this step if you use the `LOAD DATA FROM MASTER` statement, but first review the following comments about locking the master.
5. Add the following to the configuration file on the slave:

   ```
   master-host=master-hostname
   master-user=slave-user
   master-password=slave-password
   server-id=2
   ```

   The slave user and slave password are those to which you set when you granted `REPLICATION SLAVE` permission on the master. The `server-id` must be a unique number, different to the master or any other slaves in the system. There are also two other options: `master-port`, used if the master is running on a non-standard port (3306 is default), and `master-connect-retry`, a time in seconds for the slave to attempt to reconnect if the master goes down. The default is 60 seconds.

Restore the data from the master, either as you would normally restore a backup or with the statement `LOAD DATA FROM MASTER`. The latter will lock the master for the duration of the operation, which could be quite lengthy, so you may not be able to spare the downtime.

Configuring the connection pool

This task describes how to to override the connection pool.

1. Open the `<extension>\custom-connection-pool-context.xml.sample` file.

   You can also set the basic pool information in `alfresco-global.properties`.
2. Set the connection pool properties. For example:
   ```
   db.pool.initial=10
   db.pool.max=100
   ```
3. Remove the `.sample` extension from this file.
4. Modify the properties where appropriate, paying particular attention to the `validationQuery` property. This property is an SQL query that validates connections from this pool before returning them to the caller. This query must returns at least one row.

   For explanations of each property shown in the file, refer to: http://jakarta.apache.org/commons/dbcp/configuration.html

Customizing content transformations

This task describes how to customize content transformations.

1. Copy the following file:

   `<configRoot>/alfresco/content-services-context.xml`
2. Paste this file into the `<extension>` directory or in your preferred directory.
3. Open the file. Transformers start below the comment:
   ```
   <!-- Content Transformations -->
   ```
4. Locate the bean containing a transformer that is most similar to the transformer that you want to add. (It is unlikely that you would want to *modify* an existing transformer.)
5. Delete every pair of `<bean> </bean>` tags except the pair containing the similar transformer.
6. Rename and modify the bean.
7. Save the file with a meaningful name.

   If you save the file in the `<extension>` directory, the filename must end with `#context.xml`.

Setting the open file handling

In UNIX environments, the `Too many open files` error, in the Alfresco log files, is often associated with Lucene. The issue occurs where the system is configured by default to allow users a maximum of only 1024 open file handles, which is often not sufficient for the file structure used by the indexes.

Although Lucene 2.0 suffers less from this problem, errors may still occur on systems during periods of heavy use and therefore it is recommended that on production Alfresco systems, the normal file handle limit is increased.

On Linux, the global setting for the maximum number of file handles is usually set in the `/proc/sys/fs/file-max` file.

1. Run the following command:

   ```
   ulimit -n
   ```

   Run this command as the `alfresco` user.

   The output of this command returns how many file handles the current user can have open at any one time. This is about 4096.
2. Check that the `pam` configuration. In `/etc/pam.d/system-auth`, you see:

   ```
   session     required      /lib/security/$ISA/pam_limits.so
   session     required      /lib/security/$ISA/pam_unix.so
   ```
3. Edit `/etc/security/limits.conf`.
4. Add the following:

   ```
   alfresco soft nofile 4096
   alfresco hard nofile 65536
   ```

   This sets the normal number of file handles available to the alfresco user to be 4096 - the soft limit.
5. If soft limit is too few, assign more file handles, up to the hard limit, using the following:

   ```
   ulimit -n 8192
   ```

Alfresco subsystems

An Alfresco subsystem is a configurable module responsible for a sub-part of Alfresco functionality. Typically, a subsystem wraps an optional functional area, such as IMAP bindings, or one with several alternative implementations, such as authentication.

A subsystem can be thought of as miniature Alfresco server embedded within the main server. A subsystem can be started, stopped, and configured independently, and it has its own isolated Spring application context and configuration.

The application context is a child of the main context, therefore, it can reference all the beans in the main application context. However, the subsystem's beans cannot be seen by the main application context and communication with the subsystem must be through explicitly imported interfaces. The main features of subsystems are:

Multiple ‘instances’ of the same type
The same template spring configuration can be initialized with different parameters in different instances. This enables, for example, the chaining of multiple authentication subsystems through property file edits.

Dynamic existence
JMX-based server configuration capabilities in Alfresco Enterprise releases.

Own bean namespace
There is no problem of bean name uniqueness if you need multiple instances of the same subsystem. Again, this vastly simplifies the problem of building an authentication chain as there is no need to edit any template Spring configuration.

Clearly defined interfaces with the rest of the system
A subsystem's interfaces must be 'imported' in order to be used anywhere else in the system. This is done by mounting them as dynamic proxies.

Hidden implementation specifics
Implementation specifics are not visible because all of its beans are hidden in a private container.

Swapping of alternative implementations
To switch over from native Alfresco authentication to NTLM passthru authentication, you switch to a passthru authentication subsystem and the correct components are swapped in.

Separate product from configuration
A subsystem binds its configuration settings to properties and can even do this for composite data. There is no longer a need to edit or extend prepackaged Spring configuration in order to configure a subsystem for your own needs.

Subsystem categories

Every subsystem has a category and a type.

- Category is a broad description of the subsystem's function, for example, Authentication.
- Type is a name for the particular flavor of implementation, where multiple alternative implementations exist, for example, `ldap`. Where a subsystem has only one implementation, you can use the default type name of `default`.

The Alfresco-supplied subsystem categories are:

Audit
Handles the audit related functions.

Authentication
Handles all authentication related functions, including:

- Password-based authentication
- Single Sign-on (SSO) for WebClient, WebDAV, Web Scripts and SharePoint Protocol
- CIFS and FTP authentication
- User registry export (LDAP only)

The subsystem is chained so that multiple instances of different types can be configured and used together.

OOoDirect
Handles the settings for OpenOffice transformations. With this subsystem, the Alfresco server directly manages OpenOffice.

OOoJodconverter
Handles the JODConverter settings for OpenOffice transformations. With this subsystem, the JODConverter manages OpenOffice, including a pool of separate OpenOffice processes, automatic restart of crashed OpenOffice processes, automatic termination of slow OpenOffice operations, automatic restart of any OpenOffice process after a number of operations.

Synchronization
Performs regular synchronization of local user and group information with the user registry exporters (usually LDAP directories) in the authentication chain.

fileServers
Handles the properties for the CIFS, FTP, and NFS servers.

email
Handles the outbound and inbound SMTP property settings.

imap
Handles the properties for the IMAP service.

sysAdmin
Handles the properties for server administration.

thirdparty
Handles the properties for SWF Tools and ImageMagick content transformers.

wcm_deployment_receiver
Handles the properties for WCM Deployment Receiver.

Subsystem configuration files

The prepackaged subsystem configuration files form part of the core product and should not be edited.

The prepackaged subsystems are found in the `<configRoot>/classes/alfresco/subsystems` directory.

Each subsystem directory should contain one or more Spring XML bean definition metadata files, with names matching the `*-context.xml` pattern. These files are loaded by the child application context that belongs to the subsystem instance.

The XML bean definitions may contain place holders for properties that correspond to configuration parameters of the subsystem. As per standard Spring conventions, these place holders begin with `${` and end with `}`. In the following example, the value of the `ooo.user` configuration parameter will be substituted into the bean definition when it is loaded:

```
<bean id="userInstallationURI" class="org.alfresco.util.OpenOfficeURI">
      <constructor-arg>
         <value>${ooo.user}</value>
      </constructor-arg>
   </bean>
```

There is no need to declare a `PropertyPlaceholderConfigurer` bean. An appropriate one is added into the application context automatically.

Subsystem properties

A subsystem declares default values for all the properties it requires in one or more `.properties` files in its subsystem directory.

For example, there could be a `mysubsystem.properties` file, containing the following:

```
ooo.user=${dir.root}/oouser
```

Place holders are used for system-wide properties, such as `dir.root` in the `-context.xml` and `.properties` files, as the child application context will recursively expand place holders for its own properties and all the place holders recognized by its parent.

Properties files in the subsystem directory declare the configuration parameters and provide default values where these have not been supplied elsewhere. These files should not be edited in order to configure the subsystem.

Use the following methods to modify the subsystem properties:

- Subsystems and all their composite properties show under the `Alfresco:Type=Configuration` tree in JConsole.
- See Modifying global properties on page 59 for more information on how to configure a prepackaged subsystem.
- `-D` options

Mounting a subsystem

A subsystem can be mounted, that is, its existence can be declared to the main server. To mount a susbsystem, use the `ChildApplicationContextFactory` bean. This is an object that wraps the Spring application context that owns the subsystem and its beans. It initializes its application context as a child of the main Alfresco context with an appropriate `PropertyPlaceholderConfigurer` that will expand its configuration parameters.

Any instances that you define should extend the `abstractPropertyBackedBean` definition. The identifier that you define for the bean automatically becomes the subsystem category and defines where the factory will look for configuration files, in the search paths.

1. Open the core `bootstrap-context.xml file` (the file that controls the startup of beans and their order).
2. Locate the following bean definition:

```
<!--  Third party transformer Subsystem -->
 <bean id="thirdparty"
 class="org.alfresco.repo.management.subsystems.ChildApplicationContextFactory"

 parent="abstractPropertyBackedBean">
     <property name="autoStart">
         <value>true</value>
     </property>
 </bean>
```

   The `autoStart` property is set to true, meaning that the child application context will be refreshed when the server boots up, activating the beans it contains. For subsystems containing background processes or daemons (for example, the file server subsystem), it is very important to set this property, otherwise the subsystem will never activate.
3. Save your file.

Mounting a subsystem with composite properties

A subsystem is limited to flat property sets for its configuration, therefore it is difficult to allow structured data in this configuration. A composite property is a special property whose value is a list of beans.

- For example, the IMAP subsystem is mounted as:

```
<!-- IMAP Subsystem -->
 <bean id="imap"
 class="org.alfresco.repo.management.subsystems.ChildApplicationContextFactory"

 parent="abstractPropertyBackedBean">
     <property name="autoStart">
         <value>true</value>
     </property>
     <property name="compositePropertyTypes">
         <map>
             <entry key="imap.server.mountPoints">

 <value>org.alfresco.repo.imap.config.ImapConfigMountPointsBean</value>
             </entry>
         </map>
     </property>
 </bean>
```

  The subsystem declares a single composite property called `imap.server.mountPoints` with component type `org.alfresco.repo.imap.config.ImapConfigMountPointsBean`.
- The configured value of this composite property is materialized in the child application context as a `ListFactoryBean`. The bean's ID should match the name of the composite property. So, for example, in the IMAP subsystem configuration:

```
 <!--The configurable list of mount points - actually a post-processed
 composite property! -->
    <bean id="imap.server.mountPoints"
 class="org.springframework.beans.factory.config.ListFactoryBean">
        <property name="sourceList">
```

```
            <list>
                <!-- Anything declared in here will actually be ignored
and replaced by the configured composite propery value, resolved on
initialization -->
                <bean id="Repository_virtual"
class="org.alfresco.repo.imap.config.ImapConfigMountPointsBean">
                    <property name="mode">
                        <value>virtual</value>
                    </property>
                    <property name="store">
                        <value>${spaces.store}</value>
                    </property>
                    <property name="path">
                        <value>/${spaces.company_home.childname}</value>
                    </property>
                </bean>
                <bean id="Repository_archive"
class="org.alfresco.repo.imap.config.ImapConfigMountPointsBean">
                    <property name="mode">
                        <value>archive</value>
                    </property>
                    <property name="store">
                        <value>${spaces.store}</value>
                    </property>
                    <property name="path">
                        <value>/${spaces.company_home.childname}</value>
                    </property>
                </bean>
            </list>
        </property>
    </bean>
```

Other beans in the subsystem application context can use `imap.server.mountPoints` as though it were a regular list of `ImapConfigMountPointsBeans`.

Extension classpath

The `alfresco-global.properties` file can only be used to define properties that are global to the whole system. You can also control the properties of subsystems that may have multiple instances, for example, the Authentication subsystems. To do this, you need to target different values for the same properties, to each subsystem instance. You can use the extension classpath mechanism.

1. Add a property file to your application server's global classpath.

   For example, under `$TOMCAT_HOME/shared/classes`.

2. Create the path to match the following pattern to override specific properties of a subsystem instance:

   ```
   alfresco/extension/subsystems/<category>/<type>/<id>/*.properties
   ```

   The `<id>` is the subsystem instance identifier, which will be default for single instance subsystems, or the provided identifier for chained subsystems.

For example, if your authentication chain looked like this:

```
authentication.chain=alfrescoNtlm1:alfrescoNtlm,ldap1:ldap
```

Then you could put property overrides for alfrescoNtlm1 in the following file:

```
alfresco/extension/subsystems/Authentication/alfrescoNtlm/alfrescoNtlm1/
mychanges.properties
```

The default type and ID of non-chained subsystems is default, so you could put overrides for file server properties in the following file:

```
alfresco/extension/subsystems/fileServers/default/default/mychanges.properties
```

Configuring authentication

Authentication is one of the categories of the Alfresco subsystem. An authentication subsystem is a coordinated stack of compatible components responsible for providing authentication-related functionality to Alfresco.

Alfresco offers multiple alternative implementations of authentication subsystem, each engineered to work with one of the different types of back-end authentication server that you may have available in your enterprise.

The main benefits of the authentication subsystem are:

- Subsystems for all supported authentication types are pre-wired and there is no need to edit template configuration.
- There is no danger of compatibility issues between sub-components, as these have all been pre-selected. For example, your CIFS authenticator and authentication filter are guaranteed to be compatible with your authentication component.
- Common parameters are shared and specified in a single place. There is no need to specify the same parameters to different components in multiple configuration files.
- There is no need to edit the `web.xml` file. The `web.xml` file uses generic filters that call into the authentication subsystem. The `alfresco.war` file is a portable unit of deployment.
- You can swap from one type of authentication to another by activating a different authentication subsystem.
- Your authentication configuration will remain standard and, therefore, more manageable to support.
- Authentication subsystems are easily chained

Authentication subsystem types

The following table shows the authentication subsystem types supplied with Alfresco and the optional features they support.

Type	Description	Single sign-on (SSO) support	CIFS authentication	User registry entry
`alfrescoNtlm`	Native Alfresco authentication	Yes, NTLM	Yes	No
`ldap`	LDAP protocol authentication (For example, OpenLDAP)	No	No	Yes
`ldap-ad`	Authentication and user registry export from Active Directoy via the LDAP protocol	No	No	Yes
`passthru`	Windows domain server authentication	Yes, NTLM	Yes	No
`kerberos`	Kerberos Realm authentication	Yes, SPNEGO	Yes	No
`external`	Authentication using an external SSO mechanism	Yes	No§	No

If you configure a single authentication subsystem of a type that does not support CIFS authentication (for example, LDAP), then the CIFS server will be automatically disabled. If you want CIFS and LDAP, then you must set up an authentication chain.

Authentication subsystem components

The main components of an authentication subsystem are:

authentication component
Handles the specifics of talking to the back-end authentication system.

authentication Data Access Object (DAO)
Decides what user management functions are allowed, if any. For example, the ability to create a user.

authentication service
Wraps the authentication component and DAO with higher-level functions.

user registry export service (optional)
Allows Alfresco to obtain user attributes, such as email address, organization, and groups automatically.

authentication filters
Provide form or SSO-based login functions for the following:

- web client
- WebDAV
- Web scripts
- SharePoint Protocol

file server authenticators
Provide authentication functions for the following:

- CIFS protocol (optional)
- FTP protocol

Authentication chain

At least one of the authentication subsystem types will allow you to integrate Alfresco with the authentication servers in your environment. However, if integrating Alfresco with only one of these systems is not sufficient, you may want to combine multiple authentication protocols against a collection of servers.

For this reason, Alfresco has a built-in authentication chain. The authentication chain is a priority-ordered list of authentication subsystem instances. Alfresco composes together the functions of the subsystems in this list into a more powerful conglomerate.

Authentication chain functions

The functions of the chain are composed in two different ways: chained functions and pass-through functions.

Chained functions

Chained functions combine together functions of more than one subsystem.

For example, when a user logs in, Alfresco tries to match the user's credentials against each of the subsystems in the chain in order.

- If a chain member accepts the credentials, the login succeeds

- If no chain member accepts, the login fails

User registry export is also chained. During a synchronize operation, users and groups are exported from each member of the chain supporting user registry export (that is, those of type LDAP) and imported into Alfresco. Ordering in the chain is used to resolve conflicts between users and groups existing in the same directory.

Pass-through functions

Pass-through functions cannot be chained and instead pass through to a single member of the chain, which handles them directly.

Examples of pass-through functions are:

- NTLM / SPNEGO - based Single Sign-On (SSO)
- CIFS Authentication

Such pass-through functions are handled by the first member of the chain that supports that function and has it enabled.

This means that only a subset of your user base may be able to use SSO and CIFS.

Default authentication chain

The default product configuration has a simple chain with one member. This is an instance of the `alfrescoNtlm` subsystem type with and ID of `alfrescoNtlm1`.

This is expressed in the built-in defaults in the `repository.properties` file as:

```
authentication.chain=alfrescoNtlm1:alfrescoNtlm
```

You can configure the properties of `alfrescoNtlm1` using the global properties file.

This subsystem instance does not have SSO enabled, by default.

To switch from password-based login to NTLM-based SSO, set the following property in the `alfresco-global.properties` file.

```
ntlm.authentication.sso.enabled=true
```

This basic use of NTLM requires Alfresco to store its own copies of your MD4 password hash, which means your user ID and password must be the same in both Alfresco and your Windows domain.

For direct authentication with a Windows domain server, without the need to synchronize accounts in Alfresco and the domain, use the pass-through (`passthru`) subsystem type.

Configuring the authentication chain

This section describes how you can add to or completely replace the default authentication chain.

The chain is controlled by the `authentication.chain` global property.

1. Open the `alfresco-global.properties` file.
2. Locate the `authentication.chain` global property.

   This is a comma separated list of the form:

   ```
   instance_name1:type1,...,instance_namen:typen
   ```
3. Set the property to the required values.

For example, set the property to the following value:

```
alfrescoNtlm1:alfrescoNtlm,ldap1:ldap
```

When you navigate to the `Alfresco:Type=Configuration,Category=Authentication,id1=manager` MBean in global property overrides, then a new authentication subsystem instance called `ldap1` will be brought into existence and added to the end of the authentication chain.

4. Save the file.

The following example integrates Alfresco with Active Directory has the requirements:

- Built-in Alfresco users and Windows users should be able to log in, with Alfresco taking precedence
- The Windows domain server should handle CIFS authentication directly
- LDAP should be used to synchronize user and group details

To achieve these requirements, configure the following authentication chain:

```
alfrescoNtlm1:alfrescoNtlm,passthru1:passthru,ldap1:ldap
```

Next, deactivate SSO in order to activate chained password-based login, target CIFS at `passthru1` and target synchronization (but not authentication) at `ldap1` by setting the properties as follows:

alfrescoNtlm1

```
ntlm.authentication.sso.enabled=false
alfresco.authentication.authenticateCIFS=false
```

passthru1

```
ntlm.authentication.sso.enabled=false
passthru.authentication.authenticateCIFS=true
```

ldap1

```
ldap.authentication.active=false
ldap.synchronization.active=true
```

Authentication chain example with JConsole

This section describes an example walk through of setting up an authentication chain using the JMX client, JConsole.

The first time you access a vanilla Alfresco installation through Alfresco Explorer, you see a Guest home page. The login is identified as the user guest and it is unauthenticated with limited access to the Alfresco functionality.

When you login as a user with administrator privileges, this user has access to the Administration Console, which allows you to create additional users and assign passwords. By default, users and passwords are managed from within Alfresco. Unauthenticated users, like guest, still have limited access.

The default authentication within Alfresco is adequate for small-scale environments, however, you may prefer to choose an authentication method that will scale up to a production environment. For example, you may wish to:

- Disable the unauthenticated guest user access
- Enable automatic sign-on using operating system credentials or a single sign-on (SSO) server, which removes the need for a login page

- Delegate authentication responsibility to a central directory server, which removes the need to set up users manually in the Administration Console

Alfresco authentication chain

Alfresco authentication and identity management functionality is provided by a prioritized list, or chain, of configurable subsystems.

An authentication subsystem provides the following functionality to Alfresco:

- Password-based authentication for Web browsing, Sharepoint, FTP, and WebDAV
- CIFS and NFS file system authentication
- Web browser and Sharepoint Single Sign on (SSO)
- User register export (the automatic population of the Alfresco user and authority database)

Several alternative authentication subsystems exist for the most commonly used authentication protocols. These subsystem enable you to tie Alfresco to some of the most widely used authentication infrastructures. Including more than one of these subsystems in the chain can accommodate complex authentication scenarios.

Using the alfrescoNtlm subsystem with JConsole

By default, Alfresco is preconfigured with a single subsystem in its authentication chain. This section describes the authentication chain from within the JMX client, JConsole.

1. Open a command console.
2. Locate your JDK installation directory.

   For example, the JDK directory may be `java/bin`.
3. Enter the following command:

   ```
   jconsole
   ```

   The **JConsole New Connection** window displays.
4. Double-click on the Alfresco Java process.

   For Tomcat, this the Java process is usually labelled as **org.apache.catalina.startup.Bootstrap start**.

   JConsole connects to the managed bean (or MBean) server hosting the Alfresco subsystems.
5. Select the **MBeans** tab.

   The available managed beans display in JConsole.
6. Navigate to **Alfresco > Configuration > Authentication**.

   An object called `manager` (JMX object name is `Alfresco:Type=Configuration, Category=Authentication, id1=manager`) displays, which represents the chain.
7. Navigate to **Alfresco > Configuration > Authentication > managed**.

   View the authentication chain members within managed.
8. Navigate to **alfrescoNtlm1 > Attributes**.

   The configuration settings of the only member of the authentication chain, `alfrescoNtlm1`, display.

   The subsystem special `$type` attribute indicates that it is of type `alfrescoNtlm`, which is the subsystem type that deals with Alfresco authentication.

The subsystem name `alfrescoNtlm1` is the same as its type but with a number appended. This is a common convention used for subsystem naming but it does not need to be followed as a rule. The subsystem could just as easily have been called `InternalAlfresco` and have functioned exactly the same.

As implied by the type name, subsystems of type `alfrescoNtlm` are capable of automatic sign-on to internal Alfresco accounts using the NTLM protocol. However, NTLM need not be used at all and is disabled by default.

Disabling the Guest user login page

This section describes how to set the authentication configuration in JConsole to disable the unauthenticated Guest user login.

1. Run JConsole.
2. In the **Attributes** panel, select the **Value** column next to **alfresco.authentication.allowGuestLogin**.
3. Change the value to **false**.

   This authentication change will be remembered by Alfresco, even if you restart the server. When running Alfresco in a clustered configuration, this edit will be mirrored immediately across all nodes in the cluster.
4. Check that the new value for the property is persisted to the Alfresco database by checking the output from the shell running the Alfresco server, or `alfresco.log` in the directory where Alfresco was started from. You see the lines:

   ```
   17:30:03,033 User:System INFO
    [management.subsystems.ChildApplicationContextFactory] Stopping
   'Authentication' subsystem, ID: [Authentication, managed, alfrescoNtlm1]
   17:30:03,064 User:System INFO
    [management.subsystems.ChildApplicationContextFactory] Stopped
   'Authentication' subsystem, ID: [Authentication, managed, alfrescoNtlm1]
   ```

   The subsystem is not started automatically after the edit, because, in a more complex scenario, you might want to reconfigure a number of attributes before trying them out on the live server. Only once the subsystem starts up again and reads its properties will the new settings take effect.
5. Log out from Alfresco Explorer in the browser.

   After logging out, immediately the log in screen appears and whenever you access Alfresco, you are always directed to the login page, not the guest access.

Removing the login page

This section describes how to set the authentication in JConsole not display the login page.

1. Login to Alfresco Explorer using the Administration user.
2. Click **Admin Console**.
3. Create a users whose user name and password matches those of your operating system account.
4. Run JConsole.
5. Navigate to **Alfresco > Configuration > Authentication > managed > alfrescoNtlm1 > Attributes**.
6. In the **Attributes** panel, select the **Value** column next to **ntlm.authentication.sso.enabled** attribute.
7. Change the value to **true**.
8. Navigate to **Alfresco > Configuration > Authentication > managed > alfrescoNtlm1 > Operations**.
9. Click the **Start** operation.

10. Close and restart your browser and try accessing Alfresco.

    If your browser supports NTLM and its security settings allow, it will automatically log you in using your operating system account name.

Authentication using a central directory server

This section describes how to delegate authentication responsibility to a centralized directory server. Most organizations maintain their user database in a directory server supporting the LDAP protocol, such as Active Directory or OpenLDAP. When integrated with an LDAP server, Alfresco can delegate both the password checking and account setup to the LDAP server. This exposes Alfresco to your entire enterprise, and avoids the need for an administrator to manually set up user accounts, and the need to store passwords outside of the directory server.

To integrate Alfresco with a directory server, include the `ldap` or `ldap-ad` subsystem in the authentication chain. Both subsystems offer exactly the same capabilities and should work with virtually any directory server supporting the LDAP protocol. The only difference is the default values configured for their attributes: ldap is preconfigured with defaults appropriate for OpenLDAP; ldap-ad is preconfigured with defaults appropriate for Active Directory.

This example uses an Active Directory server and therefore configures in an instance of the `ldap-ad` subsystem. Insert an ldap-ad instance into the Alfresco authentication chain. There are two choices:

- Replace the authentication chain

  Remove `alfrescoNtlm1` and add an instance of `ldap-ad`. This hands over all authentication responsibility to Active Directory and means that the default accounts such as “admin” and “guest” can not be used. It is important to configure at least one user who exists in Active Directory as an administrator and enable the guest account in Active Directory, if guest access is required. As `ldap-ad` is not capable of supporting CIFS authentication (due to it requiring exchange of an MD5 password hash), it rules out use of the CIFS server for all users and the CIFS server would be disabled.
- Add to the authentication chain

  Supplement the existing capabilities of `alfrescoNtlm1` by inserting an `ldap-ad` instance before or after `alfrescoNtlm1` in the chain. This means that the default accounts can be used alongside those accounts in the directory server. The default accounts can also access Alfresco through the CIFS server, as `alfrescoNtlm` is able to drive CIFS authentication. The position of the `ldap-ad` instance in the chain will determine how overlaps or collisions between user accounts are resolved. If an “admin” account exists in both Alfresco and Active Directory, then the order for defining for who admin is would be Alfresco, if `alfrescoNtlm1` came first, or Active Directory, if the `ldap-ad` instance came first.

1. Open JConsole.
2. Navigate to **Alfresco > Configuration > Authentication > manager > Attributes**.
3. Edit the chain attribute so that its value is:

   ```
   alfrescoNtlm1:alfrescoNtlm,ldap1:ldap-ad
   ```

Configuring alfrescoNtlm

`alfrescoNtlm` is the subsystem configured by default in the Alfresco authentication chain. It performs authentication based on user and password information stored in the Alfresco repository. It is capable of supporting both form-based login and NTLM-based Single Sign-On (SSO), as well as providing authentication for the CIFS server.

The NTLM SSO functions are disabled by default, which means there are no assumptions about the availability of a Windows domain. You can activate SSO with a single property, without any changes to the `web.xml` file or further file server configuration.

NTLM

The alfrescoNtlm subsystem supports optional NTLM Single Sign-On (SSO) functions for WebDAV and the Alfresco Explorer client.

NTLM v2 is supported, which is more secure that the NTLM v1. If the client does not support NTLMv2, it will automatically downgrade to NTLMv1.

By using NTLM authentication to access Alfresco Explorer and Alfresco WebDAV sites, the web browser can automatically log in.

When SSO is enabled, Internet Explorer will use your Windows log in credentials when requested by the web server. Firefox and Mozilla also support the use of NTLM but you need to add the URI to the Alfresco site that you want to access to `network.automatic-ntlm-auth.trusted-uris` option (available through writing `about:config` in the URL field) to allow the browser to use your current credentials for login purposes.

The Opera web browser does not currently support NTLM authentication, the browser is detected and will be sent to the usual Alfresco logon page.

In this configuration, Alfresco must still store its own copy of your MD4 password hash. In order to remove this need and authenticate directly with a Windows domain controller, consider using the pass-through subsystem.

alfrescoNtlm configuration properties

The alfrescoNtlm subsystem supports the following properties.

ntlm.authentication.sso.enabled
A Boolean that when true enables NTLM based Single Sign On (SSO) functionality in the Web clients. When false and no other members of the authentication chain support SSO, password-based login will be used.

ntlm.authentication.mapUnknownUserToGuest
Specifies whether unknown users are automatically logged on as the Alfresco guest user during Single Sign-On (SSO).

alfresco.authentication.authenticateCIFS
A Boolean that when true enables Alfresco-internal authentication for the CIFS server. When false and no other members of the authentication chain support CIFS authentication, the CIFS server will be disabled.

alfresco.authentication.allowGuestLogin
Specifies whether to allow guest access to Alfresco.

If you add extra administrator users in the `authority-services-context.xml` file and are using alfrescoNtlm, the extra users (other than the admin user) will no longer have administrator rights until you add them to the `ALFRESCO_ADMINISTRATORS` group using the Administration Console.

Alfresco Share SSO using NTLM

The Alfresco Share application exists as a separate web application to the main Alfresco repository/Explorer WAR file. It can run in the same application server instance on the same machine as the main Alfresco web application, or it can run on a completely separate application server instance on a different machine.

The Share application uses HTTP(S) to communicate with the configured Alfresco repository. To use NTLM with Share:

1. Enable SSO for the `alfrescoNtlm` or `passthru` subsystem.
2. Edit the Share `web.xml` file in the `WEB-INF` folder.
3. Change the `servlet` filter by enabling the following `servlet` filter:

```
  <filter>
     <filter-name>Authentication Filter</filter-name>
     <filter-
class>org.alfresco.web.site.servlet.NTLMAuthenticationFilter</filter-
class>
     <init-param>
        <param-name>endpoint</param-name>
        <param-value>alfresco</param-value>
     </init-param>
  </filter>
```

4. Add the following filter mappings:

```
  <filter-mapping>
     <filter-name>Authentication Filter</filter-name>
     <url-pattern>/page/*</url-pattern>
  </filter-mapping>

  <filter-mapping>
     <filter-name>Authentication Filter</filter-name>
     <url-pattern>/p/*</url-pattern>
  </filter-mapping>

  <filter-mapping>
     <filter-name>Authentication Filter</filter-name>
     <url-pattern>/s/*</url-pattern>
  </filter-mapping>
```

   The NTLM settings should already be in the `web.xml` file in a commented out section.

5. Change the configuration of the Share application. To use NTLM with Share, locate the following `.sample` configuration override file:

   `<web-extension>\share-config-custom.xml.sample`

   Copy and rename the file to:

   `<web-extension>\share-config-custom.xml`

6. Edit the file and uncomment the following section:

```
 <!--
        NTLM authentication config for Share
        NOTE: you will also need to enable the NTLM authentication filter
 in Share web.xml
              change localhost:8080 below to appropriate alfresco server
 location if required
   -->
   <config evaluator="string-compare" condition="Remote">
      <remote>
         <connector>
            <id>alfrescoCookie</id>
            <name>Alfresco Connector</name>
            <description>Connects to an Alfresco instance using cookie-
based authentication</description>

 <class>org.springframework.extensions.webscripts.connector.AlfrescoConnector</
class>
         </connector>

         <endpoint>
```

```
            <id>alfresco</id>
            <name>Alfresco - user access</name>
            <description>Access to Alfresco Repository WebScripts that
 require user authentication</description>
            <connector-id>alfrescoCookie</connector-id>
            <endpoint-url>http://localhost:8080/alfresco/wcs</endpoint-
url>
            <identity>user</identity>
            <external-auth>true</external-auth>
         </endpoint>
      </remote>
   </config>
```

7. Change the `endpoint-url` value to point to your Alfresco server location.

   Note the additional `<external-auth>true</external-auth>` parameter.
8. Restart the Share.

If you have configured `alfrescoNtlm` or `passthru` in your Alfresco authentication chain and enabled SSO, NTLM will be the active authentication mechanism.

Configuring LDAP

An LDAP subsystem supports two main functions:

- user authentication - checking a user's ID and password using an LDAP bind operation
- user registry export - exposing information about users and groups to the synchronization subsystem

Either of these functions can be used in isolation or in combination. When LDAP authentication is used without user registry export, default Alfresco person objects are created automatically for all those users who successfully log in. However, they will not be populated with attributes without user registry export enabled. LDAP user registry export is most likely to be used without LDAP authentication when chained with other authentication subsystems. For example, Kerberos against Active Directory, pass-through against ActiveDirectory, and possibly Samba on top of OpenLDAP.

The user registry export function assumes that groups are stored in LDAP as an object that has a repeating attribute, which defines the distinguished names of other groups, or users. This is supported in the standard LDAP schema using the `groupOfNames` type. See the example LDIF file.

For further information, see OpenLDAP tips on page 203 and Active Directory tips on page 206.

LDAP configuration properties

Both the `ldap` and `ldap-ad` subsystem types support the following configurable properties.

Defaults for `ldap` are typical for Open LDAP and defaults for `ldap-ad` are typical for Active Directory.

ldap.authentication.active
This Boolean flag, when true enables use of this LDAP subsystem for authentication. It may be that this subsytem should only be used for user registry export, in which case this flag should be set to false and you would have to chain an additional subsystem such as passthru or kerberos to provide authentication functions.

ldap.authentication.java.naming.security.authentication
The mechanism to use to authenticate with the LDAP server. Should be one of the standard values documented here or one of the values supported by the LDAP provider. Sun's LDAP provider supports the SASL mechanisms documented here. Recommended values are:

simple
the basic LDAP authentication mechanism requiring the username and password to be passed over the wire unencrypted. You may be able to add SSL for secure access, otherwise this should only be used for testing.

DIGEST-MD5
More secure RFC 2831 Digest Authentication. Note that with Active Directory, this requires your user accounts to be set up with reversible encryption, not the default setting.

ldap.authentication.userNameFormat
How to map the user id entered by the user to that passed through to LDAP. The substring `%s` in this value will be replaced with the entered user ID to produce the ID passed to LDAP. The recommended value of this setting depends on your LDAP server.

Active Directory
There are two alternatives:

User Principal Name (UPN)
These are generally in the format of <sAMAccountName>@<UPN Suffix>. Official documentation can be found here. If you are unsure of the correct suffix to use, use an LDAP browser, such as Softerra, to browse to a user account and find its userPrincipalName attribute. For example:

```
%s@domain
```

DN
This requires the user to authenticate with part of their DN, so may require use of their common name (CN) rather than their login ID. It also may not work with structured directory layouts containing multipe organization units (OUs). For example:

```
cn=%s,ou=xyz,dc=domain
```

OpenLDAP
The format used depends on the value chosen for `ldap.authentication.java.naming.security.authentication`.

simple
This must be a DN and would be something like the following:

```
uid=%s,ou=People,dc=company,dc=com
```

DIGEST-MD5
Use this value to pass through the entered value as-is:

```
%s
```

ldap.authentication.allowGuestLogin
Identifies whether to allow unauthenticated users to log in to Alfresco as the 'guest' user.

ldap.authentication.java.naming.factory.initial
The LDAP context factory to use. There is no need to change this unless you do not want to use the default Sun LDAP context factory.

ldap.authentication.java.naming.provider.url
The URL to connect to the LDAP server, containing its name and port. The standard ports for LDAP are 389 (and 636 for SSL). For example: `ldap://openldap.domain.com:389`

ldap.authentication.escapeCommasInBind
Escape commas in the entered user ID when authenticating with the LDAP server? Useful when using simple authentication and the CN is part of the DN and contains commas.

ldap.authentication.escapeCommasInUid
Escape commas in the entered user ID when deriving an Alfresco internal user ID? Useful when using simple authentication and the CN is part of the DN and contains commas, and the escaped \, is pulled in as part of a synchronize operation. If this option is set to true it will break the default home folder provider as space names cannot contain \.

ldap.authentication.defaultAdministratorUserNames
A comma separated list of user names who should be considered administrators by default.

ldap.synchronization.active
This flag enables use of the LDAP subsystem for user registry export functions and decides whether the subsystem will contribute data to the synchronization subsystem. It may be that this subsystem should only be used for authentication, in which case this flag should be set to false.

ldap.synchronization.java.naming.security.principal
The LDAP user to connect as to do the export operation. Should be in the same format as `ldap.authentication.userNameFormat` but with a real user ID instead of `%s`.

ldap.synchronization.java.naming.security.credentials
The password for this user, if required.

ldap.synchronization.queryBatchSize
If set to a positive integer, this property indicates that RFC 2696 paged results should be used to split query results into batches of the specified size. This overcomes any size limits imposed by the LDAP server. The default value of 1000 matches the default result limitation imposed by Active Directory. If set to zero or less, paged results will not be used.

ldap.synchronization.groupQuery
The query to select all objects that represent the groups to export. This query is used in full synchronization mode, which by default is scheduled every 24 hours.

ldap.synchronization.groupDifferentialQuery
The query to select objects that represent the groups to export that have changed since a certain time. Should use the placeholder `{0}` in place of a timestamp in the format specified by `ldap.synchronization.timestampFormat`. The timestamp substituted will be the maximum value of the attribute named by `ldap.synchronization.modifyTimestampAttributeName` the last time groups were queried. This query is used in differential synchronization mode, which by default is triggered whenever a user is successfully authenticated that does not yet exist in Alfresco.

ldap.synchronization.personQuery
The query to select all objects that represent the users to export. This query is used in full synchronization mode, which by default is scheduled every 24 hours.

ldap.synchronization.personDifferentialQuery
The query to select objects that represent the users to export that have changed since a certain time. Should use the placeholder `{0}` in place of a timestamp in the format specified by `ldap.synchronization.timestampFormat`. The timestamp substituted will be the maximum value of the attribute named by `ldap.synchronization.modifyTimestampAttributeName` the last time users were queried. This query is used in differential synchronization mode, which by default is triggered whenever a user is successfully authenticated that does not yet exist in Alfresco.

ldap.synchronization.groupSearchBase
The DN below which to run the group queries.

ldap.synchronization.userSearchBase
The DN below which to run the user queries.

ldap.synchronization.modifyTimestampAttributeName
The name of the operational attribute recording the last update time for a group or user.

ldap.synchronization.timestampFormat
The timestamp format. This varies between directory servers.

Active Directory
`yyyyMMddHHmmss'.0Z'`

OpenLDAP
`yyyyMMddHHmmss'Z'`

ldap.synchronization.userIdAttributeName
The attribute name on people objects found in LDAP to use as the uid in Alfresco.

ldap.synchronization.userFirstNameAttributeName
The attribute on person objects in LDAP to map to the first name property in Alfresco.

ldap.synchronization.userLastNameAttributeName
The attribute on person objects in LDAP to map to the last name property in Alfresco.

ldap.synchronization.userEmailAttributeName
The attribute on person objects in LDAP to map to the email property in Alfresco.

ldap.synchronization.userOrganizationalIdAttributeName
The attribute on person objects in LDAP to map to the email property in Alfresco.

ldap.synchronization.defaultHomeFolderProvider
The default home folder provider to use for people created via LDAP import.

ldap.synchronization.groupIdAttributeName
The attribute on LDAP group objects to map to the group name in Alfresco.

ldap.synchronization.groupType
The group type in LDAP.

ldap.synchronization.personType
The person type in LDAP

ldap.synchronization.groupMemberAttributeName
The attribute in LDAP on group objects that defines the DN for its members.

Checking the supported SASL authentication mechanisms

1. Using an LDAP browser, such as the one from Softerra, check the values of the `supportedSASLMechanisms` attributes on the root node of your LDAP server.

   The simple authentication method will not be reported because it is not a SASL mechanism.

2. If you use OpenLDAP, you can also query using `ldapsearch`. For example:

```
ldapsearch -h localhost -p 389 -x -b "" -s base -LLL
 supportedSASLMechanisms
dn:
supportedSASLMechanisms: DIGEST-MD5
supportedSASLMechanisms: NTLM
supportedSASLMechanisms: CRAM-MD5
```

Configuring pass-through

The pass-through (`passthru`) subsystem can be used to replace the standard Alfresco user database with a Windows server/domain controller, or list of servers, to authenticate users accessing Alfresco. This saves having to create user accounts within Alfresco.

The subsystem also supports optional NTLM Single Sign-On (SSO) functions for WebDav and the Alfresco Explorer and Share clients and direct CIFS authentication for the CIFS server. This method of authentication is much more secure than simple LDAP-based authentication or form-based authentication.

Only NTLM v1 is supported in this configuration. As NTLMv2 has been designed to avoid "man-in-the-middle" attacks, it would be impossible to use in this pass through style.

Pass-through configuration properties

The `passthru` subsystem supports domain level properties.

Also relevant are the configuration steps described in Alfresco Share SSO using NTLM if you want to enable NTLM-based Single Sign-On (SSO) for the Alfresco Share client.

Domain level properties

The following properties control the set of domain controllers used for authentication. The three properties are mutually exclusive. For example, to set the `passthru.authentication.servers` property, set `passthru.authentication.domain` to be empty and `passthru.authentication.useLocalServer` to be false.

passthru.authentication.useLocalServer
A Boolean that when true indicates that the local server should be used for pass through authentication by using loopback connections into the server.

passthru.authentication.domain
Sets the domain to use for pass through authentication. This will attempt to find the domain controllers using a network broadcast. Make sure that you use the Windows NetBIOS domain name, not the forest name. The network broadcast does not work in all network configurations. In this case use the `passthru.authentication.servers` property to specify the domain controller list by name or address.

passthru.authentication.servers
A comma delimited list of server names or addresses that are used for authentication. The pass through authenticator will load balance amongst the available servers, and can monitor server online/offline status.

- Each server name/address may be prefixed with a domain name using the format `<domain>\<server>`. If specifying this in `alfresco-global.properties`, remember that the backslash character must be escaped. For example

```
passthru.authentication.servers=DOMAIN1\\host1.com,DOMAIN2\
\host2.com,host1.com
```

- If the client specifies a domain name in its login request, then the appropriate server will be used for the authentication. Domain mappings may also be specified to route authentication requests to the appropriate server.
- If a server handles authentication for multiple domains then multiple entries can be added in the server list prefixed with each domain name.
- There must be at least one entry in the server list that does not have a domain prefix. This is the catch all entry that will be used if the client domain cannot be determined from the NTLM request or using domain mapping.

Other pass-through properties

ntlm.authentication.sso.enabled
A Boolean that when true enables NTLM based Single Sign On (SSO) functionality in the Web clients. When false and no other members of the authentication chain support SSO, password-based login will be used.

ntlm.authentication.mapUnknownUserToGuest
Identifies whether unknown users are automatically logged on as the Alfresco guest user during Single Sign-On (SSO).

passthru.authentication.authenticateCIFS
A Boolean that when true enables pass-through authentication for the CIFS server. When false and no other members of the authentication chain support CIFS authentication, the CIFS server will be disabled.

passthru.authentication.authenticateFTP
A Boolean that when true enables pass-through authentication for the FTP server. The provided password is hashed and checked directly against the domain server securely using NTLM. When false and no other members of the authentication chain support FTP authentication, standard chained authentication will be used.

passthru.authentication.guestAccess
Identifies whether to allow guest access to Alfresco if the authenticating server indicates the login was allowed guest access.

passthru.authentication.defaultAdministratorUserNames
A comma separated list of user names who should be considered administrators by default. It is often useful to add the administrator user to this list.

passthru.authentication.connectTimeout
The timeout value when opening a session to an authentication server, in milliseconds. The default is 5000.

passthru.authentication.offlineCheckInterval
Specifies how often pass through servers that are marked as offline are checked to see if they are now online. The default check interval is 5 minutes. The check interval is specified in seconds.

passthru.authentication.protocolOrder
Specifies the type of protocols and the order of connection for pass through authentication sessions. The default is to use NetBIOS, if that fails then try to connect using native SMB/ port 445. Specify either a single protocol type or a comma delimited list with a primary and secondary protocol type. The available protocol types are NetBIOS for NetBIOS over TCP and TCPIP for native SMB.

Domain mappings

Domain mappings are used to determine the domain a client is a member of when the client does not specify its domain in the login request. If the client uses a numeric IP address to access the web server it will not send the domain in the NTLM request as the browser assumes it is an Internet address.

To specify the domain mapping rules that are used when the client does not supply its domain in the NTLM request you can use the `filesystem.domainMappings` composite property of the file server subsystem. There are two ways of defining a domain mapping, either by specifying an IP subnet and mask, or by specifying a range of IP addresses. The example below, when included in `alfresco-global.properties` defines mappings for two domains, `ALFRESCO` and `OTHERDOM`. For more information, see Setting composite properties in the global properties file on page 60

```
filesystem.domainMappings=ALFRESCO,OTHERDOM
filesystem.domainMappings.value.ALFRESCO.subnet=192.168.1.0
filesystem.domainMappings.value.ALFRESCO.mask=192.168.1.255
filesystem.domainMappings.value.OTHERDOM.rangeFrom=192.168.1.0
filesystem.domainMappings.value.OTHERDOM.rangeTo=192.168.1.100
```

The pass through subsystem can use the domain prefixed server name format of the `passthru.authentication.servers` property along with the domain mappings to route authentication requests to the appropriate server. A sample NTLM authentication component server list:

```
passthru.authentication.servers=ALFRESCO\\ADSERVER,OTHERDOM\\OTHERSRV
```

Configuring Kerberos

The Java Authentication and Authorization Service (JAAS) is used within the Kerberos subsystem to support Kerberos authentication of user names and passwords. You may choose to use Kerberos against an Active Directory server in preference to LDAP or NTLM as it provides strong encryption without using SSL. It would still be possible to export user registry information using a chained LDAP subsystem.

The disadvantages of using LDAP authentication against Active Directory compared with JAAS/ Kerberos are:

- the simplest approach is to use the SIMPLE LDAP authentication protocol, which should be used with SSL
- AD requires special set up to use digest MD5 authentication (reversible encryption for passwords), which may be difficult retrospectively
- LDAP can use GSSAPI and Kerberos which would be equivalent but this is more difficult to configure and has not been tested

For some pointers and background information on JAAS, the Java Authentication and Authorization Service, refer to the following web sites:

- http://java.sun.com/products/jaas/
- http://en.wikipedia.org/wiki/Java_Authentication_and_Authorization_Service

Kerberos configuration properties

To enable full Kerberos support in Alfresco requires that the CIFS server and the SSO authentication filters each have a Kerberos service ticket.

The Kerberos subsystem supports the following properties.

kerberos.authentication.realm
The Kerberos realm with which to authenticate. The realm should be the domain upper cased; an example is that if the domain is `alfresco.org` then the realm should be `ALFRESCO.ORG`.

kerberos.authentication.sso.enabled
A Boolean that when true enables SPNEGO/Kerberos based Single Sign On (SSO) functionality in the web client. When false and no other members of the authentication chain support SSO, password-based login will be used.

kerberos.authentication.authenticateCIFS
A Boolean that when true enables Kerberos authentication in the CIFS server. When false and no other members of the authentication chain support CIFS authentication, the CIFS server will be disabled.

kerberos.authentication.user.configEntryName
The name of the entry in the JAAS configuration file that should be used for password-based authentication. The default value `Alfresco` is recommended.

kerberos.authentication.cifs.configEntryName
The name of the entry in the JAAS configuration file that should be used for CIFS authentication. The default value `AlfrescoCIFS` is recommended.

kerberos.authentication.http.configEntryName
The name of the entry in the JAAS configuration file that should be used for web-based single-sign on (SSO). The default value `AlfrescoHTTP` is recommended.

kerberos.authentication.cifs.password
The password for the CIFS Kerberos principal.

kerberos.authentication.http.password
The password for the HTTP Kerberos principal.

kerberos.authentication.defaultAdministratorUserNames
A comma separated list of user names who should be considered administrators by default.

Configuring Kerberos against Active Directory

The following instructions describe how to set up accounts under Active Directory for use by Alfresco.

1. Create a user account for the Alfresco CIFS server using the Active Directory Users and Computers application.
   a. Use the **Action > New > User** menu, then enter the full name as `Alfresco CIFS` and the user login name as `alfrescocifs`.
   b. Click **Next**.
   c. Enter a password.
   d. Enable **Password never expires** and disable **User must change password at next logon**.
   e. Click **Finish**.
   f. Right click the new user account name, select **Properties**.
   g. Select the **Account** tab and enable the **Use DES encryption types for this account** and **Do not require Kerberos preauthentication** options in the **Account Options** section.
2. Create a user account for the Alfresco SSO authentication filters, as in step 1, using the full name `Alfresco HTTP` and user login name as `alfrescohttp`.
3. Use the `ktpass` utility to generate key tables for the Alfresco CIFS and web server. The `ktpass` utility is a free download from the Microsoft site, and is also part of the Win2003 Resource Kit. The `ktpass` command can only be run from the Active Directory server.

```
ktpass -princ cifs/<cifs-server-name>.<domain>@<realm> -pass <password> -
mapuser <domainnetbios>\alfrescocifs
-crypto RC4-HMAC-NT -ptype KRB5_NT_PRINCIPAL -out c:\temp
\alfrescocifs.keytab
```

```
ktpass -princ HTTP/<web-server-name>.<domain>@<realm> -pass <password> -
mapuser <domainnetbios>\alfrescohttp
-crypto RC4-HMAC-NT -ptype KRB5_NT_PRINCIPAL -out c:\temp
\alfrescohttp.keytab
```

   a. The `principal` should be specified using the server name and domain in lowercase with the realm in uppercase. The service types should match `cifs` and `HTTP`. For example, `cifs/server.alfresco.org@ALFRESCO.ORG`.
   b. The `realm` should be the domain upper cased; an example is if the domain is `alfresco.org` then the realm should be `ALFRESCO.ORG`.
   c. `<web-server-name>` is the host name that is running the Alfresco server.
   d. `<cifs-server-name>` is the NetBIOS name of the Alfresco CIFS server when running on an Active Directory client or the host name for a client that is not an Active Directory client, that is, not logged onto the domain.
   e. `<domain>` is the DNS domain, for example `alfresco.org`.
   f. `<domainnetbios>` is the netbios name, for example `alfresco`.

Some versions of the `ktpass` command can generate invalid `keytab` files. Download the latest version of the resource kit tools from the Microsoft site to avoid any problems.

4. Create the Service Principal Names (SPN) for the Alfresco CIFS and web server using the `setspn` utility. The `setspn` utility is a free download from the Microsoft site, and is also part of the Win2003 Resource Kit.

   ```
   setspn -a cifs/<cifs-server-name> alfrescocifs
   setspn -a cifs/<cifs-server-name>.<domain> alfrescocifs
   ```

   ```
   setspn -a HTTP/<web-server-name> alfrescohttp
   setspn -a HTTP/<web-server-name>.<domain> alfrescohttp
   ```

   Some versions of the `ktpass` command will add the SPN for the `principal` so you may only need to add the NetBIOS/short name versions of the SPNs. Use the `setspn -l <account-name>` command to check if the `ktpass` command set the SPN. You can list the SPNs for a server using the following:

   ```
   setspn -l <account-name>
   ```

   For example:

   ```
   setspn -l alfrescocifs
   setspn -l alfrescohttp
   ```

5. Copy the key table files created in step 3 to the server where Alfresco will run. Copy the files to a protected area such as `C:\etc\` or `/etc`.
6. Set up the Kerberos `ini` file.

   The default location is `C:\WINNT\krb5.ini` or `/etc/krb5.conf`.

   ```
   [libdefaults]
    default_realm = ALFRESCO.ORG

   [realms]
    ALFRESCO.ORG = {
     kdc = adsrv.alfresco.org
     admin_server = adsrv.alfresco.org
    }

   [domain_realm]
    adsrv.alfresco.org = ALFRESCO.ORG
    .adsrv.alfresco.org = ALFRESCO.ORG
   ```

   The realm should be specified in uppercase.

7. Set up the Java login configuration file. This would usually be in the `JRE\lib\security` folder. Create a file named `java.login.config` with the following entries:

   ```
   Alfresco {
      com.sun.security.auth.module.Krb5LoginModule sufficient;
   };

   AlfrescoCIFS {
      com.sun.security.auth.module.Krb5LoginModule required
      storeKey=true
      useKeyTab=true
      keyTab="C:/etc/alfrescocifs.keytab"
      principal="cifs/<cifs-server-name>.<domain>";
   };

   AlfrescoHTTP {
      com.sun.security.auth.module.Krb5LoginModule required
      storeKey=true
      useKeyTab=true
      keyTab="C:/etc/alfrescohttp.keytab"
      principal="HTTP/<web-server-name>.<domain>";
   };

   com.sun.net.ssl.client {
      com.sun.security.auth.module.Krb5LoginModule sufficient;
   };
   ```

```
other {
   com.sun.security.auth.module.Krb5LoginModule sufficient;
};
```

8. Enable the login configuration file in the main Java security configuration file, usually at `JRE\lib\security\java.security`. Add the following line:

   ```
   login.config.url.1=file:${java.home}/lib/security/java.login.config
   ```

Kerberos client configuration

1. When using Firefox on Windows as your client, you will need to add your Alfresco server name to the `network.negotiate-auth.trusted-uris` variable You can access the variable going to the special URL: `about:config`.
2. When using Firefox under Linux, you will need to add your Alfresco server name to `network.negotiate-auth.trusted-uris` but you will need, in addition, to get a Kerberos ticket using the `kinit` command.

   The ticket can correspond to a different user than your Linux user name.

   For example:

   ```
   kinit user1
   ```

   Where `user1` is an Active Directory user. If the client and the server are on the same machine, you will need to go to the `eternl` interface. The `loopback` interface will not be able to authenticate. You can view your tickets using `klist`.

Debugging Kerberos

You can debug Kerberos issues using the log4j properties.

For example:

```
log4j.logger.org.alfresco.web.app.servlet.KerberosAuthenticationFilter=debug
log4j.logger.org.alfresco.repo.webdav.auth.KerberosAuthenticationFilter=debug
```

The following is a sample login output:

```
18:46:27,915 DEBUG [app.servlet.KerberosAuthenticationFilter] New Kerberos auth
 request from 192.168.4.95 (192.168.4.95:38750)
18:46:28,063 DEBUG [app.servlet.KerberosAuthenticationFilter] User user1 logged
 on via Kerberos
```

Configuring external authentication

The `external` authentication subsystem can be used to integrate Alfresco with any external authentication system.

The external authentication system can be integrated with your application server in such a way that the identity of the logged-in user is passed to servlets via the `HttpServletRequest.getRemoteUser()` method. As this is the standard way for application servers to propagate user identities to servlets, it should be compatible with a number of SSO solutions, including Central Authentication Service (CAS).

The subsystem also allows a proxy user to be configured, such that requests made through this proxy user are made in the name of an alternative user, whose name is carried in a configured HTTP request header. This allows, for example, the Share application and other Alfresco Surf applications to act as a client to an SSO-protected Alfresco application and assert the user name in a secure manner.

External configuration properties

The external subsystem supports the following properties.

external.authentication.enabled
A Boolean property that when true indicates that this subsystem is active and will trust remote user names asserted to it by the application server.

external.authentication.defaultAdministratorUserNames
A comma separated list of user names who should be considered administrators by default.

external.authentication.proxyUserName
The name of the remote user that should be considered the proxy user.
The default is `alfresco-system`. Requests made by this user will be made under the identity of the user named in the HTTP Header indicated by the `external.authentication.proxyHeader` property. If not set, then the HTTP Header indicated by the `external.authentication.proxyHeader` property is always assumed to carry the user name.

This is not secure unless this application is not directly accessible by other clients.

external.authentication.proxyHeader
The name of the HTTP header that carries the name of a proxied user. The default is `X-Alfresco-Remote-User`, as used by Share.

external.authentication.userIdPattern
An optional regular expression to be used to extract a user ID from the HTTP header. The portion of the header matched by the first bracketed group in the regular expression will become the user name. If not set (the default), then the entire header contents are assumed to be the proxied user name.

Configuring OpenOffice

Within Alfresco, you can transform a document from one format to another and this feature requires you to install OpenOffice.

Alfresco supports the following OpenOffice subsystems:

OOoDirect
The direct OpenOffice integration, in which the Alfresco server manages OpenOffice directly. This subsystem is enabled, by default.

OOoJodconverter
The JODConverter integration, which is a library that improves the stability and performance of OpenOffice.org (OOo) within Alfresco. This subsystem is disabled, by default. The OOoJODConverter runs on the same machine as the Alfresco server. The JODConverter supports:

- a pool of separate OpenOffice processes
- automatic restart of crashed OpenOffice processes
- automatic termination of slow OpenOffice operations
- automatic restart of any OpenOffice process after a number of operations (this is a workaround for OpenOffice memory leaks)

Changing the OpenOffice subsystem

The default subsystem for OpenOffice transformations is OOoDirect. You can change the preferred OpenOffice subsystem to OOoJodconverter.

The JODConverter requires OpenOffice.org 3.0.0 or later and recommends 3.1.0+.

There are two methods that you can use to change OpenOffice subsystem.

- `alfresco-global.properties` file

- Runtime administration using your JMX client

Global properties file

1. Open the `alfresco-global.properties` file.
2. Uncomment the following lines:
   ```
   #ooo.enabled=false
   #jodconverter.enabled=true
   ```
3. Save the file.
4. Restart the Alfresco server.

JMX interface runtime administration

1. Open your JMX client, for example, JConsole.
2. Locate the **OOoDirect** subsystem.
3. Edit the **ooo.enabled** value to `false`.
4. Restart the subsystem.
5. Locate the **OOoJodconverter** subsystem.
6. Edit the **jodconverter.enabled** value to true.
7. Restart the subsystem.

Although it is possible to run both subsystems, Alfresco recommends that you enable only one at a time.

OOoDirect subsystem configuration properties

The following properties can be configured for the OOoDirect subsystem.

ooo.exe
Specifies the OpenOffice installation path.

ooo.enabled
Enables or disables the OOoDirect subsystem.

OOoJodconverter subsystem configuration properties

The following properties can be configured for the OOoJodconverter subsystem.

jodconverter.enabled
Enables or disables the JodConverter process(es).

jodconverter.maxTasksPerProcess
Specifies the number of transforms before the process restarts. The default is 200.

jodconverter.officeHome
Specifies the name of the OpenOffice install directory. For example, (Windows) `C:/alfresco/OpenOffice.org` and (Mac) `/Applications/OpenOffice.org.app/Contents`.

jodconverter.portNumbers
Specifies the port numbers used by each processing thread. The number of process will match the number of ports. The default numbers are 2022, 2023, and 2024.

jodconverter.taskExecutionTimeout
Specifies the maximum number of milliseconds that an operation is allowed to run before it is aborted. It is used to recover from operations that have hung. The default is 120000 milliseconds (2 minutes).

jodconverter.taskQueueTimeout
Specifies the maximum number of milliseconds a task waits in the transformation queue before the process restarts. It is used to recover hung OpenOffice processes. The default is 30000 milliseconds (30 seconds).

Configuring synchronization

The synchronization subsystem manages the synchronization of Alfresco with all the user registries (LDAP servers) in the authentication chain.

The synchronization subsystem supports two modes of synchronization:

Full
All users and groups are queried, regardless of when they were last modified. All local copies of these users and groups already existing are then updated and new copies are made of new users and groups. Since processing all users and groups in this manner may be fairly time consuming, this mode of synchronization is usually only triggered on the very first sync when the subsystem first starts up. However, synchronization can also be triggered in this mode by the scheduled synchronization job, if `synchronization.synchronizeChangesOnly` is set to false.

Differential
Only those users and groups changed since the last query are queried and created/updated locally. This differential mode is much faster than full synchronization. By default, it is triggered when the subsystem starts up after the first time and also when a user is successfully authenticated who does not yet have a local person object in Alfresco. This means that new users, and their group information, are pulled over from LDAP servers as and when required with minimal overhead.

Synchronization deletion

Users and groups that are created as a result of a synchronization operation are tagged with an originating zone ID. This records the identifier of the authentication subsystem instance from which the user or group was queried.

On synchronization with a zone, only those users and groups tagged with that zone are candidates for deletion. This avoids the accidental deletion of built-in groups, such as `ALFRESCO_ADMINISTRATORS`.

Collision resolution

If there are overlaps between the contents of two user registries in the authentication chain (for example, where two user registries both contain a user with the same user name), then the registry that occurs earlier in the authentication chain will be given precedence. This means that exactly the same order of precedence used during authentication will be used during synchronization.

For example, if user `A` is queried from zone `Z1` but already exists in Alfresco in zone `Z2`:

- `A` is moved to `Z1` if it is earlier in the authentication chain
- `A` is ignored if `Z1` is later in the authentication chain

Synchronization configuration properties

The following properties can be configured for the synchronization subsystem.

synchronization.synchronizeChangesOnly
Specifies if the scheduled synchronization job is run in differential mode. The default is false, which means that the scheduled sync job is run in full mode. Regardless of this setting a differential sync may still be triggered when a user is successfully authenticated who does not yet exist in Alfresco.

synchronization.import.cron
Specifies a cron expression defining when the scheduled synchronization job should run, by default at midnight every day.

synchronization.syncOnStartup
Specifies whether to trigger a differential sync when the subsystem starts up. The default is true. This ensures that when user registries are first configured, the bulk of the synchronization work is done on server startup, rather than on the first login.

synchronization.syncWhenMissingPeopleLogIn
Specifies whether to trigger a differential sync when a user is successfully authenticated who does not yet exist in Alfresco. The default is true.

synchronization.autoCreatePeopleOnLogin
Specifies whether to create a user with default properties when a user is successfully authenticated, who does not yet exist in Alfresco, and was not returned by a differential sync (if enabled with the property above). The default is true. Setting this to false allows you to restrict Alfresco to a subset of those users who could be authenticated by LDAP; only those created by synchronization are allowed to log in. You can control the set of users in this more restricted set by overriding the user query properties of the LDAP authentication subsystem.

Configuring file servers

The File Server subsystem allows access to the Alfresco data stores through the SMB/CIFS, FTP, and NFS protocols. This allows you to browse to the repository using Windows Explorer or by creating a Network Place.

As with other Alfresco subsystems, the File Server subsystem exposes all of its configuration options as properties that can be controlled through a JMX interface or the global properties file.

The sections that follow describe each of the configurable properties supported by the File Server subsystem.

Configuring SMB/CIFS server

The server includes Java socket-based implementations of the SMB/CIFS protocol that can be used on any platform.

The server can listen for SMB traffic over the TCP protocol (native SMB) supported by Windows 2000 and later versions, and the NetBIOS over TCP (NBT) protocol, supported by all Windows versions. There is also a Windows-specific interface that uses Win32 NetBIOS API calls using JNI code. This allows the Alfresco CIFS server to run alongside the native Windows file server.

The default configuration uses the JNI-based code under Windows and the Java socket based code under Linux, Solaris, and Mac OS X.

CIFS file server properties

The following properties can be configured for the SMB/CIFS server.

cifs.enabled
Enables or disables the CIFS server.

cifs.serverName
Specifies the host name for the Alfresco CIFS server. This can be a maximum of 16 characters and must be unique on the network. The special token `{localname}` can be used in place of the local server's host name and a unique name can be generated by prepending/ appending to it.

cifs.domain
An optional property. When not empty, specifies the domain or workgroup to which the server belongs. This defaults to the domain/workgroup of the server, if not specified.

cifs.hostannounce
Enables announcement of the CIFS server to the local domain/workgroup so that it shows up in Network Places/Network Neighborhood.

cifs.sessionTimeout
Specifies the CIFS session timeout value in seconds. The default session timeout is 15 minutes. If no I/O occurs on the session within this time then the session will be closed by the server. Windows clients send keep-alive requests, usually within 15 minutes.

Java-based SMB properties

The following properties will only take effect on non-Windows servers, where the Java-based SMB implementation is used, unless it is enabled on Windows using the advanced Spring bean definition overrides.

cifs.broadcast
Specifies the broadcast mask for the network.

cifs.bindto
Specifies the network adapter to which to bind. If not specified, the server will bind to all available adapters/addresses.

cifs.tcpipSMB.port
Controls the port used to listen for the SMB over TCP/IP protocol (or native SMB), supported by Win2000 and above clients. The default port is 445.

cifs.ipv6.enabled
Enables the use of IP v6 in addition to IP v4 for native SMB. When `true`, the server will listen for incoming connections on IPv6 and IPv4 sockets.

cifs.netBIOSSMB.namePort
Controls the NetBIOS name server port on which to listen. The default is 137.

cifs.netBIOSSMB.datagramPort
Controls the NetBIOS datagram port. The default is 138.

cifs.netBIOSSMB.sessionPort
Controls the NetBIOS session port on which to listen for incoming session requests. The default is 139.

cifs.WINS.autoDetectEnabled
When `true` causes the `cifs.WINS.primary` and `cifs.WINS.secondary` properties to be ignored.

cifs.WINS.primary
Specifies a primary WINS server with which to register the server name.

cifs.WINS.secondary
Specifies a secondary WINS server with which to register the server name.

cifs.disableNIO
Disables the new NIO-based CIFS server code and reverts to using the older socket based code.

Windows native SMB

The following property will only take effect on Windows servers, where by default a JNI-based CIFS implementation is in use.

cifs.disableNativeCode
When `true`, switches off the use of any JNI calls and JNI-based CIFS implementations.

Running SMB/CIFS from a normal user account

On Unix-like systems such as Linux and Solaris, the default Alfresco setup must be run using the root user account so that the CIFS server can bind to the privileged ports (TCP 139/445 UDP 137/138).

The CIFS server can be configured to run using non-privileged ports and then use firewall rules to forward requests from the privileged ports to the non-privileged ports.

1. To configure the CIFS server to use non-privileged ports, use the following property settings:

```
cifs.tcpipSMB.port=1445
cifs.netBIOSSMB.namePort=1137
cifs.netBIOSSMB.datagramPort=1138
cifs.netBIOSSMB.sessionPort=1139
```

   Other port numbers can be used but must be above 1024 to be in the non-privileged range.

   The firewall rules should then be set up to forward requests:

   - TCP ports 139/445 to TCP 1139/1445
   - UDP ports 137/138 to UDP 1137/1138

2. On Mac OS X the following commands can be used:

```
sysctl -w net.inet.ip.fw.enable=1
sysctl -w net.inet.ip.forwarding=1
sysctl -w net.inet.ip.fw.verbose=1
sysctl -w net.inet.ip.fw.debug=0
ipfw flush
ipfw add 100 allow ip from any to any via lo0
# Forward native SMB and NetBIOS sessions to non-privileged ports
ipfw add 200 fwd <local-ip>,1445 tcp from any to me dst-port 445
ipfw add 300 fwd <local-ip>,1139 tcp from any to me dst-port 139
# Forward NetBIOS datagrams to non-privileged ports (does not currently
 work)
ipfw add 400 fwd <local-ip>,1137 udp from any to me dst-port 137
ipfw add 500 fwd <local-ip>,1138 udp from any to me dst-port 138
```

   Replace `<local-ip>` with the IP address of the server that Alfresco is running on.

3. On Linux, the following commands can be used to get started, but be aware these commands will delete all existing firewall and NAT rules and could be a security risk:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
modprobe iptable_nat
iptables -F
iptables -t nat -F
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
iptables -t nat -A PREROUTING -p tcp --dport 445 -j REDIRECT --to-ports
 1445
iptables -t nat -A PREROUTING -p tcp --dport 139 -j REDIRECT --to-ports
 1139
iptables -t nat -A PREROUTING -p udp --dport 137 -j REDIRECT --to-ports
 1137
iptables -t nat -A PREROUTING -p udp --dport 138 -j REDIRECT --to-ports
 1138
```

The UDP forwarding does not work, which affects the NetBIOS name lookups. A workaround is either to add a DNS entry matching the CIFS server name and/or add a static WINS mapping, or add an entry to the clients `LMHOSTS` file.

SMB/CIFS advanced Spring overrides

The SMB/CIFS server beans are declared in the `file-servers-context.xml` file in `<configRoot>\classes\alfresco\subsystems\fileServers\default\`. Using the subsystem extension classpath mechanism, you can place site specific customization of these default values in a Spring bean file in `<extension>\subsystems\fileServers\default\default\custom-file-servers-context.xml` (note that the `default\default` part of the path is intentional).

The main bean that drives the CIFS server configuration is called `cifsServerConfig`. This has several properties that can be populated with child beans that control various optional SMB implementations.

tcpipSMB
Controls the Java-based SMB over TCP/IP implementation, which is compatible with Windows 2000 clients and later.

netBIOSSMB
Controls the Java-based NetBIOS over TCP/IP implementation, which is compatible with all Windows clients.

win32NetBIOS
Controls the JNI-based NetBIOS over TCP/IP implementation, which is only enabled for Alfresco servers running on Windows.

When one of the above properties is not set, it deactivates support for the corresponding protocol implementation. The `tcpipSMB` and `netBIOSSMB` beans have a platforms property that allows their configuration to be targeted to Alfresco servers running on specific platforms. The property is formatted as a comma-separated list of platform identifiers. Valid platform identifiers are `linux`, `solaris`, `macosx`, and `aix`.

1. To run the native SMB over TCP/IP protocol under Windows, you need to disable Windows from using the port by editing, or creating, the following registry key:

   ```
   [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetBT\Parameters]
    "SMBDeviceEnabled"=dword:00000000
   ```
2. To enable the Java socket based NetBIOS implementation under Windows disable NetBIOS on one or all network adapters.

   This can be done using the Network Connections control panel in the advanced TCP/IP properties for each adapter.
3. The `serverComment` of the `cifsServerConfig` bean controls the comment that is displayed in various information windows.
4. The `sessionDebugFlags` property of the `cifsServerConfig` bean enables debug output levels for CIFS server debugging. The value should be in the form of a comma-separated list of the flag names.

Flag	Description
`NetBIOS`	NetBIOS layer
`State`	Session state changes
`Tree`	File system connection/disconnection
`Search`	Folder searches
`Info`	File information requests

Flag	Description
`File`	File open/close
`FileIO`	File read/write
`Tran`	Transaction requests
`Echo`	Echo requests
`Errors`	Responses returning an error status
`IPC`	IPC$ named pipe
`Lock`	File byte range lock/unlock
`Pkttype`	Received packet type
`Dcerpc`	DCE/RPC requests
`Statecache`	File state caching
`Notify`	Change notifications
`Streams`	NTFS streams
`Socket`	NetBIOS/native SMB socket connections
`PktPool`	Memory pool allocations/de-allocations
`PktStats`	Memory pool statistics dumped at server shutdown
`ThreadPool`	Thread pool

5. The `log4j.properties` file must also have SMB/CIFS protocol debug output enabled using:
   ```
   log4j.logger.org.alfresco.smb.protocol=debug
   ```
6. The following logging level must also be enabled to log debug output from the core file server code:
   ```
   log4j.logger.org.alfresco.fileserver=debug
   ```

Configuring the FTP file server

This section describes how to configure the FTP file server.

FTP file server properties

The following properties can be configured for the FTP server.

ftp.enabled
Enables or disables the FTP server.

ftp.port
Specifies the port that the FTP server listens for incoming connections on. Defaults to port 21.

On some platforms, ports below 1024 require the server to be run under a privileged account.

FTP advanced Spring overrides

The FTP server beans are declared in the `file-servers-context.xml` file in `<configRoot>\classes\alfresco\subsystems\fileServers\default`. Using the subsystem extension classpath mechanism, site specific customisation of these default values can be placed in a Spring bean file in `<extension>\subsystems\fileServers\default\default\custom-file-servers-context.xml` (note that the `default\default` part of the path is intentional).

The following properties can be overridden on the `ftpServerConfig` bean.

bindTo

Specifies the address the FTP server binds to, if not specified the server will bind to all available addresses.

rootDirectory

Specifies the path of the root directory as an FTP format path, that is, using forward slashes. The first part of the path should be the file system name, optionally followed by one or more folder names, for example:

```
/Alfresco/myfolder/
```

charSet

Specifies the character set to be used. The character set name should be a valid Java character set, see the Java `CharSet` class for more information.

1. The `debugFlags` property enables debug output levels for FTP server debugging. The value should be a comma-separated list of flag names from the following table:

Flag	Description
`State`	Session state changes
`Search`	Folder searches
`Info`	File information requests
`File`	File open/close
`FileIO`	File read/write
`Error`	Errors
`Pkttype`	Received packet type
`Timing`	Time packet processing
`Dataport`	Data port
`Directory`	Directory commands

2. Configure logging levels for the FTP server in `$ALF_HOME/tomcat/webapps/alfresco/WEB-INF/classes/log4j.properties` using:

```
log4j.logger.org.alfresco.ftp.protocol=debug
log4j.logger.org.alfresco.ftp.server=debug
```

Configuring the NFS file server

It is recommended that TCP connections are used to connect to the Alfresco NFS server. Using a read/write size of 32K will also help to optimize the performance.

NFS file server properties

The following properties can be configured for the NFS server.

nfs.enabled

Enables or disables the NFS server.

nfs.user.mappings

A composite property that configures the user ID/group ID to the Alfresco user name mappings that are used by the current RPC authentication implementation.

For example, the following configuration gives `admin` a `uid` and `gid` of 0 and `auser` a `uid` and `gid` of 501.

```
nfs.user.mappings=admin,auser
nfs.user.mappings.value.admin.uid=0
nfs.user.mappings.value.admin.gid=0
nfs.user.mappings.value.auser.uid=501
nfs.user.mappings.value.auser.gid=501
```

NFS advanced Spring overrides

The NFS server beans are declared in the `file-servers-context.xml` file in `<configRoot>\classes\alfresco\subsystems\fileServers\default`. Using the subsystem extension classpath mechanism, site specific customisation of these default values can be placed in a Spring bean file in `<extension>\subsystems\fileServers\default\default\custom-file-servers-context.xml` (note that the default\default part of the path is intentional).

The following properties can be overridden on the `nfsServerConfig` bean.

portMapperEnabled
Enables the built-in portmapper service. This would usually be enabled on Windows where there is no default portmapper service. Under Linux/Unix operating systems, the built-in portmapper service can be used, which also saves having to run the Alfresco server using the root account.

threadPool
Sets the size of the RPc processing thread pool. The minimum number of threads is 4, the default setting is 8.

packetPool
Sets the size of the packet pool used to receive RPC requests and send RPC replies. The minimum number of packets is 10, the default setting is 50.

portMapperPort
The port number to run the portmapper service on. The default port is 111.

mountServerPort
The port number to run the mountserver service on. The default is to allocate an available non-privileged port.

nfsServerPort
The port number to run main NFS server service on. The default is to allocate the default NFS port: 2049. This will likely clash with a running native NFS server.

1. The `debugFlags` property enables debug output levels for NFS server debugging. The value should be in the form of a comma-separated list of the flag names in the following table.

Flag	Description
`RxData`	Request data details
`TxData`	Response data details
`DumpData`	Hex dump request/response data
`Search`	Folder searches
`Info`	File information requests
`File`	File open/close
`FileIO`	File read/write
`Error`	Errors
`Directory`	Directory commands
`Timing`	Time packet processing
`Session`	Session creation/deletion

2. The log4j.properties file must also have NFS protocol debug output enabled using:

```
log4j.logger.org.alfresco.nfs.server=debug
```

3. The following logging level must also be enabled to log debug output from the core file server code:

```
log4j.logger.org.alfresco.fileserver=debug
```

4. There are also the following log4j output options for the NFS/mount/portmapper services:

```
log4j.logger.org.alfresco.nfs.protocol=debug
```

5. Output server level debug information from the NFS, mount and portmapper services.

```
log4j.logger.org.alfresco.nfs.protocol.auth=debug
```

Configuring email

The email subsystem allows you to configure the outbound and inbound SMTP email settings to interact with Alfresco.

There are two methods of running Alfresco email server:

- Running the email server process in the same JVM context as the repository
- Running the email server remotely and communicate with the repository using Remote Method Invocation (RMI)

OutboundSMTP configuration properties

The following properties can be configured for the OutboundSMTP subsystem type.

You must set the Outbound email configuration for Share invitations to work correctly. If you do not set the email configuration, when you invite a user to a site, the user will not receive the assigned task notification.

mail.host=yourmailhost.com
Specifies the name of the SMTP host.

mail.port=25
Specifies the port number on which the SMTP service runs (the default is 25).

mail.username=username
Specifies the user name of the account from which you want email to be sent.

mail.password=password
Specifies the password for the user name.

mail.encoding=UTF-8
Specifies UTF-8 encoding for email.

mail.from.default=admin@alfresco.com
Specifies the email address from which all email notifications are sent.

mail.smtp.auth=false
Specifies whether you authorization is required.

mail.protocol=smtp
Specifies the default protocol to be used for email.

The following properties can be set to define a test message when the subsystem starts.

mail.testmessage.send=false
Defines whether or not to send a test message.

mail.testmessage.to=
Specifies the recipient of the test message.

mail.testmessage.subject=Outbound SMTP
Specifies the message subject of the test message.

mail.testmessage.text=Outbound SMTP email subsystem is working.
Specifies the message body of the test message.

InboundSMTP configuration properties

The InboundSMTP email subsystem type allows you to configure the behavior of the email server and service.

The following properties can be set for Inbound SMTP email.

email.inbound.unknownUser=anonymous
Specifies the user name to authenticate as when the sender address is not recognized.

email.inbound.enabled=true
Enables or disables the inbound email service. The service could be used by processes other than the email server (for example, direct RMI access), so this flag is independent of the email service.

email.server.enabled=true
Enables the email server.

email.server.port=25
Specifies the default port number for the email server.

email.server.domain=alfresco.com
Specifies the default domain for the email server.

email.server.allowed.senders=.*
Provides a comma-separated list of email REGEX patterns of allowed senders. If there are any values in the list, then all sender email addresses must match. For example: `.*\@alfresco\.com`, `.*\@alfresco\.org`.

email.server.blocked.senders=
Provides a comma-separated list of email REGEX patterns of blocked senders. If the sender email address matches this, then the message will be rejected. For example: `.*\@hotmail\.com`, `.*\@googlemail\.com`.

Configuring the RMI email service

You can run the email server remotely on a separate JVM and server, and have it interact with the Alfresco server using Remote Method Invocation (RMI).

1. Browse to the folder `<configRoot>/classes/alfresco`.
2. Open the configuration file `remote-email-service-context.xml`.

   This file contains the `emailService` bean, specifying the related RMI configuration.
3. Modify the RMI configuration as required. For example:

Value	Description
`<serviceUrl></serviceUrl>`	Specifies the valid RMI URL.
`<serviceInterface></serviceInterface>`	Specifies the interface used for RMI.

Handling messages by target node type

This section describes the default behaviors for incoming email to different types of referenced nodes.

You can modify or extend the default behaviors by adding in custom handlers.

Folder(Space)
Content added with emailed aspect.

Forum(Discussion)
Content specialized to Post with emailed aspect; if email subject matches a topic, then add to topic, otherwise create new topic based on subject.

Topic(Post)
Content specialized to Post with emailed aspect; if referenced node is a Post, add to Post’s parent Topic.

Document(Content)
If discussion exists, same as for forums, otherwise add discussion with email as initial topic and post.

Groups and permissions for email

An email arriving at the Alfresco email server is unauthenticated. An authentication group, `EMAIL_CONTRIBUTORS` , must be created to allow permissions to be handled at a high level by the administrator.

When an email comes into the system, the only identification is the sender's email address. The user is look-up is based on the email address.

- If a matching user is not found, then the current user is assumed to be unknown, if unknown exists
- If unknown does not exist, then the email is rejected as authentication will not be possible
- If the user selected is not part of email contributor's group, then the email is rejected

The current request's user is set and all subsequent processes are run as the authenticated user. If any type of authentication error is generated, then the email is rejected. The authentication will also imply that the authenticated user may not have visibility of the target node, in which case the email is also rejected. Effectively, this means that the target recipient of the email does not exist, at least not for the sender.

The current default server configuration creates the `EMAIL_CONTRIBUTORS` group and adds the `admin` user to this group.

Configuring IMAP Protocol support

IMAP protocol support allows email applications that support IMAP (including Outlook, Apple Mail, Thunderbird, and so on) to connect to and interact with Alfresco repositories, directly from the email application.

The IMAP client must be configured to use `localhost` as its incoming IMAP mail server.

IMAP mount points

IMAP mount points are used to control which folders are available using IMAP and the mode in which they are accessed. Modes are used to define the type of interaction available.

The IMAP integration offers the following access modes:

Archive
Allows emails to be written to and read from Alfresco by the IMAP client by drag and drop, copy/paste, and so on, from the email client.

Virtual
Documents managed by Alfresco may be viewed as emails from the IMAP client. Documents are shown as virtual emails with the ability to view metadata and trigger actions on the document, using links included in the email body.

Mixed

A combination of both archive and virtual modes, that is, both document access and email management are available.

By default, a single mount point called **IMAP Home** is defined for **Company Home** running in Mixed mode.

Enabling the IMAP Protocol

The IMAP protocol server is disabled by default. You need to enable the IMAP protocol server to start interaction between the email client and the Alfresco repository.

1. Open the `alfresco-global.properties` file.
2. Uncomment the following sample configuration entries:

```
imap.server.enabled=true
imap.server.port=143
```

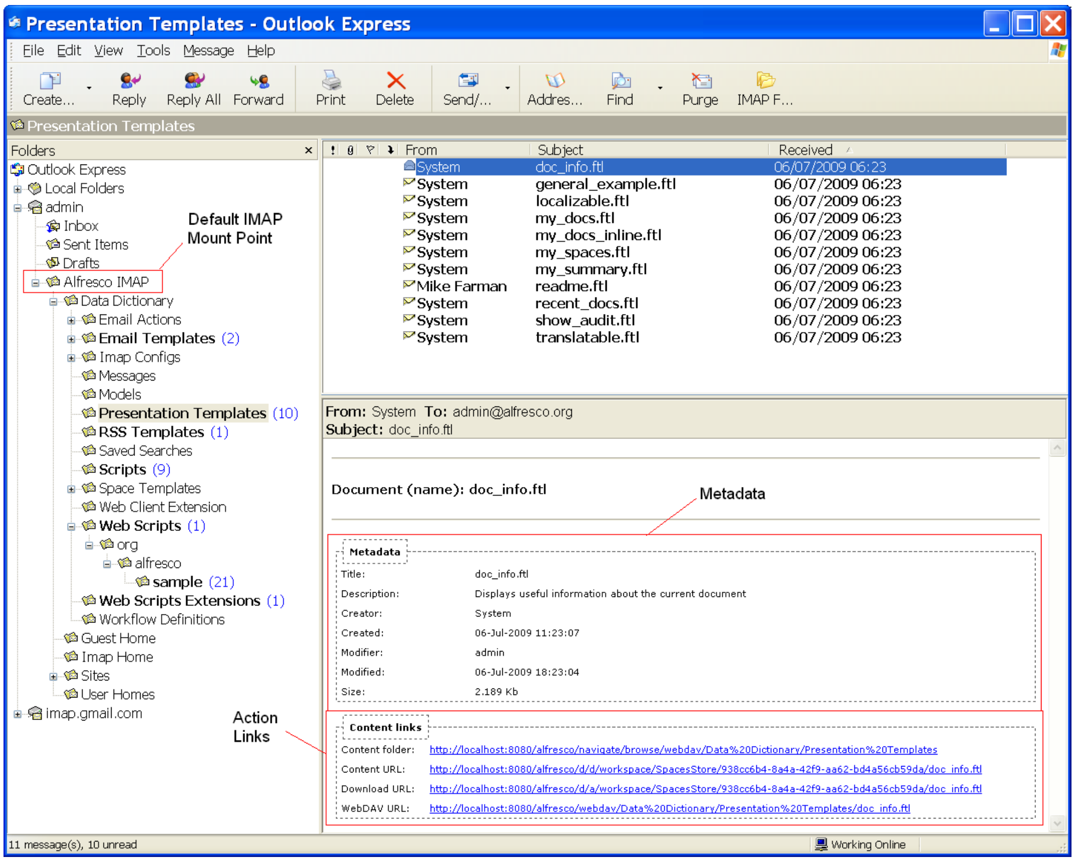
3. Restart your Alfresco server.

Once the Alfresco server has restarted, the new configuration will take effect.

Virtual view email format

The virtualized view uses presentation templates to generate the mail body and display document metadata, action links (for download, view, webdav, folder) and Start Workflow form (HTML view only).

The templates are stored in the repository in **Company Home > Data Dictionary > Imap Configs > Templates**. Separate templates are available to generate either a HTML or plain text body, based on the the format request by the email client. The templates can be customized to change the metadata and actions available in the email body.



From	Subject	Received
System	doc_info.ftl	06/07/2009 06:23
System	general_example.ftl	06/07/2009 06:23
System	localizable.ftl	06/07/2009 06:23
System	my_docs.ftl	06/07/2009 06:23
System	my_docs_inline.ftl	06/07/2009 06:23
System	my_spaces.ftl	06/07/2009 06:23
System	my_summary.ftl	06/07/2009 06:23
Mike Farman	readme.ftl	06/07/2009 06:23
System	recent_docs.ftl	06/07/2009 06:23
System	show_audit.ftl	06/07/2009 06:23
System	translatable.ftl	06/07/2009 06:23

From: System **To:** admin@alfresco.org
Subject: doc_info.ftl

Document (name): doc_info.ftl

Metadata

Title:	doc_info.ftl
Description:	Displays useful information about the current document
Creator:	System
Created:	06-Jul-2009 11:23:07
Modifier:	admin
Modified:	06-Jul-2009 18:23:04
Size:	2.189 Kb

Content links

Content folder:	http://localhost:8080/alfresco/navigate/browse/webdav/Data%20Dictionary/Presentation%20Templates
Content URL:	http://localhost:8080/alfresco/d/d/workspace/SpacesStore/938cc6b4-8a4a-42f9-aa62-bd4a56cb59da/doc_info.ftl
Download URL:	http://localhost:8080/alfresco/d/a/workspace/SpacesStore/938cc6b4-8a4a-42f9-aa62-bd4a56cb59da/doc_info.ftl
WebDAV URL:	http://localhost:8080/alfresco/webdav/Data%20Dictionary/Presentation%20Templates/doc_info.ftl

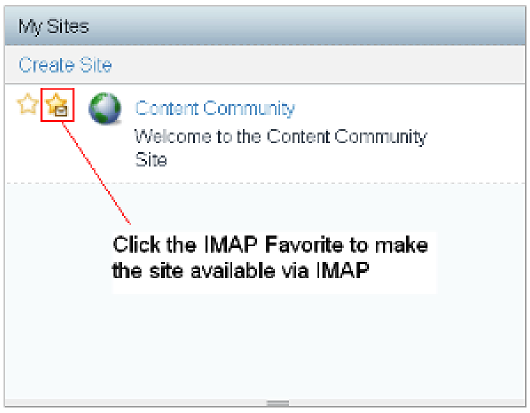
11 message(s), 10 unread

Working Online

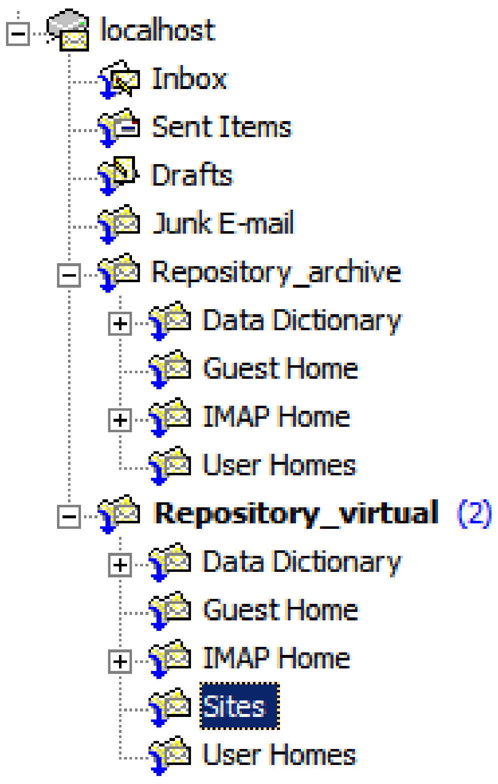
Marking sites as IMAP favorites

To have access to Alfresco Share sites using IMAP, the site(s) need to be added to your list of sites using Share IMAP Favorites.

1. Select **IMAP Favorites** in the Share **My Sites** dashlet on your **My Dashboard** page:


My Sites
Create Site
Content Community
Welcome to the Content Community Site
Click the IMAP Favorite to make the site available via IMAP

2. Refresh your IMAP view to see the new sites.



You can see the site added to the IMAP Sites folder.

Configuring system properties

The sysAdmin subsystem allows real time control across some of the general repository properties. The sysAdmin subsystem replaces the `RepoServerMgmt` management bean.

sysAdmin subsystem properties

The following properties can be configured for the sysAdmin subsystem.

server.maxusers
The maximum number of users who are allowed to log in or -1 if there is no limit.

server.allowedusers
A comma-separated list of users who are allowed to log in. Leave empty if all users are allowed to log in.

server.transaction.allow-writes
A Boolean property that when true indicates that the repository will allow write operations (provided that the license is valid). When false the repository is in read-only mode.

The following properties specify the parameters that control how Alfresco generates URLs to the repository and Share. These parameters may need to be edited from their default values to allow the URLs to be resolved by an external computer.

alfresco.context
Specifies the context path of the Alfresco repository web application. The default is `alfresco`.

alfresco.host
Specifies the host name where the Alfresco server is running. The default value is `${localname}`. If this is used for the value of this property, the token `${localname}` will be automatically replaced by the domain name of the repository server.

alfresco.port
Specifies the port number. The default is `8080`.

alfresco.protocol
Specifies the protocol to use. The default is `http`.

share.context
Specifies the Share root context. The default is `share`.

share.host
Specifies the host name where the Share web tier is running. The default value is `${localname}`.

share.port
Specifies port number. The default is `8080`.

share.protocol
Specifies the protocol to use. The default is `http`.

Configuring WCM deployment receiver properties

The WCM deployment receiver subsystem allows real time control of the WCM deployment receiver properties.

WCM deployment receiver subsystem properties

The following properties can be configured for the WCM deployment receiver subsystem.

wcm-deployment-receiver.poll.delay
Specifies how long to wait before polling. For example, `5000`.

wcm-deployment-receiver.rmi.service.port
Specifies the port number for the RMI service. For example, `44101`.

Customizing Alfresco Share

This section describes how to customize Alfresco Share configuration items.

A number of options are available to customize Alfresco Share. Many of these mechanisms are provided by the underlying Surf platform, therefore a knowledge of Surf is useful for anyone wishing to implement substantial customizations.

The following files are loaded by SURF and therefore any configuration changes will affect all SURF-based applications.

- `<web-extension>\webscript-framework-config-custom.xml.sample`
- `<web-extension>\web-framework-config-custom.xml.sample`

To ensure that the configurations you make to forms affect only the Share application, you can use the custom configuration file named `share-config-custom.xml.sample`.

1. Open the following file:

   `<web-extension>/share-config-custom.xml.sample`
2. Uncomment any `<config>` items that you want to enable.
3. Add any `<config>` items that you want to include.
4. Save the edited file without the `.sample` extension.

Share repository document library

The Share repository document library is a feature that gives full access to the Alfresco repository.

The default content structure for Alfresco Share is based on sites, and this does not give full visibility of the content in the repository. By enabling the repository document library configuration setting, you have access to multiple navigation options, for example, folders and categories, tags, and filters. This feature also allows you to recreate and edit text files, for example, within the Data Dictionary.

It is possible to copy or move files to the Share document library without any repository permissions.

Configuring the Share repository document library

The repository document library is included in the `share.war` file. If you install Alfresco using the one of the bundles or the WAR file, you must manually enable this feature.

1. Edit the following file:

   `<web-extension>/share-config-custom.xml.sample`
2. Locate the `Repository Library config section`.

```
<!-- Repository Library config section -->
   <config evaluator="string-compare" condition="RepositoryLibrary"
 replace="true">
      <!--
         Whether the link to the Repository Library appears in the header
 component or not.
      -->
      <visible>false</visible>
```
3. In the `<visible>false</visible>` line, replace `false` with `true`.

4. Save the file without the `.sample` extension.
5. Restart the Alfresco server.

The **Repository** link is now available in the Share banner.

Share themes

When you run Alfresco Share, the look and feel is set by a default theme. This section describes how to select one of the alternative themes available in Share, and also how to create and use your own themes for corporate branding.

Share themes consist of a directory containing a CSS and images files, and they can be located in the `theme` directory (`<TOMCAT_HOME>/webapps/share/WEB-INF/classes/alfresco/site-data/themes`). The default theme is called `default.xml`.

The following themes are available in Share:

- Blue theme (default)
- Yellow theme
- Green theme
- High contrast black

The default theme, which comprises the CSS and image assets used across all pages, displays in a new Alfresco installation.

Selecting themes

Only an Administrator user can select the Share theme. Any change to the theme will affect all users of the Alfresco instance from the next time that they log in.

The installation wizards install the default theme and the sample alternative themes. The available themes are installed in the `<configRootShare>/classes/alfresco/site-data/themes` directory.

1. Click **Admin Console** on the toolbar.
2. In the tools list in the browsing pane, click **Application**.
3. Select the required theme from the menu:
   - **Green Theme**
   - **Yellow Theme**
   - **High Contrast Theme**
   - **Default Theme**
4. Click **Apply**.

The new theme displays in Share. The new theme persists across sessions.

Creating a new theme

Additional themes may be defined by creating a new theme directory containing the necessary files, as well as the corresponding XML file, whose name must match that of the theme's directory.

1. Make a new directory within the `/themes` directory.

   Do not include spaces in the directory name.
2. Copy the contents of an existing theme directory to the new theme directory.

For example, copy the `greenTheme` directory.

3. Open the following files:
    a. `base.css`
    b. `ie6.css`
    c. `ie7.css`
    d. `presentation.css`
    e. `yui/assets/skin.css`
4. Specify the new theme by searching for `.yui-skin-greenTheme` and replacing with `.yui-skin-XXX` where `XXX` is the name of the new theme directory.
5. Save the files.

The new theme is then available in the list of themes on the **Application** page in the **Admin Console**.

Editing a theme

A theme consists of some CSS files, an image directory, and a directory for assets for YUI. To create a new look, change the `presentation.css` file and, if required, replace or add images to the `/images` directory.

1. Open the `presentation.css` file.
2. Locate the properties at the end of the `presentation.css` file.
3. Edit the following four properties:
    a. `colour`
    b. `background`
    c. `background-color`
    d. `border`

    Any change to these properties will change the theme.

```
/ Theme colors /
.theme-color-1,
a.theme-color-1,
a.theme-color-1:visited,
a.theme-color-1:hover
{
   color: #6CA5CE;
}

.theme-color-2,
a.theme-color-2,
a.theme-color-2:visited,
a.theme-color-2:hover
{
   color: #038603;
}

.theme-color-3,
a.theme-color-3,
a.theme-color-3:visited,
a.theme-color-3:hover
{
   color: #C7DBEB;
}

.theme-color-4,
a.theme-color-4,
a.theme-color-4:visited,
a.theme-color-4:hover
{
   color: #0D3B57;
}

/ Theme background colors /
 .theme-bg-color-1,
div.theme-bg-color-1
{
   background-color: #6CA5CE;
}

 .theme-bg-color-2,
div.theme-bg-color-2
{
   background-color: #fffbdd;
}

 .theme-bg-color-3,
div.theme-bg-color-3
{
   background-color: #DEE8ED;
}

 .theme-bg-color-4,
div.theme-bg-color-4
{
   background-color: #EBEFF1;
}

.theme-bg-color-5,
div.theme-bg-color-5
{
   background-color: #2B6EB5;
}

 .theme-bg-1
{
   / background-image: url(images/navbar-bg.png); /
}

 .theme-bg-2
{
   / background-image: url(images/navbar-bg-2.png); /
}

 / Theme border type/colors /
 .theme-border-1
{
   border-color: #457f63;
   border-style: dotted;
}

 .theme-border-2
{
   border-color: #2B6EB5;
}
```

4. Locate the `YUI Theme Changes` section.

   This section allows changes to the YUI components.
5. Edit the properties in this section to change the theme.

Forms

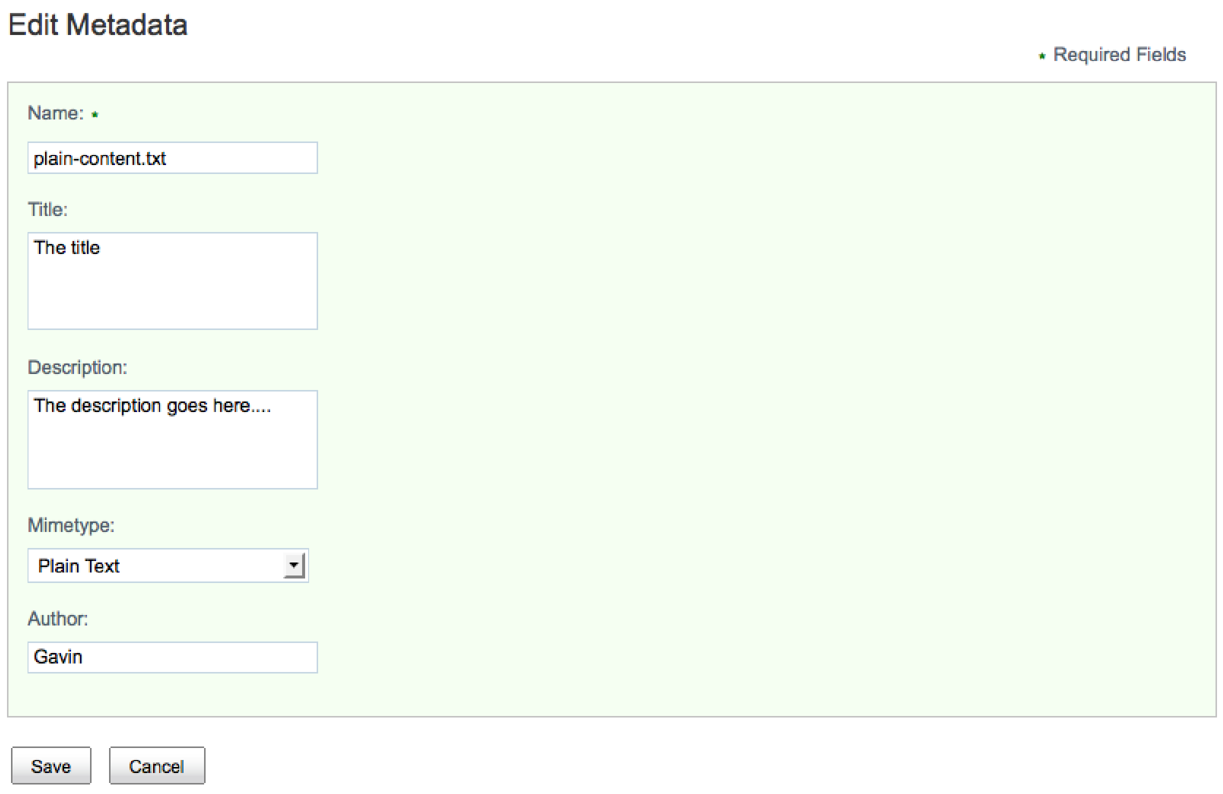
Alfresco Share presents data view and entry forms throughout its user interface, which are built on the Surf framework. This framework provides a convention for implementing forms.

Both DM (Document Management) and WCM (Web Content Management) forms use the same services, meaning that Alfresco uses only one configuration syntax and one set of UI controls for forms.

Use of forms in Share

Forms are used in the **View Metadata** and **Edit Metadata** pages within Share.

The following screen shot shows the form component on the **Edit Metadata** page.



Edit Metadata

* Required Fields

Name: *	plain-content.txt
Title:	The title
Description:	The description goes here....
Mimetype:	Plain Text
Author:	Gavin

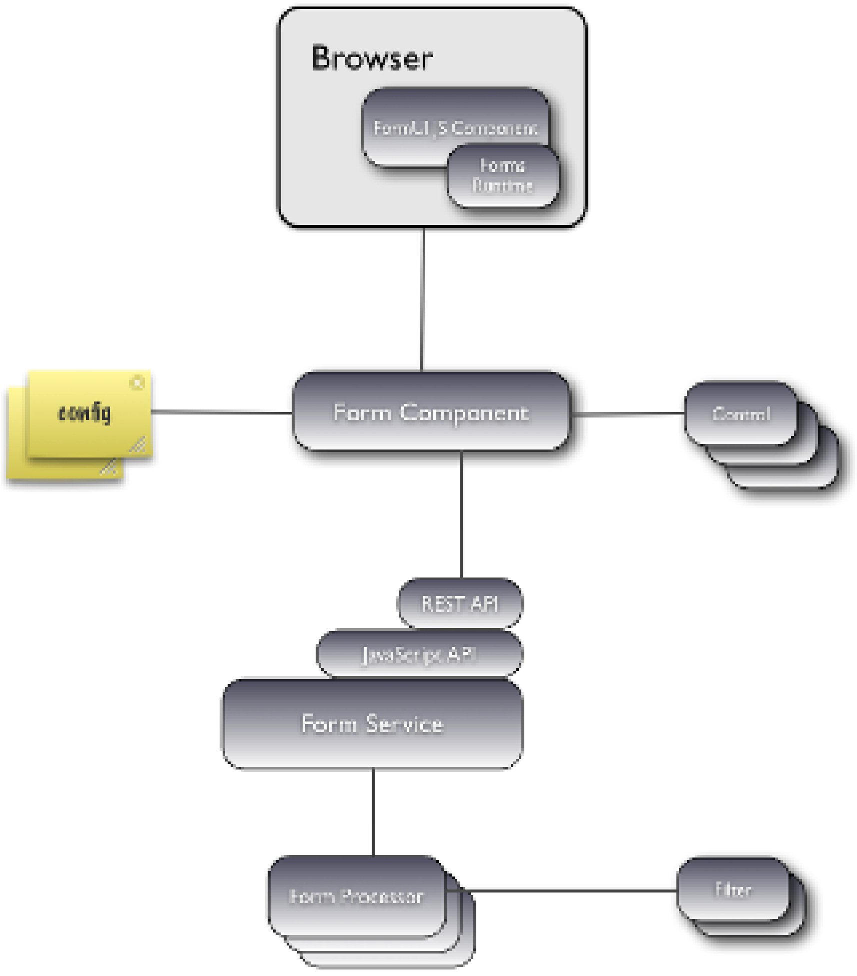
The content of the form is completely driven from configuration custom types, custom aspects. Their properties and associations can be displayed within Share.

Forms architecture

The forms engine consists of four major parts:

- Form service
- Form component
- Form configuration
- JavaScript formUI component, which includes the forms runtime

The following diagram shows a high-level architecture diagram of the forms engine.



The forms runtime is responsible for the execution of a form. It manages all input, validation (client or call-back), events, submission, and it consists of a small, lightweight JavaScript library. An unobtrusive JavaScript pattern is used, where behavior is added to the HTML form elements when the page loads. The forms runtime provides the following capabilities:

- Mandatory property handling
- Validation (enforceable at submission, as the user types or when a field loses focus), which includes:
    - Regular expression matching
    - String length
    - Email address
    - Is number
    - Numeric range
    - Date range
    - List of values
- Repeating fields (for handling multi-valued properties)

Forms event sequence

When a request is made to a page containing the form component, the following sequence of events occurs.

1. The form component looks up the form configuration for the item being requested.
2. The form component retrieves the list of fields to display (if any) and requests a form definition for those fields and the item from the form service via its REST API.
3. The form service looks for a form processor that can process the kind of item.
4. The form processor is asked to generate a form definition.
5. The form processor executes any registered filters before and after the main processing.
6. The REST API takes the result from the form processor/form service and constructs the form definition JSON response.
7. The form component receives the result from the form service and combines it with the form configuration to produce the form UI model.
8. The form component Freemarker template iterates around the fields and includes the relevant controls.
9. The form component Freemarker template instantiates the FormUI JavaScript component.
10. The FormUI JavaScript instantiates the forms runtime and registers all validation handlers.

For a description of the available form controls, refer to Forms reference on page 228

At this point, the form is ready for the user to interact. When the user interacts with the form, the forms runtime constantly checks the validation rules enabling and disabling the **Submit** button appropriately. When the user submits the form, the following sequence of events occurs.

1. The browser submits the form by calling the REST API.
2. The form service looks for a form processor that can process the kind of item.
3. The form processor is asked to persist the form data.
4. The form processor executes any registered filters before and after the main processing.
5. The REST API takes the result from the form processor/form service and constructs the JSON response.
6. The browser receives the response and processes it appropriately.

Configuring forms

The default forms configuration is specified in the `<configRootShare>/classes/alfresco/web-framework-config-commons.xml` file. This file contains all the default controls and constraint handlers for the Alfresco content model and the form configuration for the `cm:content` and `cm:folder` types. This file also contains an example of configuring the `cm:content` type.

You should apply all your forms customizations to a custom configuration file. Several files can be used to hold the configurations, for example:

- `<web-extension>/webscript-framework-config-custom.xml`
- `<web-extension>/web-framework-config-custom.xml`

These files are loaded by Surf and therefore any configuration to the forms will affect all Surf-based applications. To ensure that the configurations you make to forms affect only the Share application, use the custom configuration file named `share-config-custom.xml.sample`.

1. Open the `<web-extension>/share-config-custom.xml.sample` file.
2. Modify the forms configuration settings using the XML configuration syntax.
3. Save the file without the `.sample` extension.

Customizing forms controls

One of the most common customization is to add new controls. A control is the label for a field and the interface that the user interacts with for setting the value of the field.

A control is a Freemarker template snippet that includes the markup to define the control. The model for the template includes the field and form being generated, represented by a `field` and `form` object, respectively. The following snippet shows the structure of the `field` object, using the `cm:name` property as an example:

```
{
   kind : "field",
   id : "prop_cm_name",
   configName : "cm:name",
   name : "prop_cm_name",
   dataType : "d:text",
   type : "property",
   label : "Name",
   description : "Name",
   mandatory : true
   disabled : false,
   repeating : false,
   dataKeyName : "prop_cm_name",
   value : "plain-content.txt",
   control:
   {
      params: {},
      template : "controls/textfield.ftl"
   }
}
```

Although the `id` property provides a unique identifier for the field, it is only scoped to the current form. If there are multiple forms on the page containing the same field, this identifier will not be unique. The model property `fieldHtmlId` should be used as the identifier for the control, as this is guaranteed to be unique for the page.

The state of the `disabled` property must always be adhered to when implementing controls as this is driven from the field definition returned from the FormService and from the `read-only` attribute in the form configuration. If the `disabled` property is set to true, the control should never allow the value to be edited.

The control is also responsible for rendering an appropriate UI representation for the mode the form is currently in. The form mode can be retrieved from the `form.mode` property. A pattern used by most the out-of-the-box controls is shown below.

```
<#if form.mode == "view">
   // view representation goes here...
<#else>
   // edit and create representation goes here...
</#if>
```

The final rule for controls is that they must supply the field current value in a DOM element that has a `value` property and the `id` property set to the value of `fieldHtmlId` Freemarker variable.

For advanced controls, that is, association, date, period, and so on, this usually requires a hidden `form` field.

Customizing the validation handler

A validation handler is a small JavaScript function that gets called by the forms runtime when a field value needs to be validated.

The interface for a validation handler is shown below.

```
/**
 * Validation handler for a field.
 *
 * @param field {object} The element representing the field the validation is
 for
 * @param args {object} Object containing arguments for the handler
 * @param event {object} The event that caused this handler to be called, maybe
 null
 * @param form {object} The forms runtime class instance the field is being
 managed by
 * @param silent {boolean} Determines whether the user should be informed upon
 failure
 * @param message {string} Message to display when validation fails, maybe null
 * @static
 */
function handler-name(field, args, event, form, silent, message)
```

The definition of the built in "mandatory" validation handler is shown below.

```
Alfresco.forms.validation.mandatory = function mandatory(field, args, event,
 form, silent, message)
```

The `field` parameter is usually the HTML DOM element representing the field's value, which is normally an HTML input DOM element, so that the value property can be accessed. The structure of the `args` parameter is dependent on the handler being implemented. By default, these will be the parameters of the constraint defined on the field.

The handler is responsible for taking the value from the field and uses the `args` parameter to calculate whether the current value is valid or not, returning `true` if it is valid and `false` if it is not.

Displaying Type metadata

The configuration to define the fields for the `cm:content` type exists in the system file called `web-framework-config-commons.xml`. Configure the type metadata in the `share-config-custom.xml` file in `<web-extension>`.

The following snippet shows the forms definition in the `share-config-custom.xml` file.

```
<config evaluator="node-type" condition="cm:content">
   <forms>
      <form>
         <field-visibility>
            <show id="cm:name" />
            <show id="cm:title" force="true" />
            <show id="cm:description" force="true" />
            <show id="mimetype" />
            <show id="cm:author" force="true" />
            <show id="size" for-mode="view" />
            <show id="cm:creator" for-mode="view" />
            <show id="cm:created" for-mode="view" />
            <show id="cm:modifier" for-mode="view" />
            <show id="cm:modified" for-mode="view" />
         </field-visibility>
      </form>
   </forms>
</config>
```

The configuration defines that the `cm:name` property is visible in all modes, whereas the `cm:creator`, `cm:created`, `cm:modifier`, and `cm:modified` properties are visible in view mode only.

The `mimetype` and `size` properties are known as transient properties because they do not exist as properties in the model. These properties are formed from the `cm:content` property. The `NodeFormProcessor` knows about these properties and generates a field definition to represent them so that they will appear in the forms.

The `force` attribute ensures that the `NodeFormProcessor` searches the entire Alfresco content model for the property or association definition before returning anything.

1. Open the `<web-extension>/share-config-custom.xml` file.
2. Enter the configuration for custom types.

   The following example configuration shows the `my:text`, `my:dateTime` and `my:association` properties being configured for the custom `my:example` type.

   ```
   <config evaluator="node-type" condition="my:example">
      <forms>
         <form>
            <field-visibility>
               <show id="my:text" />
               <show id="my:dateTime" />
               <show id="my:association" />
            </field-visibility>
         </form>
      </forms>
   </config>
   ```
3. Add more fields to the default configuration.

   The following example shows how to show the node's `DBID` property for all `cm:content` nodes.

   ```
   <config evaluator="node-type" condition="cm:content">
      <forms>
        <form>
           <appearance>
              <field id="cm:description">
                 <control>
                    <control-param name="rows">20</value>
                    <control-param name="columns">20</value>
                 </control>
              </field>
           </appearance>
        </form>
     </forms>
   </config>
   ```

   The full prefix version of the type is required in the `condition` attribute.

   The `force` attribute forces the `NodeFormProcessor` to search the entire Alfresco content model for the property or association definition before returning anything.
4. Save your file.

Displaying aspect metadata

Add the properties and associations defined on aspects by adding them to the list of fields to show for a type. The aspects that appear can be defined on a type by type basis, and you can control the ordering of the fields.

1. Open the `<web-extension>\share-config-custom.xml` file.
2. Enter the configuration for custom types.

   The following example configuration shows the `cm:from` and `cm:to` properties for the `cm:effectivity` aspect.

   ```
   <config evaluator="node-type" condition="cm:content">
      <forms>
         <form>
            <field-visibility>
               <show id="cm:name" />
               <show id="cm:title" force="true" />
               <show id="cm:description" force="true" />
               <show id="mimetype" />
               <show id="cm:author" force="true" />
               <show id="size" for-mode="view" />
               <show id="cm:creator" for-mode="view" />
   ```

```
            <show id="cm:created" for-mode="view" />
            <show id="cm:modifier" for-mode="view" />
            <show id="cm:modified" for-mode="view" />

            <!-- cm:effectivity aspect -->
            <show id="cm:from"/>
            <show id="cm:to"/>
         </field-visibility>
      </form>
   </forms>
</config>
```

3. Add custom aspects to the default configuration by overriding the configuration.

   The following example shows how to add the fields of an example aspect to all forms for the `cm:content` type.

```
 <config evaluator="node-type" condition="cm:content">
   <forms>
      <form>
         <field-visibility>
            <!-- fields from my example aspect -->
            <show id="my:aspectProperty" />
            <show id="my:aspectAssociation" />
         </field-visibility>
      </form>
   </forms>
</config>
```

4. Save the file.

Configuring a form control

Most of the built in controls have parameters, which allow some basic customization.

1. Open the `<web-extension>\share-config-custom.xml` file.
2. Change the number of rows and columns used for the `textarea` control that the `cm:description` field uses by default.

```
<config evaluator="node-type" condition="cm:content">
   <forms>
      <form>
         <appearance>
            <field id="cm:description">
               <control>
                  <control-param name="rows">20</value>
                  <control-param name="columns">80</value>
               </control>
            </field>
         </appearance>
      </form>
   </forms>
</config>
```

3. If all `textarea` controls in the application need to have these settings, configure the control in the `default-controls` element. For example:

```
<config evaluator="node-type" condition="cm:content">
   <forms>
      <default-controls>
         <type name="d:mltext">
            <control-param name="rows">20</control-param>
            <control-param name="columns">80</control-param>
         </type>
      </default-controls>
   </forms>
</config>
```

4. Save the file.

Grouping fields

For longer forms, you can group fields together in logical grouped or nested sections.

1. Open the `<web-extension>\share-config-custom.xml` file.
2. Enter the configuration for custom types.

   The following example configuration shows how to group some fields from an imaginary custom `my:example` type.

```
<config evaluator="model-type" condition="my:example">
   <forms>
      <form>
         <field-visibility>
            <show id="cm:name" />
            <show id="my:text" />
            <show id="my:mltext" />
            <show id="my:boolean" />
            <show id="my:int" />
            <show id="my:long" />
            <show id="my:double" />
            <show id="my:float" />
            <show id="my:status" />
            <show id="my:restricted-string" />
            <show id="my:date" />
            <show id="my:dateTime" />
         </field-visibility>
         <appearance>
            <set id="text" appearance="fieldset" label="Text Fields" />
            <set id="number" appearance="panel" label="Number Fields" />
            <set id="date" appearance="fieldset" label="Date Fields" />

            <field id="cm:name" set="text" />
            <field id="my:text" set="text" />
            <field id="my:mltext" set="text" />
            <field id="my:boolean" set="text" />

            <field id="my:int" set="number" />
            <field id="my:long" set="number" />
            <field id="my:double" set="number" />
            <field id="my:float" set="number" />

            <field id="my:date" set="date" />
            <field id="my:dateTime" set="date" />
         </appearance>
      </form>
   </forms>
</config>
```

   Nested sets are also supported. Use the `parent` attribute in the `set` element. The following example configuration shows the fields of the `my:example` type in a nested set.

```
<config evaluator="model-type" condition="my:example">
   <forms>
      <form>
         <field-visibility>
            <show id="cm:name" />
            <show id="my:text" />
            <show id="my:mltext" />
            <show id="my:boolean" />
            <show id="my:int" />
            <show id="my:long" />
            <show id="my:double" />
            <show id="my:float" />
         </field-visibility>
         <appearance>
            <set id="builtin" appearance="fieldset" label="Built In" />
            <set id="custom" appearance="fieldset" label="Custom Data" />
            <set id="text" parent="custom" appearance="panel"
 label="Text" />
            <set id="number" parent="custom" appearance="panel"
 label="Numbers" />

            <field id="cm:name" set="builtin" />

            <field id="my:text" set="text" />
            <field id="my:mltext" set="text" />
            <field id="my:boolean" set="text" />

            <field id="my:int" set="number" />
            <field id="my:long" set="number" />
            <field id="my:double" set="number" />
            <field id="my:float" set="number" />
         </appearance>
      </form>
   </forms>
</config>
```

3. Save the file.

Changing the default set label

Fields that do not specify a set belong to the implicit `default` set. They are rendered together, by default, but without any visual grouping.

1. Open the `<web-extension>\share-config-custom.xml` file.
2. Enter the configurations for the set label.

   The appearance of the default set can be controlled in the same way as other sets, for example, using an identifier of an empty string.

   ```
   <set id="" appearance="panel" />
   ```

   This will render a panel around all the fields with a label of **Default**.

   To specify a different label, add the `label` attribute. For example, the following label will be **General**.

   ```
   <set id="" appearance="panel" label="General" />
   ```

   You can also use a message bundle key.

   ```
   <set id="" appearance="panel" label-id="form.set.general" />
   ```
3. Save the file.

Providing a custom form control

If none of the out-of-the-box controls are sufficient, you can add new controls and reference them. Controls are Freemarker template snippets, therefore, they contain only the HTML markup required to represent the control. The templates need to be stored in the `site-webscripts` directory, which will usually be in the application server shared classpath.

- The following example configuration shows a very simple custom text field control that always displays with a green background, white text, and 700 pixels wide. For a production system, use a CSS class; however, this example shows a hard coded style.

  ```
  <div class="form-field">
     <#if form.mode == "view">
        <div class="viewmode-field">
           <span class="viewmode-label">${field.label?html}:</span>
           <span class="viewmode-value">${field.value?html}</span>
        </div>
     <#else>
        <label for="${fieldHtmlId}">${field.label?html}:<#if
   field.mandatory><span class="mandatory-indicator">*</span></#if></label>
  ```

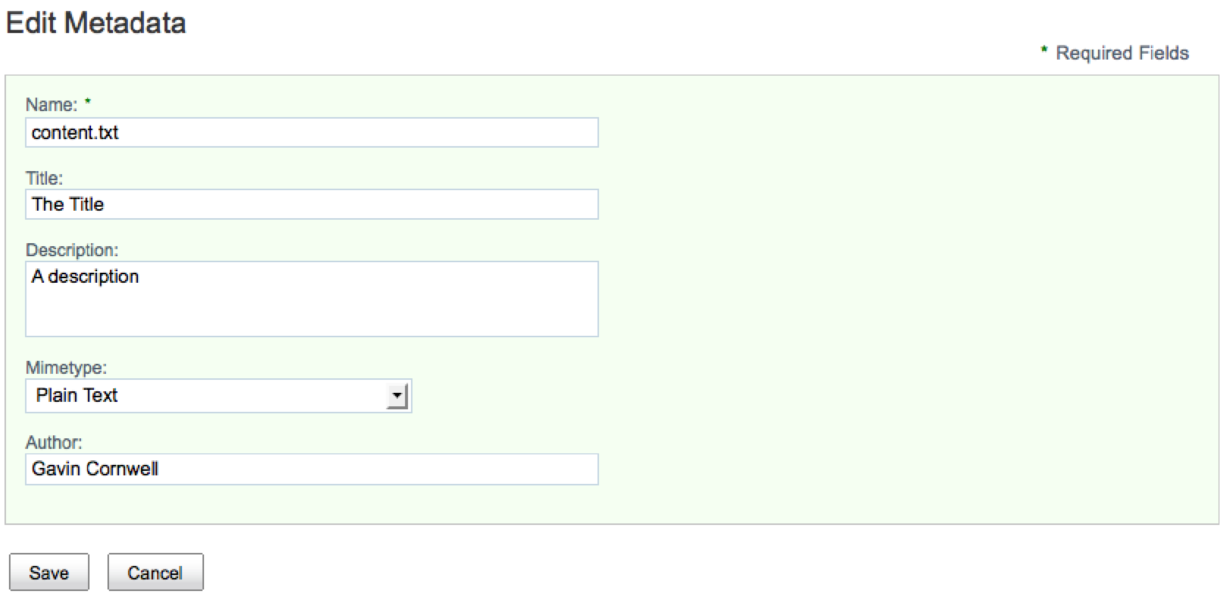
```
      <input id="${fieldHtmlId}" type="text" name="${field.name}"
 value="${field.value}"
                   style="background-color: green; color: white; width:
 700px;" <#if field.disabled>disabled="true"</#if> />
   </#if>
</div>
```

- The following example configuration shows this control being used for the `cm:name` property, with a file name of `my-textfield.ftl`.

```
<config evaluator="node-type" condition="cm:content">
   <forms>
      <form>
         <appearance>
            <field id="cm:name">
               <control template="/my-textfield.ftl" />
            </field>
         </appearance>
      </form>
   </forms>
</config>
```

Changing the field label position

By default, forms are rendered with the field labels positioned above the input control.



Name: * content.txt
Title: The Title
Description: A description
Mimetype: Plain Text
Author: Gavin Cornwell

To change this layout, provide a custom CSS to override the default style rules. Control dependencies can be provided via custom configuration.

1. Add the custom CSS in the `custom-label-layout.css` file, located in the `/custom/forms` directory within the web application.
2. Add the following configuration:

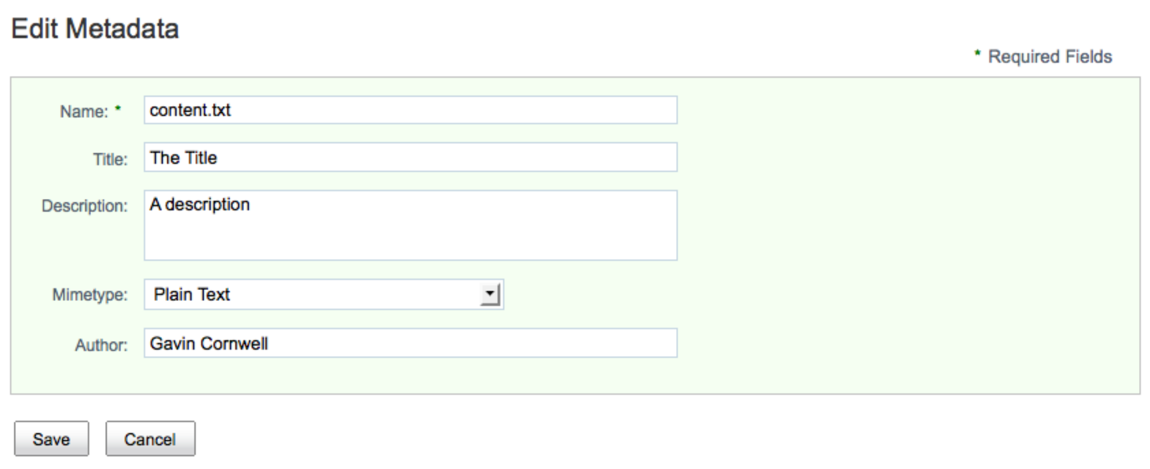
```
<config>
   <forms>
      <dependencies>
         <css src="/custom/forms/custom-label-layout.css" />
      </dependencies>
   </forms>
</config>
```

3. To move the labels to the left of the input control, the following CSS should be present in the `custom-label-layout.css` file:

```
.form-container label
{
   display: inline;
   float: left;
   text-align: right;
   width: 6em;
   margin-right: 1em;
   margin-top: 0.4em;
}
```

4. Save the file.

   The result of this customization is shown as:



Edit Metadata

* Required Fields

Name: *	content.txt
Title:	The Title
Description:	A description
Mimetype:	Plain Text
Author:	Gavin Cornwell

Providing a custom form template

The default template that renders the form UI generates one column of fields. There are scenarios where more control may be required over the layout. To enable these scenarios, it is possible to replace the default template with a custom template. A different template can be provided for each form mode.

Store the custom templates in the `site-webscripts` directory, which is usually be in the application server shared classpath.

1. The example below shows the **Edit** form for the standard `cm:content` type being configured to render with two columns of fields.

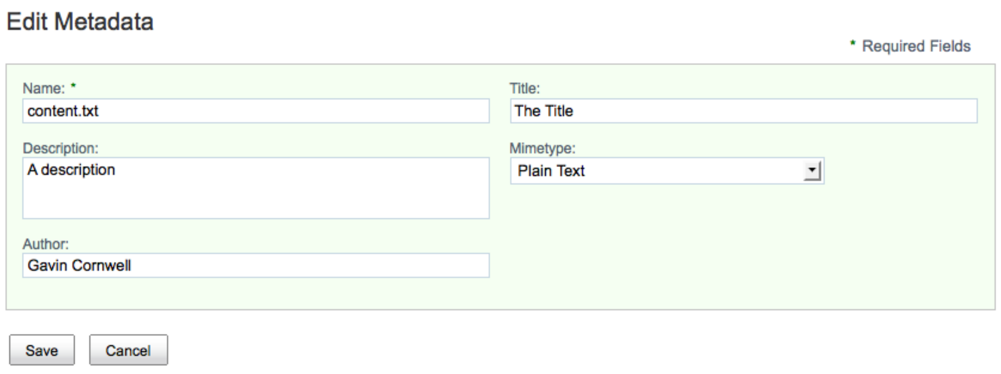
```
<config evaluator="node-type" condition="cm:content">
   <forms>
      <form>
            <edit-form template="/2-column-edit-form.ftl" />
      </form>
   </forms>
</config>
```

   The example template `2-column-edit-form.ftl` is available in the distribution in the `samples` folder.

   The following example shows the contents of the `2-column-edit-form.ftl` file. It uses some of the Freemarker macros available in `form.lib.ftl` but supplies its own `renderSetWithColumns` macro to render the HTML required to create the grid using the YUI grid CSS capabilities.

```
<#import "/org/alfresco/components/form/form.lib.ftl" as formLib />

<#if error?exists>
   <div class="error">${error}</div>
<#elseif form?exists>

   <#assign formId=args.htmlid + "-form">
   <#assign formUI><#if args.formUI??>${args.formUI}<#else>true</#if></
#assign>

   <#if formUI == "true">
      <@formLib.renderFormsRuntime formId=formId />
   </#if>

   <div id="${formId}-container" class="form-container">

      <#if form.showCaption?exists && form.showCaption>
         <div id="${formId}-caption" class="caption"><span
 class="mandatory-indicator">*</span>${msg("form.required.fields")}</div>
      </#if>

      <#if form.mode != "view">
         <form id="${formId}" method="${form.method}" accept-
charset="utf-8" enctype="${form.enctype}" action="${form.submissionUrl}">
      </#if>

      <div id="${formId}-fields" class="form-fields">
        <#list form.items as item>
            <#if item.kind == "set">
               <@renderSetWithColumns set=item />
            <#else>
               <@formLib.renderField field=item />
            </#if>
         </#list>
      </div>

      <#if form.mode != "view">
         <@formLib.renderFormButtons formId=formId />
         </form>
      </#if>

   </div>
</#if>

<#macro renderSetWithColumns set>
   <#if set.appearance?exists>
      <#if set.appearance == "fieldset">
         <fieldset><legend>${set.label}</legend>
      <#elseif set.appearance == "panel">
         <div class="form-panel">
            <div class="form-panel-heading">${set.label}</div>
            <div class="form-panel-body">
      </#if>
   </#if>

   <#list set.children as item>
      <#if item.kind == "set">
         <@renderSetWithColumns set=item />
      <#else>
         <#if (item_index % 2) == 0>
         <div class="yui-g"><div class="yui-u first">
         <#else>
         <div class="yui-u">
         </#if>
         <@formLib.renderField field=item />
         </div>
         <#if ((item_index % 2) != 0) || !item_has_next></div></#if>
      </#if>
   </#list>

   <#if set.appearance?exists>
      <#if set.appearance == "fieldset">
         </fieldset>
      <#elseif set.appearance == "panel">
            </div>
         </div>
      </#if>
   </#if>
</#macro>
```

2. When the configuration and template is in place, the **Edit Metadata** page for a `cm:content` type in Share has the following appearance.



Edit Metadata

* Required Fields

Name: * content.txt

Title: The Title

Description: A description

Mimetype: Plain Text

Author: Gavin Cornwell

Customizing Alfresco Explorer

There are several different ways that you can customize the Explorer configuration items. The Explorer configuration file is called `web-client-config-custom.xml`.

- Modify the Explorer configuration file in the `<extension>` directory.
  a. Open the `<extension>\web-client-config-custom.xml` file.
  b. Uncomment any `<config>` items that you want to enable.
  c. Save the file.
- Modify the Explorer configuration file in the `META-INF` directory of a JAR file.
  a. Open the `META-INF\web-client-config-custom.xml` file.

     For example, on Tomcat, the `META-INF` directory is in `$TOMCAT_HOME\webapps \alfresco\`.
  b. Uncomment any `<config>` items that you want to enable.
  c. Save the file.
  d. Move the custom JAR file into the `WEB-INF\lib` directory.

  Use this method to package the Explorer customizations together with any Java classes you have as part of an extension.
- Modify the Explorer configuration file in the repository.
  a. Browse to the following space: **Company Home > Data Dictionary > Web Client Extensions**.
  b. Open the `web-client-config-custom.xml` file.
  c. Uncomment any `<config>` items that you want to enable.
  d. Save the file.

     Use this method to edit a file directly in the repository and then to reload it using the Explorer. This means that you do not have to restart the server and applies versions to the configuration changes.
- Modify the Explorer configurations within an AMP file.
  a. Open the `module-context.xml` file.

b. Add the following bean definition:

```
 <bean id="myModule_configBootstrap"
 class="org.alfresco.web.config.WebClientConfigBootstrap" init-
method="init">
      <property name="configs">
        <list>
           <value>classpath:alfresco/module/myModuleId/my-web-client-
custom.xml</value>
        </list>
      </property>
   </bean>
```

c. Save the file.

Alfresco Explorer configuration settings

This section describes some of the configuration settings that can be used to modify the behavior of Alfresco Explorer.

The default settings for Explorer are defined in the `<configRoot>\web-client-config.xml` file. To activate the Explorer configurations, add your settings to the `<extension>\web-client-config-custom.xml` file.

1. Open the `<extension>\web-client-config-custom.xml` file.
2. Locate the `<alfresco-config>` section.
3. Add your preferred settings to configure Alfresco Explorer. For example, you can set the following:

Property	Description
`<language-select>false</language-select>`	Set to true to enable automatic language selection from the browser locale.
`<user-group-admin>false</user-group-admin>`	Set to true to remove the administrator action from the Administration Console.
`<allow-user-config>false</allow-user-config>`	Set to true to prevent users from changing their passwords or configuring their person settings.
`<zero-byte-file-upload>false</zero-byte-file-upload>`	Set to true to prevent empty (zero byte) files from being uploaded.
`<breadcrumb-mode>location</breadcrumb-mode>`	Sets the default path-based view to a location-based view. By default, the breadcrumbs in Explorer show the history based on where you have visited. This will show the full path to a space rather than a visit history.
`<edit-link-type>http</edit-link-type>`	Sets the edit link type to use for online editing. The default is inline editable. The options are `http` or `webdav`. Due to heightened security in recent browser versions, it is not advised to use CIFS for online editing. CIFS is not a supported value in the `<edit-link-type>` setting.

For example, the `<alfresco-config>` setting is as follows:

```
<alfresco-config>
   <config>
      <client>
         <!-- pickup language automatically from browser locale -->
         <language-select>false</language-select>
         <!-- do not display User/Group admin console pages -->
         <user-group-admin>false</user-group-admin>
         <!-- do not allow users to configure their person settings or
 change password -->
         <allow-user-config>false</allow-user-config>
         <!-- do not allow empty files to be uploaded -->
         <zero-byte-file-upload>false</zero-byte-file-upload>
         <!-- set location-based view for breadcrumb -->
         <breadcrumb-mode>location</breadcrumb-mode>
      </client>
   </config>
</alfresco-config>
```

4. Modify the configuration settings within the `<alfresco-config>` section.
5. Save the file.

Configuring Web Content Management

This section describes how to configure Web Content Management (WCM).

WCM is an application for managing web content. Content is managed in sandboxes, which isolate each user's changes, and content can be submitted to a shared (staging) sandbox, and then deployed either to a flat file system or another instance of Alfresco. Content is collected and displayed using forms, and workflow controls the approval and publishing of content.

Configuring the virtualization server

WCM preview allows editorial users to preview the website in-context (with the style, markup, and composition of the website), but using the content from any sandbox in the system. Preview provides a way for contributors and reviewers to view changes as if they had been published to the live site, without publishing those changes. This is primarily used for two activities:

- To allow content approvers to review a change they have been asked to approve in the context of the website.
- To allow content contributors to view their changes in the context of the website, while they are in the process of making those changes.

The virtualization server is the default WCM preview functionality when WCM is installed. The virtualization server (which is a slightly modified Tomcat server) interprets data in the AVM repository as a set of virtual websites, allowing users to browse these websites prior to deployment. These websites may include both static sites (or example, `.html`, `.gif`, `.png`) and simple Java-based sites (virtualized servlets and JSPs).

Each Web Project is assumed to be a self contained J2EE web application, and each sandbox in the Web Project is automatically "deployed" as a J2EE web application to the virtualization server. The virtualization server lazily "deploys" each sandbox the first time it receives a preview request, and no physical deployment occurs. The virtualization server has been customized to read the web application directly from the Alfresco repository, rather than from the file system.

To ensure the transparent preview of any sandbox in any Web Project in the system, Alfresco uses dynamic host names (in tandem with wildcard DNS) for accessing the virtualization server. From this enhanced host name, the virtualization server is able to determine from which sandbox and Web Project to read content and web application assets. This ensures that the web application itself has no knowledge of any WCM constructs (sandboxes, Web Projects, virtualization server, and so on); it runs as if it is in a native Tomcat server and the virtualization system ensures that assets are read from the correct sandbox for each preview request.

Each virtual view corresponds to a virtual website within its own virtual host. You can see the virtual view in the URL itself. The following URL shows the general format of an Alfresco virtual hyperlink:

```
http://virtual-hostname.www--sandbox.virtualization-domain:port/request-path
```

To configure virtual websites, the `virtualization-domain` and all its subdomains must resolve in DNS to the IP address of the virtualization server (also known as a “wildcard DNS” mapping). There are two ways to achieve this:

- Use the appropriate subdomain of ip.alfrescodemo.net (see Using ip.alfrescodemo.net on page 140)
- Configure a nameserver, and setting up a wildcard domain pointing at your virtualization server's IP address (see Configuring wildcard DNS on a nameserver on page 141)

Using ip.alfrescodemo.net

To set up virtualization for a single machine or a LAN, use the appropriate subdomain of `ip.alfrescodemo.net`.

In this example, `192.168.1.5` is the IP address of the machine hosting the virtualization server. Alfresco has set up a nameserver at ip.alfrescodemo.net that is able to resolve any domain name of the form `AAA-BBB-CCC-DDD.ip.alfrescodemo.net` (or any of its subdomains) as the IP address `AAA.BBB.CCC.DDD`.

For example, if your browser asks for the virtual host name: `alice.mysite.www--sandbox.192-168-1-5.ip.alfrescodemo.net`, the IP address returned will be: `192.168.1.5`. Therefore, ip.alfrescodemo.net provides "wildcard dns" for all valid IPV4 address. By default, the virtualization server is configured to use the virtualization domain `127-0-0-1.ip.alfrescodemo.net`. This returns the IP address `127.0.0.1`. This special IP address always refers to your local machine (hence its name: `localhost`). Therefore, if you use the default virtualization domain `127-0-0-1.ip.alfrescodemo.net`, you will only be able to do in-context preview on the same machine that hosts the virtualization server. To enable everybody on a LAN to use in-context preview, you need to use a network-accessible IP address.

1. Open the `$VIRTUAL_TOMCAT_HOME/conf/alfresco-virtserver.properties` file.

   Where `$VIRTUAL_TOMCAT_HOME` represents the directory in which you installed the virtualization server.
2. Locate the property `alfresco.virtserver.domain=127-0-0-1.ip.alfrescodemo.net`.
3. Edit the property to contain the hyphen-encoded IP address you want.

   For example: `alfresco.virtserver.domain=192-168-1-5.ip.alfrescodemo.net`.

   When specifying `alfresco.virtserver.domain` that uses `ip.alfrescodemo.net`, the IP address for a DNS wildcard domain must be hyphen-encoded. Otherwise, `ip.alfrescodemo.net` will assume an error, and return `127.0.0.1` for all DNS name lookups within your malformed virtualization domain.

   For example:

   Valid: `alfresco.virtserver.domain=192-168-1-5.ip.alfrescodemo.net`

   Malformed: `alfresco.virtserver.domain=192.168.1.5.ip.alfrescodemo.net`

   Alfresco can now generate URLs that address the virtualization server's network-accessible address, such as `192.168.1.5` and not `127.0.0.1`, allowing users on a LAN to use in-context preview.
4. Restart the Tomcat instance that runs Alfresco.
5. Near WCM assets, click the icon resembling an eyeball.

   A list of URLs displays, for example: `http://<hostname>.www--sandbox.192-168-1-5.ip.alfrescodemo.net`

   These links will resolve to your virtualization server on `192.168.1.5`.
6. To view this in isolation, run one of the following commands:

- `ping alice.mysite.www--sandbox.192-168-1-5.ip.alfrescodemo.net`
- `ping mysite.www--sandbox.19`

Configuring wildcard DNS on a nameserver

This task describes how to configure a nameserver, and set up a wildcard domain pointing at your virtualization server machine's IP address to:

- Virtualize content on a machine or LAN with no access to the Internet (for example: a demo laptop)
- Attain faster speeds on the first DNS lookup of a name than when served locally (IP address answers have a one day TTL)
- Keep everything centralized if you already have BIND, djbdns, Microsoft DNS, or other DNS solution in place
- Be independent from the Alfresco alfrescodemo.net uptime

1. Set the directory where you installed the virtualization server, for example: `$VIRTUAL_TOMCAT_HOME`.

   In this example, the LAN's domain name is `localdomain.lan`, a wildcard DNS record resolves to an IP address such as `192.168.1.5`, and the wildcard DNS names are within `ip.localdomain.lan`.
2. Open the file `$VIRTUAL_TOMCAT_HOME/conf/alfresco-virtserver.properties`.
3. Modify the value of the `alfresco.virtserver.domain` property to resemble the following:

   ```
   alfresco.virtserver.domain=ip.localdomain.lan
   ```

   You cannot use the “hosts” file to create a wildcard DNS domain on Windows or Unix. To create a DNS wildcard that is usable when you are not connected to the Internet, you must install, configure, and maintain a real nameserver.

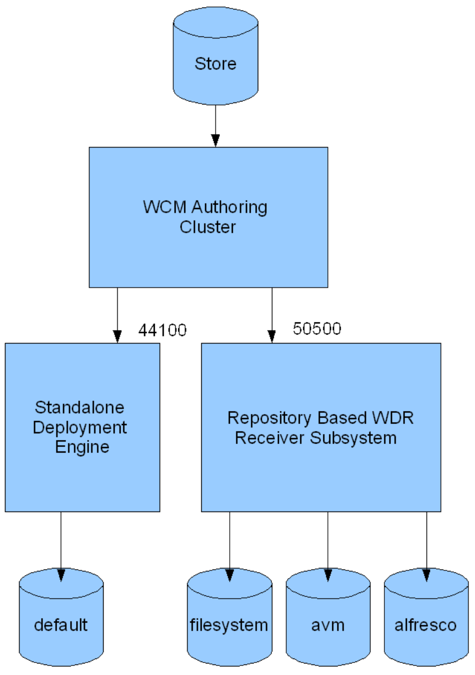
WCM deployment

WCM deployment provides a framework for pushing content from an Alfresco WCM authoring environment into another environment. For example you can push out content to a flat file system, to be served up by Apache or IIS, or to another instance of Alfresco.

The WCM authoring environment provides the facilities to produce an approved and consistent view of a web project called a snapshot. WCM deployment takes those snapshots and pushes them out to either live or test servers. The WCM Deployment Engine receives deployments from an Alfresco WCM authoring environment and delegates the contents of the deployment to the appropriate deployment receiver target.

There are two implementations of the WCM Deployment Engine:

- Standalone deployment engine, which is a small lightweight framework that is independent of the Alfresco repository
- Deployment engine, which is a subsystem within the Alfresco repository


Store
WCM Authoring Cluster
44100
50500
Standalone Deployment Engine
Repository Based WDR Receiver Subsystem
default
filesystem
avm
alfresco

WCM deployments are sent to an Alfresco deployment engine, which delegates the deployment to the registered deployment targets. Deployment targets are registered using the Web Project wizard.

Deployment targets

A deployment target is a named destination for a WCM deployment. WCM will deploy through a deployment engine to a deployment target.

There are four deployment targets defined.

default
The file system deployment target for the Standalone Deployment Engine.

filesystem
The file system deployment target for the Web Delivery Runtime (WDR) Deployment Engine.

avm
The AVM deployment target for the Web Delivery Runtime (WDR) Deployment Engine.

alfresco
The DM deployment target for the Web Delivery Runtime (WDR) Deployment Engine.

You also can add your own deployment targets.

WCM standalone receiver

The Standalone Deployment Engine consists of a small server that receives updates from an Alfresco repository and publishes them to its deployment targets. It does not have the benefits of a full repository but is a small and tightly focused server.

It is configured with a single file system deployment target, which places the contents of the deployment onto a flat file system, and is typically served up by a web or application server. For performance reasons, only the difference between the old version and the new version is sent.

The destination file server receiver has to be running with its RMI registry port and service port (by default, 44100 and 44101, respectively) accessible, unless you have configured it to run over another transport.

Configuring the WCM standalone deployment receiver

This section describes how to configure the WCM standalone deployment receiver. The configuration files needed for configuration are the `deployment.properties` file and the `application-context.xml` file.

Ensure that you have installed the WCM standalone deployment receiver. See Installing the WCM standalone deployment receiver on page 30.

1. Locate the `deployment` installation directory, for example:
    - (Windows) `c:/Alfresco/Deployment`
    - (Linux) `/opt/alfresco/deployment`

   The `deployment` directory contains deployment target definitions and plug in definitions. Your spring definitions of the deployment targets are read from this directory. All files matching the pattern `deployment/*-context.xml` and `deployment/*-target.xml` are loaded by the deployment engine. Targets are loaded after the context files, so each target has all spring beans available when it is defined.
2. Open the `deployment.properties` file.

   The `deployment.properties` file contains a simple configuration for the standalone deployment receiver. The file is created by the deployment project **deployer**.
3. Edit the configurations as follows:

```
; filesystem receiver configuration
deployment.filesystem.datadir=depdata
deployment.filesystem.logdir=deplog
deployment.filesystem.metadatadir=depmetadata
deployment.filesystem.autofix=true
deployment.filesystem.errorOnOverwrite=false

; default filesystem target configuration
deployment.filesystem.default.rootdir=target
deployment.filesystem.default.name=default

; Deployment Engine configuration
deployment.rmi.port=44100
deployment.rmi.service.port=44101

; Stand alone deployment server specific properties
deployment.user=admin
deployment.password=admin
```

4. Save your file.
5. Restart the deployment engine.

The following is a sample `application-context.xml` file and it shows how the `deployment.properties`, context files, and the targets are read.

```
<beans>
   <bean id="properties"

 class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
     <property name="ignoreUnresolvablePlaceholders">
          <value>true</value>
     </property>
        <property name="locations">
           <list>
              <value>classpath:deployment.properties</value>
```

```
            </list>
        </property>
    </bean>

     <import resource="classpath*:deployment/*-context.xml" />
     <import resource="classpath*:deployment/*-target.xml" />

</beans>
```

The standalone deployment engine uses `log4j` for problem determination, and logging for the deployment engine is placed in the `log` directory.

Your spring definitions of the deployment targets are read from this directory. All files matching the pattern `deployment/*-target.xml` are loaded by the deployment engine.

Targets are loaded after the context files. So each target has all spring beans available to it when it is defined

Configuring the standalone deployment receiver as a Windows service

Solution Steps for Windows 32 bit - Alfresco 32r Standalone Deployment Receiver (New name for FSR)

1. Download Java Wrapper from following URL.

   `http://wrapper.tanukisoftware.org/doc/e ... wnload.jsp`

   For example, Community Version for 32 bit windows.
2. Unzip the wrapper to a folder.

   For example, `C:\wrapper`.
3. Download the Alfresco Standalone Deployment Receiver.
4. Copy all the JAR files from the Standalone Deployment Receiver to the `wrapper\lib` directory.
5. Create a empty JAR file called `deployment-config.jar`.
6. Add the `deployment.properties` file and the deployment folder with all of your deployment target XML files from the Standalone Deployment Receiver to the new JAR file.
7. Copy the JAR file to the `wrapper\lib` directory.
8. Copy following files from Standalone Deployment Receiver to the `wrapper\bin` directory.
   a. `application-context.xml`
   b. `shutdown-context.xml`
9. Copy the following files from the `wrapper\src\bin` directory to the `wrapper\bin` directory, and then remove `.in` from file name to make it batch file.
   a. `InstallApp-NT.bat.in` (after rename it will be `InstallApp-NT.bat`)
   b. `UninstallApp-NT.bat.in` (after rename it will be `UninstallApp-NT.bat`
10. Open the `wrapper/conf/wrapper.conf` file and make following changes:
    a. Change `wrapper.java.command` to `%JAVA_HOME%/bin/java`.
    b. Change `wrapper.java.mainclass` to `org.tanukisoftware.wrapper.WrapperStartStopApp`.
11. Add following classpath entries. Remove any default entries for classpath:
    a. `wrapper.java.classpath.1=../lib/wrapper.jar`
    b. `wrapper.java.classpath.2=%JAVA_HOME%/lib/tools.jar`
    c. `wrapper.java.classpath.4=../lib/alfresco-deployment-3.2.0r.jar`

   d. `wrapper.java.classpath.5=../lib/alfresco-core-3.2.0r.jar`
   e. `wrapper.java.classpath.6=../lib/alfresco-repository-3.2.0r.jar`
   f. `wrapper.java.classpath.7=../lib/commons-logging-1.1.jar`
   g. `wrapper.java.classpath.8=../lib/jug-lgpl-2.0.0.jar`
   h. `wrapper.java.classpath.9=../lib/log4j-1.2.15.jar`
   i. `wrapper.java.classpath.10=../lib/spring-2.0.8.jar`
   j. `wrapper.java.classpath.11=../lib/deployment-config.jar`
   k. `#wrapper.java.classpath.10=../lib/deployment.properties`
   l. `#wrapper.java.classpath.11=../lib/shutdown-context.xml`
   m. `#wrapper.java.classpath.12=../lib/application-context.xml`

12. Add following additions to parameters:
    a. wrapper.java.additional.2=-Dcatalina.base=..
    b. wrapper.java.additional.3=-Dcatalina.home=..
    c. wrapper.java.additional.4=-Djava.io.tmpdir=../temp
13. Add following Application Parameters:
    a. wrapper.app.parameter.1=org.alfresco.deployment.Main
    b. wrapper.app.parameter.2=1
    c. wrapper.app.parameter.3=application-context.xml
    d. wrapper.app.parameter.4=org.alfresco.deployment.Main
    e. wrapper.app.parameter.5=TRUE
    f. wrapper.app.parameter.6=1
    g. wrapper.app.parameter.7=shutdown-context.xml
14. Change service name related changes as per below:
    a. Name of the service.
    b. ii. wrapper.name=Alfresco Standalone Deployment Receiver
    c. # Display name of the service
    d. wrapper.displayname=Alfresco Standalone Deployment Receiver
    e. Description of the service
    f. wrapper.description=Alfresco Standalone Deployment Receiver
15. Click the `wrapper\bin\InstallApp-NT.bat` file. You should be able to see service name "Alfresco Standalone Deployment Receiver". For the first time, you need to start the service manually.

Check service log in the `wrapper\log\wrapper.log` file.

Alfresco Web Editor

The Alfresco Web Editor is a Spring Surf-based web application that provides in-context editing capabilities for Alfresco repository content. The editor provides a mechanism for non-technical users to make edits to Alfresco content directly within a web page.

The Alfresco Web Editor uses the Forms Service default template.

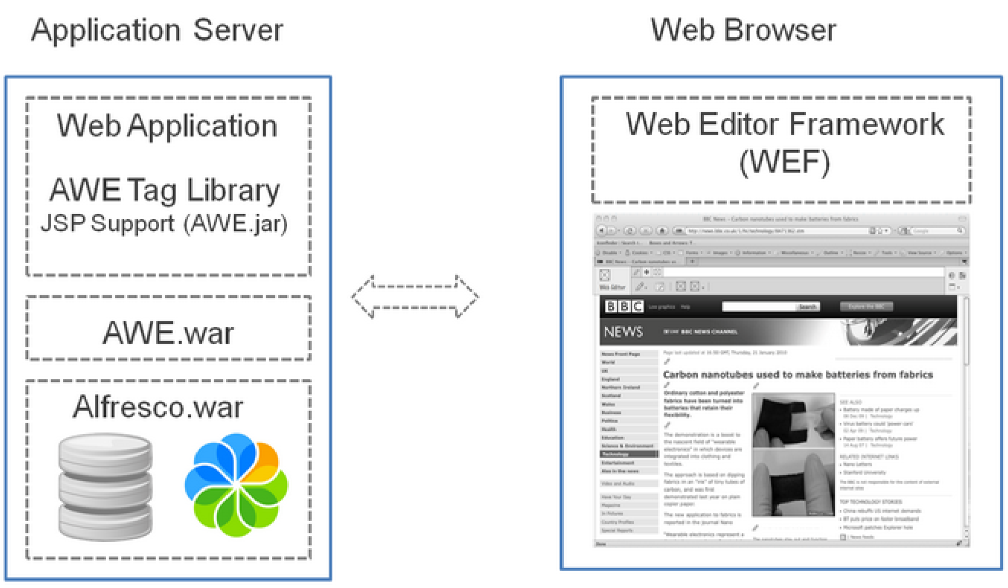
The Alfresco Web Editor is packaged as a stand-alone WAR file so that it can be deployed to web applications that are in the sample instance, or remote, to the Alfresco server. When it is deployed, an Alfresco banner displays in your deployed web pages showing the Alfresco Web

Editor tab and identifies the editable content. By default, it assumes that you have JavaScript enabled but it can also run without JavaScript.

Alfresco Web Editor deployment overview

The simplest way to deploy the Alfresco Web Editor is to use the pre-built WAR (`awe.war`) file and to deploy it in the same application server instance of your web application.

The following diagram shows an example Alfresco Web Editor deployment in the same application server as the Alfresco repository.



The Alfresco Web Editor is a Spring Surf-based application, therefore it is also possible to deploy it in a different application server instance from the Alfresco repository.

The deployment comprises the following components:

AWE.war
: The Alfresco Web Editor WAR file.

Web Application
: Your own web application.

AWE tag library
: Provides the ability to mark areas of the page as editable. The areas marked can represent any property or content from the Alfresco repository.

Web Editor Framework (WEF)
: The client-side JavaScript framework on which the Web Editor is built. It is built using YUI and can be extended easily. New tabs and buttons can be packaged and dropped into the framework. This provides the core Alfresco product features, and also provides the ability to build additional custom plugins.

  When the Alfresco Web Editor is enabled, the WEF renders the tool bar and basic in-context editing buttons and functionality. If the WEF is deployed as standalone, the default blank tool bar is rendered.

Deploying the Alfresco Web Editor

The Alfresco Web Editor distribution consists of a single zip file named `alfresco-community-webeditor-3.3.zip`.

1. Shut down your Alfresco server.
2. Download the `alfresco-community-webeditor-3.3.zip` file.
3. Deploy the `awe.war` file into the same application server instance as the Alfresco repository.
4. Copy the `alfresco-webeditor-taglib-3.3.jar` file to the `WEB-INF/lib` folder of your application.
5. To include the tag library in your application, add the following tag library declaration to your JSP page:

   ```
   <%@ taglib uri="http://www.alfresco.org/tags/awe" prefix="awe" %>
   ```

   Once the tag library is declared, you can use the startTemplate, endTemplate and markContent tags within your application.
6. Restart your Alfresco server.

Deploying the Alfresco Web Editor to a Spring Surf application

The Alfresco Web Editor distribution also includes all the files required to provide the functionality within an existing Spring Surf application.

1. Copy the following files to your application `WEB-INF/lib` directory:
   a. `yui-2.7.0.jar`
   b. `spring-webeditor-1.0.0.CI-SNAPSHOT.jar`
   c. `alfresco-forms-client-3.3.jar`
   d. `alfresco-webeditor-plugin-3.3.jar`

   The `yui` and `spring-webeditor` JAR files represent the Web Editor Framework (WEF) upon which the Web Editor is built. The remaining `alfresco-form-client` and `alfresco-webeditor-plugin` JAR files provide the Web Editor functionality.
2. If you plan to use the Web Editor within the application (rather than the application being a host for the Web Editor services) you also must copy the following additional files into the `WEB-INF/lib` directory:
   a. `spring-webeditor-client-jsp-1.0.0.CI-SNAPSHOT.jar`
   b. `alfresco-webeditor-taglib-3.3.jar`
3. If you use the additional files, define a servlet filter in your application's `web.xml` file.

   If you do not provide the filter, the tags will be ignored. The following filter configuration is required:

   ```
   <filter>
       <filter-name>Alfresco Web Editor Filter</filter-name>
       <description>Enables support for the Alfresco Web Editor</
   description>
       <filter-class>org.alfresco.web.awe.filter.WebEditorFilter</filter-
   class>
       <init-param>
          <param-name>contextPath</param-name>
          <param-value>/your-context-path</param-value>
       </init-param>
   </filter>

   <filter-mapping>
      <filter-name>Alfresco Web Editor Filter</filter-name>
      <url-pattern>/*</url-pattern>
   </filter-mapping>
   ```
4. Set the `contextPath` parameter.

If you do not provided a value for this parameter, a default `contextPath` of `/awe` is presumed.

No further configuration is required as all the necessary Spring context files and Alfresco configuration files are contained within the JAR files. However, there is no default hook point for custom form configuration but this can be located anywhere within your application.

Alfresco Web Editor configuration

The following Web Editor components must be configured:

- tag library, that is, the `markContent` tag used to define editable content
- servlet filter
- form configuration

Configuring the tag library

The tag library comprises the following tags:

- `startTemplate`
- `markContent`
- `endTemplate`

1. The `startTemplate` tag bootstraps the WEF using a script element that executes a web script. Place this tag in the `head` section of your page.

   The `startTemplate` tag has only one optional attribute.

   toolbarLocation
   Controls the initial location of the tool bar. The valid values are: `top`, `left`, and `right`. The default is `top`.

   The following shows an example of how to use the `startTemplate` tag:

   ```
   <awe:startTemplate toolbarLocation="top" />
   ```

2. Use the `markContent` tag to indicate an editable area of the page.

   The tag renders an edit icon that, when clicked, displays a form for editing the corresponding Alfresco content and properties, or both.

   The `markContent` tag has two mandatory attributes and two optional attributes.

   id
   The mandatory id attribute specifies the NodeRef of the Alfresco node to be edited.

   title
   The mandatory title attribute defines a descriptive title for the editable area being marked. The title used will be used in the quick edit drop down menu of editable items, as the title of form edit popup/dialog and the 'alt' text and tooltip text of the edit icon.

   formId
   The optional formId attribute specifies which form will be used when the marked area is edited. See the Form Configuration section below for more details.

   nestedMarker
   The optional nestedMarker attribute defines whether the editable area is nested within another HTML tag that represents the content being edited. If set to "true" the whole parent element is highlighted when the area is selected in the quick edit drop down menu. If set to "false" only the edit icon is highlighted.

An example use of the markContent tag is shown below.

```
<awe:markContent id="<%=subTextNodeRef%>" formId="description"
 title="Edit Description" nestedMarker="true" />
```

3. The `endTemplate` tag initializes the Web Editor with details of all the marked content areas on the page. It also renders a script element that executes the WEF resources web script, which starts the process of downloading all the assets required to render and display the tool bar and all configured plugins. Place this tag just before the closing body element.

   The `endTemplate` tag does not have any attributes.

   The following shows an example of how to use the `endTemplate` tag:

   ```
   <awe:endTemplate />
   ```

Configuring the servlet filter

The `startTemplate`, `markContent`, and `endTemplate` tags will only render their output if they detect the presence of the Web Editor servlet filter. The tags can remain in the JSP page in production and have no effect until the servlet filter configuration is added to the `web.xml` file.

1. Add the following servlet filter configuration to the web application's `web.xml` file:

   ```
   <filter>
      <filter-name>Alfresco Web Editor Filter</filter-name>
      <description>Enables support for the Alfresco Web Editor</description>
      <filter-class>org.alfresco.web.awe.filter.WebEditorFilter</filter-
   class>
   </filter>

   <filter-mapping>
      <filter-name>Alfresco Web Editor Filter</filter-name>
      <url-pattern>/*</url-pattern>
   </filter-mapping>
   ```

   This enables the tags.

2. Set the following two optional parameters:

   ```
   <init-param>
      <param-name>contextPath</param-name>
      <param-value>/quickstart</param-value>
   </init-param>

   <init-param>
      <param-name>debug</param-name>
      <param-value>true</param-value>
   </init-param>
   ```

   These parameters control the `contextPath` that is used when URLs to the Web Editor are generated and the debug mode.

Configuring Web Editor forms

The Alfresco Web Editor uses a form to edit the node referenced by a `markContent` tag. By default, the form displayed will contain the `cm:title`, `cm:description`, and `cm:content` fields. An alternative form can be used by providing the `markContent` tag with a `formId` attribute.

Out of the box, only two other forms are configured: a form with an identifier of `title`, and one with an identifier of `description`. As the identifiers indicate, the forms display a single property: `cm:title` and `cm:description`, respectively. The node type is presumed to be `cm:content`.

If you have custom types or wish to specify other properties, you can use the the forms configuration techniques.

Sample web application using Alfresco Web Editor

A sample customer WAR file is available in the Alfresco Web Editor distribution. It demonstrates how a customer may use Alfresco Web Editor in a very simple JSP-based web application. This sample must not be used in a production environment and is not supported.

A sample customer tag library is provided, which includes two tags. These tags are included as a demonstration sample and should never be used in a production environment.

`content`
: Allows content to be pulled from an Alfresco repository and sends output to a JSP page. The `content` tag requires one mandatory attribute called `nodeRef`

`property`
: Allows properties to be pulled from an Alfresco repository and sends output to a JSP page. The `property` tag requires two mandatory attributes: `nodeRef` and `property`.

The following example show the use of these tags:

```
<customer:content nodeRef="<%=mainTextNodeRef%>" />
<customer:property nodeRef="<%=subTextNodeRef%>" property="cm:description" />
```

The sample customer application consists of several, simple JSP pages that display the content and properties of two nodes from the repository. Update the `/includes/noderefs.jsp` page to provide the NodeRefs of two nodes in your repository.

By default, the sample pulls content from the Alfresco repository located at `http://localhost:8080/alfresco`, using a user name and password of `admin`. These values can be supplied using `context-param` values in the `web.xml` file, for example:

```
<context-param>
      <param-name>org.customer.alfresco.host</param-name>
      <param-value>localhost</param-value>
   </context-param>

   <context-param>
      <param-name>org.customer.alfresco.port</param-name>
      <param-value>8080</param-value>
   </context-param>

   <context-param>
      <param-name>org.customer.alfresco.context</param-name>
      <param-value>alfresco</param-value>
   </context-param>

   <context-param>
      <param-name>org.customer.alfresco.username</param-name>
      <param-value>admin</param-value>
   </context-param>

   <context-param>
      <param-name>org.customer.alfresco.password</param-name>
      <param-value>admin</param-value>
   </context-param>
```

Administering Alfresco

This chapter provides information for maintaining and managing the Alfresco system. The following topics are covered:

Share Admin Console

The Admin Console enables Alfresco Administrators to create and manage both users and groups from within Share. It also provides the functionality for setting application preferences. Only those users who are members of the group ALFRESCO_ADMINISTRATORS have access to the Admin Console.

Specifying application preferences

The Application tool in the Admin Console browsing pane enables you to set application preferences.

Selecting a theme

The Application feature enables you to select a color scheme for the Share user interface.

The change affects all users of the Share instance from the next time they log in. The selected theme persists across sessions.

1. Access the Admin Console and click **Application** in the browsing pane.

   The **Options** page appears.
2. Select the desired theme from the list provided.
3. Click **Apply**.

The page refreshes to display with the selected theme.

Managing users

The Users tool in the Admin Console browsing pane enables you to create and manage the Share user accounts.

Searching for and viewing a user account

The User Search feature enables you to locate any Share user and view that user’s account information.

1. Access the Admin Console and click **Users** in the browsing pane.

   The **User Search** page appears.
2. In the search box, type the full or partial name of the desired user.

   You must enter a minimum of one (1) character. The search is not case sensitive.
3. Click **Search**.

   In the results table, click the column headings to sort the results as desired.

   In the first column, a green dot indicates the user account is currently enabled; a red dot indicates the account is disabled.
4. Click the name of the desired user to display the related user profile and account details.

The **User Profile** page displays. From here you can edit or delete the user account.

Creating a user account

The Admin Console enables you to easily create users accounts.

1. Access the Admin Console and click **Users** in the browsing pane.

   The **User Search** page appears.
2. Click **New User**.

   The **New User** page appears. Fields marked with an asterisk (*) are required.
3. Complete the required user details.
4. If desired, add the user to existing user groups:
   a. In the search box, type the full or partial name of the desired group.

      You must enter a minimum of one (1) character. The search is not case sensitive.

   b. Click **Search**.

   c. In the list of returned results, click **Add** to the right of each group you want the user to be a part of.

      The groups appear beneath the **Groups** list. Click a group to remove it.

   d. Perform additional searches as necessary to locate and add more groups.
5. In the **Quota** box, specify the maximum space available for this user and select the appropriate unit (GB, MB, or KB).

   This information is not required. When no quota is provided, the user has no space limitations.
6. Click **Create User**.

   If you intend to immediately create another user, click **Create and Create Another**. This creates the user account specified and clears the fields without returning you to the **User Search** page.

Editing a user account

Edit a user account to change a user’s personal information, group affiliation, quota, and password.

1. Access the Admin Console and click **Users** in the browsing pane.

   The **User Search** page appears.
2. Search for and select the desired user.
3. On the **User Profile** page, click **Edit User**.

   The **Edit User** page appears.

4. Edit the user's personal details as necessary: **First Name**, **Last Name**, and **Email**.
5. Edit the groups to which this user belongs:
    a. To add a user to a group, use the search field provided to locate the desired group. You must enter a minimum of one (1) character. Click **Add** to the right of each group you want the user to be a part of. The groups the user belongs to display beneath the **Groups** list.
    b. To remove a user from a group, simply click the group you want to remove beneath the **Groups** list.
6. Provide or edit the **Quota**, which indicates the maximum space available for this user. Select the appropriate unit.
7. Change the password, if necessary.
8. Click **Use Default** to reset the user's picture to the default image.
9. Click **Save Changes**.

Deleting a user account

Delete a user account to remove the user from the system.

To retain the user account while denying the user access to the application, consider simply disabling the user account.

1. Access the Admin Console and click **Users** in the browsing pane.

   The **User Search** page appears.
2. Search for and select the desired user.
3. On the **User Profile** page, click **Delete User**.

   A message prompts you to confirm the deletion.
4. Click **Delete**.

Disabling a user account

Disable a user account to prevent a user from having any access to the application. You perform this task as part of editing a user account.

1. Access the Admin Console and click **Users** in the browsing pane.

   The **User Search** page appears.
2. Search for and select the desired user.
3. On the **User Profile** page, click **Edit User**.

   The **Edit User** page appears.
4. Click **Disable Account**.

   A checkmark indicates the account for the current user will be disabled.
5. Click **Save Changes**.

   On the **User Profile** page, the Account Status appears as **Disabled**. On the **User Search** page, the user displays in the search results list with a red dot, indicating the account is disabled.

Changing a user's password

You can change a user's password as part of editing the user account.

1. Access the Admin Console and click **Users** in the browsing pane.

   The **User Search** page appears.
2. Search for and select the desired user.

3. On the **User Profile** page, click **Edit User**.

   The **Edit User** page appears.
4. Enter and confirm the new password for this user in the **New Password** and **Verify Password** boxes.

   The password is case sensitive.
5. Click **Save Changes**.

Managing the user's group membership

Within a user account, you have the opportunity to manage the user’s membership in existing user groups. You can edit a use account at any time to add and remove the user from groups.

1. Access the Admin Console and click **Users** in the browsing pane.

   The **User Search** page appears.
2. Search for and select the desired user.
3. On the **User Profile** page, click **Edit User**.

   The **Edit User** page appears.
4. Edit the groups to which this user belongs:
   a. To add a user to a group, use the search field provided to locate the desired group. You must enter a minimum of one (1) character. Click **Add** to the right of each group you want the user to be a part of. The groups the user belongs to display beneath the **Groups** list.
   b. To remove a user from a group, simply click the group you want to remove beneath the **Groups** list.
5. Click **Save Changes**.

Managing groups

The Groups tool in the Admin Console browsing pane enables you to create and manage Share user groups.

Browsing the user groups

The Groups page contains a multi-paned panel that allows you to navigate the hierarchy of Share user groups.

1. Access the Admin Console and click **Groups** in the browsing pane.
2. On the **Groups** page, click **Browse**.

   The leftmost pane displays all top-level user groups.
3. Click a group to display its contents in the panel directly to the right.

   The content can be subgroups and/or individual users. Text at the bottom of this panel indicates the number of groups and users that belong to the selected group.
4. As you browse the group structure, a navigation path is displayed at the top of the panel indicating your selections stemming from the initial pane. Click any link in this path to step back to that selection.
5. To browse a different group, click the first link in the navigation path to return to the top-level groups, then select a new group to browse.

Searching for a group

The Search feature enables you to locate any Share user group, regardless of where it exists in the group hierarchy. Once located, you can edit or delete the group.

1. Access the Admin Console and click **Groups** in the browsing pane.
2. In the search box, type the full or partial identifier, not display name, of the desired group.

   You must enter a minimum of one (1) character. The search is not case sensitive.
3. Click **Search**.

   In the results table, click the column headings to sort the results as desired.

Creating a new group

The Admin Console enables you to create both top level user groups and subgroups within existing groups.

1. Access the Admin Console and click **Groups** in the browsing pane.
2. On the **Groups** page, click **Browse**.

   The leftmost pane displays all top-level user groups.
3. Navigate to the user group where you want to create the new group.

   - To create a top-level group, click the **New Group** icon at the top of the initial pane.
   - To create a subgroup, browse the group structure to locate the desired parent group. Select this group and then click the **New Subgroup** icon at the top of the pane immediately to the right.

   The **New Group** page appears. Fields marked with an asterisk (*) are required.
4. Complete the required user details.
5. Click **Create Group**.

   If you intend to immediately create another group at the same level, click **Create and Create Another**. This creates the group specified and clears the fields without returning you to the **Groups** page.

Editing an existing group

Edit a user group to change the group’s display name. Once created, you cannot edit the group’s Identifier.

1. Access the Admin Console and click **Groups** in the browsing pane.
2. On the **Groups** page, click **Browse**.

   The leftmost pane displays all top-level user groups.
3. Navigate the group structure or use the search feature to locate the user group you want to edit.

   You must enter a minimum of one (1) character. The search is not case sensitive.
4. Position the cursor over the desired group to display its available actions.
5. Click the **Edit Group** icon.
6. Edit the group’s **Display Name**.
7. Click **Save Changes**.

Deleting an existing group

Delete a user group to remove it from the system.

1. Access the Admin Console and click **Groups** in the browsing pane.
2. On the **Groups** page, click **Browse**.

   The leftmost pane displays all top-level user groups.

3. Navigate the group structure or use the search feature to locate the user group you want to delete.

   You must enter a minimum of one (1) character. The search is not case sensitive.
4. Position the cursor over the desired group to display its available actions.
5. Click the **Delete Group** icon.

   A message prompts you to confirm the deletion.
6. Click **Delete**.

Managing group membership

To populate a user group, you can add both individual users and existing user groups.

1. Access the Admin Console and click **Groups** in the browsing pane.
2. On the **Groups** page, click **Browse**.

   The leftmost pane displays all top-level user groups.
3. Navigate the group structure to locate the user group you want to work with. Click the desired user group to select it.
4. Using the icons in the pane directly to the right of where you selected the group, perform the desired action:
   a. To add a user, click the **Add User** icon. Using the search feature provided, locate the user you want to add to the selected group. Click **Add** to the right of the desired user.
   b. To add a group, click the **Add Group** icon. Using the search feature provided, locate the group you want to add to the selected group. Click **Add** to the right of the desired user.

   The individual user or group is added as a child to the group selected in the panel.

Alfresco Explorer administrative tasks

You perform various administrative functions in the Administration Console located in Alfresco Explorer.

Refer to the Explorer user help for full documentation on these administrative features:

- Managing categories
- Importing the ACP file into a space
- Exporting a space and its contents
- Viewing System Information
- Using the Node Browser

Exporting and importing

This section describes how to export and import information from a repository, and then import that information into either the same or another repository.

Export and import is useful for:

- Bulk extraction and loading of personal or team information from one location to another
- Integration with third party systems

- You can only perform an export/import between compatible versions of Alfresco, which generally means the same version

- You should not perform an upgrade (Versions 2.x-3.x) using export/import
- It is not recommended that you backup and restore personal, team, or complete repository scope

The export procedure produces one or more Alfresco Content Package (ACP) files, which hold the exported information. As with all files, you can place them somewhere secure, or transfer them using methods, such as email, FTP, and so on. The scope of information to export is configurable, but typically involves specifying the location within the repository to export. The hierarchical nature of the repository means that every item within that location is exported. For example, exporting the folder **My Documents** will export the folder, its contents, and all sub-folders. Security settings allow the export of only those items that are readable by the user performing the export.

Import of an ACP file is the reverse of an export. The information held in the ACP file is placed into the repository location chosen at import time. By default, the import process creates a copy of the ACP held information.

ACP files

An Alfresco Content Package (ACP) is a single file (with an extension of `.acp`) that bundles together the metadata and content files for the information to be transported.

An ACP file is simply a ZIP archive whose structure is as follows:

```
/<packagename>.xml
/<packagename>/
   contentNNN.pdf
   contentNNN.txt
   ...
```

The `packagename` is assigned on export.

`<packagename>.xml` contains an XML rendition of the transported information in the form of repository nodes. There is no restriction on the information types that can be transported but typically a bundle contains folders and files. Other information types may include forums, rules, preferences, tasks, or anything that is described by a repository content model.

The XML conforms to the export and import view schema, which describes the transported nodes in terms of their types, aspects, properties, associations, and permissions. Content properties are handled specifically where the binary content of the property is held in a separate file under the `packagename` directory of the ZIP archive, and the XML contains a reference to the file.

Although the repository provides different ways to create an ACP file (that is, to export), it is also possible to manually create one using any means. This is very useful for system to system integration.

Exporting spaces

You can export any spaces and content to which you have access. However, you can only import and export from the same version. If this is not the case, refer to the Migrating on page 177 section.

1. In Alfresco Explorer, browse to the space you want to export.
2. To export the space, either:
   - Click the **Administration Console** icon, and then click **Export**, or
   - Click **View Details**, and in the **Actions** list, click **Export**
3. In Package Name, specify a name for the file.

4. Choose **Click here to select the destination**.
5. Navigate the spaces to select a destination space for your file.
6. Select the **Current space** option.
7. Check at least one of the boxes for **Include Children** and **Include this Space**.

   Children means everything that you see in the Browse Spaces and Content Items panes.
8. Optionally, check **Run export in background**.
9. Click **OK**.
10. Click **Close**.

    Browse to the space that you specified as the destination to see the `.acp` export file.

    After exporting an ACP file, you should move it to a secure location.

Importing spaces

You can import to any spaces and content to which you have access.

1. In Alfresco Explorer, browse to the space into which you want to import.
2. To import a space, either:
   - Click the **Administration Console** icon, and then click **Import**, or
   - Click **View Details**, and in the **Actions** list, click **Import**
3. In the Import pane, browse to the location of the `ACP` file you want to import.
4. Click **Upload**.
5. Optionally, check **Run import in background**.
6. Click **OK**.
7. Click **Close**.

The information from the ACP file now resides in the destination space.

Using rules to import to a space

You can use a rule in a space to import a file to the same space or to another space in which you have permissions.

1. Browse to the space into which you want to import.
2. Click **More Actions**.
3. In the menu, click **Manage Content Rules**.
4. Click **Create Rule** to open the Create Rule Wizard.
5. In the Select Condition menu, click **Items which contain a specific value in its name**.
6. Click **Set Values and Add**.
7. In File name pattern, type `*.acp`.
8. Click **OK**, and then click **Next**.
9. In Select Action, click **Import an Alfresco content package**.
10. Click **Set Values and Add**.
11. In Import To, click the space to which you want to import.
12. Click **OK** twice, and then click **Next** to open the Enter Details pane.
13. In the Type menu, click **Inbound**.

14. In Title, type a meaningful name for the rule to identify the rule from any other rules that might apply in the space.
15. In Description, type a meaningful description that allows other users to understand what the rule does.
16. Deselect **Apply rule to sub spaces**.
17. Check **Run rule in background**.
18. Click **Next**, and then click **Finish**.

Test the rule by inserting an `ACP` file in the space that corresponds to space containing the import rule. The import process starts and places the items held in the ACP file into the destination folder specified with "Import To".

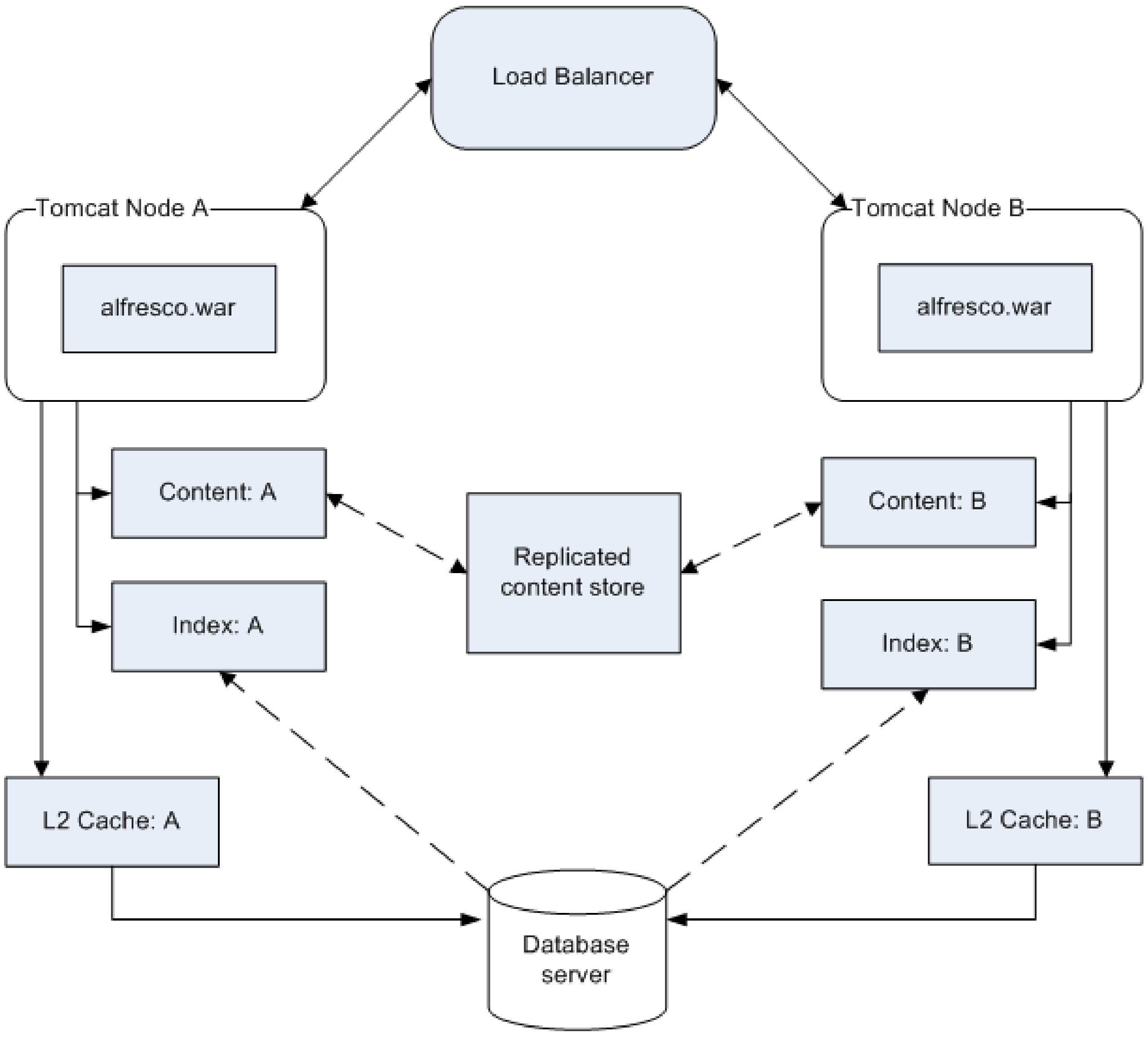
The import will be initiated regardless of how the ACP file was placed into the folder. For example, the import will initiate if the ACP file was placed there using CIFS, FTP, WebDAV, Explorer, or API. This is particularly powerful for system to system data integration.

High availability systems

This section describes how to implement multiple Alfresco instances in a high availability configuration.

High Availability (HA) clusters are implemented in Alfresco to improve the availability of services and to improve performance of these services. Availability is enhanced through redundant nodes that provide services when other nodes fail. When integrated with a load balancer, performance is enhanced by distributing, or balancing, server workload across a collection of nodes.

A cluster represents a collection of nodes. For example, a setup of two Tomcat nodes on a single computer is known as a *co-located*, or *vertical* cluster. In this configuration, each node consists of a complete and separate working implementation of Alfresco. This configuration will provide high performance when combined with a load balancer, but it does not ensure complete availability as there is no redundancy provided for the server computer itself.



High availability components

To ensure redundancy during runtime, the following components of the Alfresco system must be replicated and/or synchronized across each cluster:

- Content store
- Indexes
- Database
- Level 2 cache

Content store replication

The underlying content binaries are distributed by either sharing a common content store between all machines in a cluster or by replicating content between the clustered machines and a shared store(s).

Each cluster stores content in, and retrieves content from its primary content store. Both content stores share a replicated content store. Content placed in a primary content store is copied to the replicated content store in a process known as outbound replication. The effect is that two copies of the content exist, one in the primary content store and another in the replicated content store.

If a request for that content occurs on another node, the application first looks for the content item in the primary content store of that node. If it is not there, the application looks to the replicated content store.

If inbound replication is configured, the replicated content store copy is then copied to the requesting node's primary content store.

Index synchronization

Indexes provide searchable references to all nodes in Alfresco. However, the index store is transaction-aware and cannot be shared between servers. The indexes can be recreated from references to the database tables. To keep the indexes up to date, a timed thread updates each index directly from the database.

When a node is created (this may be a user node, content node, or space node), metadata is indexed and the transaction information is persisted in the database. When the synchronization thread runs, the other index is updated using the transaction information stored in the database.

The tables that contain the index tracking information are:

Table	Description
`alf_server`	Each committing server has an entry in this table.
`alf_transaction`	Each write operation to nodes in Alfresco generates a unique transaction ID. The transaction is attached to the entry in the server table. The column `commit_time_ms` ensures that the transactions can be ordered by commit time. A unique GUID (due to historical reasons) is also attached here.
`alf_node`	An entry is created here for each node, including nodes that have been deleted. With each modification, the status is updated to refer to the transaction entry for the committing transaction.

When indexes are replayed on a machine (for rebuilding or tracking) the transactions are time-ordered. The server can then determine which nodes have been modified or deleted in a particular transaction.

Database synchronization

It is possible to have co-located databases which synchronize with each other. To use co-located databases, refer to your database vendor documentation.

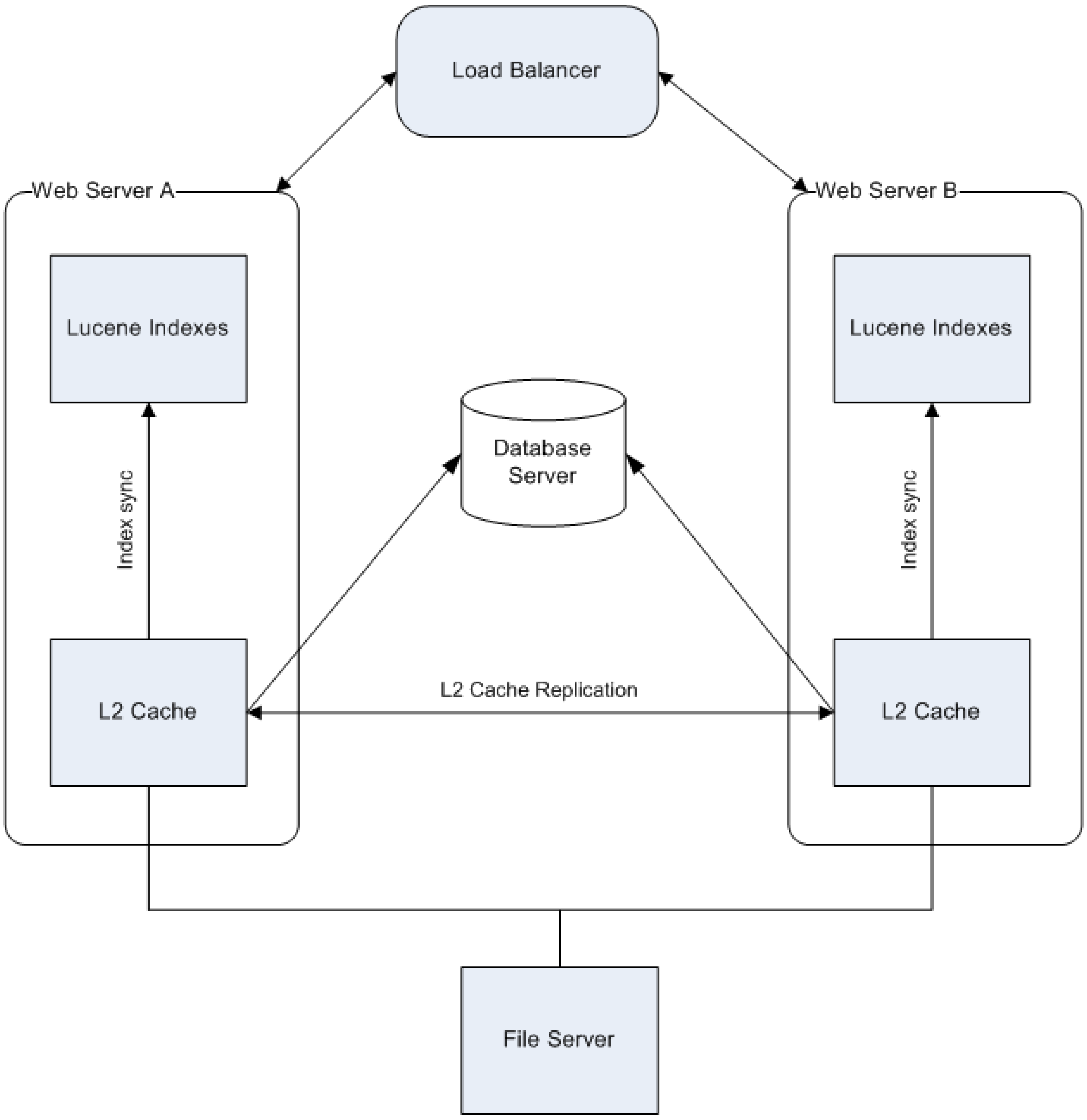
Level 2 cache replication

The Level 2 (L2) cache provides out-of-transaction caching of Java objects inside the Alfresco system. Alfresco provides support for EHCache. Using EHCache does not restrict the Alfresco system to any particular application server, so it is completely portable.

The L2 cache objects are stored in memory attached to the application scope of the server. Sticky sessions are used to keep a user that has already established a session on one server for the entire session. By default the cache replication make use of RMI to replicate changes to all nodes in the cluster using the Peer Cache Replicator. Each replicated cache member notifies all other cache instances when its content has changed.

High availability scenario

This scenario shows a single repository database and file system (for the content store) and multiple web application servers accessing the content simultaneously. The configuration does not guard against repository file system or database failure, but allows multiple web servers to share the web load, and provides redundancy in case of a web server failure. Each web server has local indexes (on the local file system).



A hardware load balancer balances the web requests among multiple web servers. The load balancer must support 'sticky' sessions so that each client always connects to the same server during the session. The file system and database will reside on separate servers, which allows us to use alternative means for file system and database replication. The configuration in this case will consist of L2 Cache replication, and index synchronization.

Configuring the simple repository clustering example

This task describes the set up procedure for the simple scenario example.

1. Use EHCache replication to update every cache in the cluster with changes from each server.
2. Open the sample `ehcache-custom.xml.sample.cluster` file.
3. Save the file as `ehcache-custom.xml`.
4. Change the following settings:

```
<ehcache>
   <diskStore
```

```
     path="java.io.tmpdir"/>
   <cacheManagerPeerProviderFactory

 class="net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory"
           properties="peerDiscovery=automatic,
                       multicastGroupAddress=230.0.0.1,
                       multicastGroupPort=4446"/>
   <cacheManagerPeerListenerFactory

 class="net.sf.ehcache.distribution.RMICacheManagerPeerListenerFactory"
           properties="port=40001, socketTimeoutMillis=2000"/>
...
</ehcache>
```

Level 2 cache is a technology to speed up Hibernate. When the application makes a Hibernate query, it does not have to do a (costly) SQL request if the object is already present in the Level 2 cache. For debugging purposes, you can disable the L2 cache. Hibernate will keep working, but at a slower rate. If you have an issue with EHCache, that is, the cache cluster test fails, you may want to confirm this by disabling the L2 cache setting in the `custom-hibernate-dialect.properties` file. For example:

```
hibernate.cache.use_second_level_cache=false
```

5. If you are using JGroups, ensure that you have set the `alfresco.cluster.name=someclustername` property.
6. Change the reindex behaviour properties in global properties `alfresco-global.properties` file.

   For example:

```
# ######################################### #
# Index Recovery and Tracking Configuration #
# ######################################### #
#
# Recovery types are:
#    NONE:     Ignore
#    VALIDATE: Checks that the first and last transaction for each store
 is represented in the indexes
#    AUTO:     Validates and auto-recovers if validation fails
#    FULL:     Full index rebuild, processing all transactions in order.
 The server is temporarily suspended.
index.recovery.mode=VALIDATE
# FULL recovery continues when encountering errors
index.recovery.stopOnError=false
index.recovery.maximumPoolSize=5
# Set the frequency with which the index tracking is triggered.
# For more information on index tracking in a cluster:
#    http://wiki.alfresco.com/wiki/
High_Availability_Configuration_V1.4_to_V2.1#Version_1.4.5.2C_2.1.1_and_later
# By default, this is effectively never, but can be modified as required.
#    Examples:
#       Once every five seconds: 0/5 * * * * ?
#       Once every two seconds : 0/2 * * * * ?
#       See http://quartz.sourceforge.net/javadoc/org/quartz/
CronTrigger.html
index.tracking.cronExpression=* * * * * ? 2099
index.tracking.adm.cronExpression=${index.tracking.cronExpression}
index.tracking.avm.cronExpression=${index.tracking.cronExpression}
# Other properties.
index.tracking.maxTxnDurationMinutes=10
index.tracking.reindexLagMs=1000
index.tracking.maxRecordSetSize=1000
index.tracking.maxTransactionsPerLuceneCommit=100
index.tracking.disableInTransactionIndexing=false
```

7. The triggers for DM (document management) and AVM (web content management, where applicable) index tracking are combined into one property for simplicity. These can be set separately, if required. The following properties should typically be modified in the clustered environment:

   ```
   index.tracking.cronExpression=0/5 * * * * ?
   index.recovery.mode=AUTO
   ```

8. Set the recovery mode to AUTO to ensure that the indexes are fully recovered, if missing or corrupt, and to top the indexes up during bootstrap in the case where the indexes are out of date. This happens frequently when a server node is introduced to the cluster. AUTO will ensure that backup, stale or no indexes can be used for the server.

   Index tracking relies heavily on the approximate commit time of transactions. This means that machines in a cluster need to be time-synchronized, the more accurately the better. The default configuration only triggers tracking every five seconds and enforces a minimum age of transaction of one second. This is controlled by the property `index.tracking.reindexLagMs`. For example, if the clocks of the machines in the cluster can only be guaranteed to within five seconds, then the tracking properties might look like this:

   ```
   index.tracking.cronExpression=0/5 * * * * ?
   index.recovery.mode=AUTO
   index.tracking.reindexLagMs=10000
   ```

9. Index tracking keeps a history of transactions that might yet be committed based on the IDs of new transactions that appear. To ensure that long-running transactions are properly detected in a cluster, set the maximum transaction duration using the following property:

   ```
   index.tracking.maxTxnDurationMinutes=10
   ```

10. The number of concurrent threads that will be assigned to index rebuilding can be set. Each of these threads picks up and indexes a batch of transactions and indexes them as a single Lucene batch. This reduces I/O and contention for Lucene merges and makes index rebuilding faster, especially during a full index rebuild. The defaults cater for a low throughput of small write transactions, but will perform full index rebuilding in good-sized batches. If your use cases includes many large transactions, then it would be better to lower the number of transactions in the index batch.

    ```
    index.recovery.maximumPoolSize=5
    index.tracking.maxTransactionsPerLuceneCommit=100
    ```

Configuring server clusters

This task describes how to configure server clusters.

1. Open the `alfresco-global.properties` file.
2. Set the following properties to point to a shared file system for all the cluster nodes:
   a. `dir.contentstore`
   b. `dir.contentstore.deleted`
   c. `dir.auditcontentstore`
3. Save the `alfresco-global.properties` file.
4. Rename `ehcache-custom.xml.sample.cluster` to `ehcache-custom.xml` to enable cache clustering.

   This activates the multicast discovery protocol so that Alfresco nodes can automatically discover each other.
5. Ensure the network does not block UDP multicast.

   Alfresco uses the `ehCache API` for caching, which employs UDP multicast for peer discovery.

6. Configure the cluster discovery options for cluster communication.

   Alfresco uses JGroups as its discovery mechanism to broadcast heartbeat messages around the cluster. JGroups can be configured to send these messages using virtually any network configuration. There are two available: UDP multicast (the default) and TCP unicast.
7. Ensure all the nodes in the cluster have their clocks synchronized within one (1) second. If this cannot be guaranteed, then increase the value of the `index.tracking.reindexLagMs` property.

Initiating clustering

Alfresco uses JGroups to send the initial broadcast messages announcing a server's availability. After initial setup, it is possible for a server to enter a cluster by setting the `alfresco.cluster.name` property.

1. Activate the `ehcache-custom.xml.sample.cluster` by renaming the file to `ehcache-custom.xml`.
2. Set the required properties using global property overrides or system properties (Java command line -D options)
   a. `alfresco.cluster.name`: The name of the cluster. This property is mandatory. Without it, neither JGroups nor index tracking will be enabled.
   b. Any JGroups properties, especially if the TCP stack is required.
   c. `index.recovery.mode`: Set this to AUTO to ensure indexes are refreshed properly on startup.
   d. `index.tracking.cronExpression`: This does not need to be set and is 0/5 * * * * ? by default. The index tracking code will not activate unless the cluster name has been set.
3. Configure the content stores.

Configuring JGroups and Alfresco clustering

This section describes the options available for configuring JGroups and other Alfresco-specific cluster options.

When setting up cluster communications, Alfresco requires servers to discover other instances of Alfresco on a network. In older Alfresco releases, this discovery process used a UDP multicast message (provided by EHCache). Servers in the cluster picked up the message and used the information to set up inter-server communication for inter-cache communication.

To provide a more flexible cluster discovery process, JGroups is integrated into the repository. JGroups is a toolkit for multicast communication between servers. It allows inter-server communication using a highly configurable transport stack, which includes UDP and TCP protocols. Additionally, JGroups manages the underlying communication channels, and cluster entry and exit.

Configuring JGroups

This section describes how to initialize clustering with JGroups. The configuration settings for JGroups clustering are set in a file called`<configRoot>/alfresco/jgroups/alfresco-jgroups-XYZ.xml`, where `XYZ` is the name of the protocol being used. There is a separate configuration file available for each stack: `alfresco-jgroups-TCP.xml`, and `alfresco-jgroups-UDP.xml`.

To initialize JGroups clustering, you must set the following property:

```
alfresco.cluster.name=<mycluster>
```

Where `<mycluster>` is the name used by JGroups to distinguish unique inter-server communication. This allows machines to use the same protocol stacks, but to ignore broadcasts from different clusters.

You can also create a cluster from the Java command line option:

```
-Dalfresco.cluster.name=mycluster
-Dalfresco.tcp.inital_hosts-remotehostname[8000]
```

1. Select the appropriate file for the required protocol stack and edit the properties:

   The default JGroups configuration will probably be adequate. However, the protocol stacks can be redefined completely using any of the standard JGroups transport protocols.

   - Edit the `<configRoot>\classes\alfresco\jgroups\alfresco-jgroups-TCP.xml` file to set the properties that define TCP protocol stack.
   - Edit the `<configRoot>\classes\alfresco\alfresco-jgroups-UDP.xml` file to set the properties that define the UDP protocol stack.

   The general properties for both protocol stacks are:

General property	Description
`alfresco.jgroups.bind_address`	The address to which you bind local sockets.
`alfresco.jgroups.bind_interface`	The interface to which you bind local sockets.

   The properties for the TCP stack are:

TCP property	Description
`alfresco.tcp.start_port`	The port that the server will start listening on. The default is 7800.
`alfresco.tcp.initial_hosts`	A list of hosts and start ports that must be pinged. This can call potential members of the cluster, including the current server and servers that might not be available. The port listed in square brackets is the port to start pinging. The default is `localhost[7800]`. For example: `HOST_A[7800],HOST_B[7800]`
`alfresco.tcp.port_range`	The number of increments to make to each host's port number during scanning. Each host has a port number in square brackets, for example, 7800. If the host does not respond to the ping message, the port number will be increased and another attempt made.

   The properties for the UDP stack are:

UDP property	Description
`alfresco.udp.mcast_addr`	The multicast address to broadcast on. The default is 230.0.0.1.
`alfresco.udp.mcast_port`	The port to use for UDP multicast broadcasts. The default is 4446.
`alfresco.udp.ip_ttl`	The multicast "time to live" to control the packet scope. The default is 2.

The default is UDP and is specified in the system `repository.properties` file. The JGroups configuration file is built using the protocol string `alfresco.jgroups.defaultProtocol=UDP`.

2. To select the TCP communication method, add the following to the `alfresco-global.properties` file:

   `alfresco.jgroups.defaultProtocol=TCP`
3. To specify your own JGroups configuration file, add the following to the `alfresco-global.properties` file:

   ```
   alfresco.jgroups.configLocation=classpath:some-classpath.xml
   alfresco.jgroups.configLocation=file:some-file-path.xml
   ```

   Where `some-file-path` is the name of your configuration file.

If you change or extend the JGroups configuration and you prefer not to set the properties using the system properties (Java command line `-D` options) , then override the bean. Values set directly on the VM using the Java command line are always taken in preference in this case.

Using EHCache multicast discovery

This section describes how to configure cluster discovery to use the EHCache multicast. Use this if you do not wish to use JGroups.

1. Open the `<extension>/ehcache-custom.xml` file.
2. Replace the `cacheManagerPeerProviderFactory` with the following:

   ```
   <cacheManagerPeerProviderFactory

    class="net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory"
               properties="peerDiscovery=automatic,
                           multicastGroupAddress=230.0.0.1,
                           multicastGroupPort=4446"/>
   ```

   You must also set the `alfresco.cluster.name` property in order to activate index tracking.

   If you have several alfresco clusters on the same network, ensure that the addresses and ports used for Ehcache UDP broadcasts (using the `multicastGroupAddress` and `multicastGroupPort` parameters) are unique to each cluster. If you use the same parameters for all the clusters, each cluster will try to communicate with the other clusters, leading to potential issues.

   For example, you may have a testing or validation Alfresco cluster environment and a production Alfresco cluster environment on the same network. When you copy or transfer your testing configuration files to the production environment, you must change the parameters.
3. Save the file.

Clustering through shared content stores

This section describes how to configure clustering through shared content stores.

1. Ensure the following stores exist on a shared file system:
   a. Content Store
   b. Audit Content Store
   c. Deleted Content Store
2. Open the `<ClasspathRoot>/alfresco-global.properties` file.

3. Add the following properties:

```
dir.contentstore=<new_location>/contentstore
dir.contentstore.deleted=<new_location>/contentstore.deleted
dir.auditcontentstore=<new_location>/audit.contentstore
```

4. Save the file.

Verifying the cluster

This section describes how to verify that clustering is working for the various components involved. You will need direct Explorer access to each of the machines in the cluster. The operation is done on machine one (M1) and verified on the other machines (Mx). The process can be switched around with any machine being chosen as M1.

Testing cache clustering

1. On M1, login as user `admin`.
2. Browse to the Guest Home space, locate the tutorial PDF document and view the document properties.
3. On Mx, login as `admin`.
4. Browse to the Guest Home space, locate the tutorial PDF document and view the document properties.
5. On M1, modify the tutorial PDF description field, and add `abcdef`.
6. On Mx, refresh the document properties view of the tutorial PDF document.
7. The description field must have changed to include `abcdef`.

Index clustering

1. Repeat the steps for testing cache clustering.
2. On M1, perform an advanced search of the description field for `abcdef` and verify that the tutorial document is returned.
3. On Mx, search the description field for `abcdef`. Allow time for index tracking (10s or so), and the document will show up in the search results.

Testing content replication and sharing

1. On M1, add a text file to Guest Home containing `abcdef`.
2. Refresh the view of Guest Home and verify that the document is visible.

   This relies on index tracking, so it may take a few seconds for the document to be visible.
3. On Mx, perform a simple search for `abcdef` and ensure that the new document is retrieved.
4. Open the document and ensure that the correct text is visible.
5. Perform a simple search for `abcdef` and ensure that the new document is retrieved.

Testing WCM clustering

The following sections describe how to test WCM clustering.

Web project creation

1. On M1, create a web project, and add only the **Name** and **DNS Name** fields.
2. On M2, verify that the web project is created on M2.

3. On M1, edit the web project by adding values in the **Title** and **Description** fields.
4. On M2, verify the web project title and descriptions.

Web project data node synchronization

1. On M2, stop the Alfresco server.
2. On M1, import the sample website `alfresco-sample-website.war` into the `admin` user sandbox.
3. On M2, start the Alfresco server.
4. On M2, after synchronization time, browse the web project through the `admin` user sandbox and open `robots.txt` from the website root.

Web project user invite

1. On M1, create a new user.
2. On M2, invite the new user to a web project.
3. On M1, show all sandboxes and observe the new user's sandbox.

Tracking clustering issues

This section describes how to track issues with clustering in a high availability environment.

1. Enable the following log categories:
   - `log4j.logger.net.sf.ehcache.distribution=DEBUG`

     Check that heartbeats are received from from live machines.
   - `log4j.logger.org.alfresco.repo.node.index.IndexTransactionTracker=DEBUG`

     Remote index tracking for Alfresco DM.
   - `log4j.logger.org.alfresco.repo.node.index.AVMRemoteSnapshotTracker=DEBUG`

     Remote index tracking for AVM.
2. Enable the following JGroups logs:
   - `log4j.logger.org.alfresco.repo.jgroups=debug`

     `log4j.logger.org.alfresco.repo.cache.jgroups=debug`

     Checks heartbeat messages sent and received by machines in the cluster, and watches the entry and exit of cluster members.
3. If cache clustering is not working, the EHCache website describes some common problems. The remote debugger can be downloaded as part of the EHCache distribution files and executed:

```
> java -jar ehcache-1.3.0-remote-debugger.jar
Command line to list caches to monitor: java -jar ehcache-remote-
debugger.jar path_to_ehcache.xml
Command line to monitor a specific cache: java -jar ehcache-remote-
debugger.jar path_to_ehcache.xml cacheName
```

Back up and restore

This section provides information for the following:

- Backing up and restoring the Alfresco repository
- Backing up and restoring Lucene indexes
- Restoring a MySQL database

Backing up the repository

Backing up an Alfresco repository involves backing up the following:

- The directory pointed to by the `dir.root` setting
- The database Alfresco is configured to use

To restore the backup successfully, the directory and database must be backed up as a single unit. When you restore an Alfresco backup, you must restore both the `dir.root` directory and the Alfresco database from the same backup set. Otherwise, the repository will be corrupted.

The `dir.root` directory is defined in the `alfresco-global.properties` file. By default, this directory is named `alf_data` and is located within the directory where Alfresco is installed.

This section only describes the process for backing up the Alfresco content repository. It assumes that the various binaries (operating system, database, JDK, application server, Alfresco, and so on) are being backed up independently of the process described here.

Hot backup

Hot backup allows you to back up the states of the Alfresco application and the database while these are still running. There is, therefore, no interruption in service to the users.

In an Alfresco configuration, MySQL uses InnoDB tables. MySQL itself does not have hot backup facilities for InnoDB tables; it has it only for MyISAM tables. However, there are some commercial tools that will perform hot backups of InnoDB tables:

- InnoDB Hot Backup tool: http://www.innodb.com/order.php

  This tool is mentioned in the MySQL documentation as a means to hot backup MySQL InnoDB tables.

- MySQL Plugin for Arkeia Server Backup: http://www.arkeia.com/products/asb/

If you have one or more replication slave servers (see Setting up replication on page 78), you can avoid the complexity of hot backups by doing normal backups from one of the slaves. Only one slave server needs to be shut down, so there is no impact on users.

Performing a hot backup

Alfresco includes a background job responsible for backing up the Lucene indexes that (by default) is configured to run at 3am each night. The hot backup process must not run concurrently with this background job, so you should either ensure that the hot backup completes by 3am, or wait until the index backup job has completed before initiating a hot backup.

The WCM functionality includes a background job called the `avmOrphanReaperJob` that must not be allowed to run while a hot backup is in progress. By default this job is scheduled to run every minute, but this can be reconfigured by overriding the default schedule defined in `scheduled-jobs-context.xml`. Make sure you are familiar with Alfresco's repository configuration mechanism before making any configuration changes. If you are not using the Alfresco WCM functionality, no action is required.

1. Back up the database that Alfresco is configured to use using your database vendor's backup tools.

   Backup capabilities vary widely between relational database products, and you should ensure that any backup procedures that are instituted are validated by a qualified, experienced Database Administrator before being put into a production environment.

2. Back up the following file system subdirectories of the Alfresco `dir.root` directory using your preferred tool (for example, `rsync`, `xcopy`).

Never attempt to back up the `lucene-indexes` subdirectory while Alfresco is running. This will cause Lucene index corruption.

   a. `contentstore`
   b. `contentstore.deleted`
   c. `audit.contentstore`
   d. `backup_lucene_index`

3. Store both the database and `dir.root` backups together as a single unit.

Performing a cold backup

By default, the `dir.root` contains both the content and indexes. For a cold backup, you can back up just the content and perform a full reindex when a backup is restored. However, a full reindex can be a time consuming process, so the steps below include the indexes in the backup.

1. Stop Alfresco. See Stopping the Alfresco server on page 54.
2. Back up the database Alfresco is configured to use, using your database vendor's backup tools.
3. In parallel, back up `dir.root` in its entirety.
4. Store both the database and `dir.root` backups together as a single unit.
5. Start Alfresco. See Starting the Alfresco server on page 53.

Restoring the repository

This task describes how to restore the Alfresco repository.

1. Stop the Alfresco server.
2. Copy the current `dir.root` to a temporary location.
3. Restore the `dir.root` that you previously backed up.
4. Restore the database from `dir.root`.

   For instructions on restoring the database, refer to Restoring a MySQL database on page 172.
5. Start the Alfresco server.

Backing up and restoring Lucene indexes

Lucene is a text search engine library written entirely in Java. It is used within Alfresco for full-text search. This section describes how to back up and restore the Lucene indexes.

Backing up the Lucene indexes

This task describes how to perform a safe back up of the Lucene indexes.

1. Copy the following file:

   `<configRoot>/alfresco/scheduled-jobs-context.xml`
2. Locate the `<extension>` directory, and paste the copied file into this location.
3. Delete each pair of `<bean> </bean>` tags (excluding the pair containing the `indexBackupTrigger` bean).

   This bean contains the following properties:

```
<!-- trigger at 3am each day -->
        <property name="cronExpression">
            <value>0 0 3 * * ?</value>
        </property>
```

4. Modify the `cronExpression` values.

   For example, the value `0 0 3 * * ?` specifies the backup is triggered at 3am every day.
5. Save the file as `custom-scheduled-jobs-context.xml`.

   As your override file ends with `-context.xml`, you do not need to point to your file.

   After each backup scheduled in the `indexBackupTrigger` bean, perform your normal backup of this directory.

   Each backup will overwrite the existing backup.

Backing up the Lucene backup directory

This task describes how to back up the Lucene backup directory.

1. Browse to the following file:

   `<configRoot>/alfresco/core-services-context.xml`
2. In `<configRoot>/alfresco/core-services-context.xml`, locate the following bean:

   `luceneIndexBackupComponent`

   The `targetLocation` property contains the backup location, with the default value `<dir.root>/backup-lucene-indexes`.

Restoring the Lucene indexes

In addition to full restorations, you can also use the backup sets created through either the cold or hot backup procedures to restore just the Lucene indexes. This is useful where the repository itself does not need to be restored but for some reason the Lucene indexes are stale.

1. Stop the Alfresco server.
2. Move the existing `dir.root/lucene-indexes` directory to another location.
3. Perform the following, depending on whether you are doing a hot or cold backup:
   - For cold backups, restore `dir.root/lucene-indexes` from the most recent backup step.
   - For hot backups, restore `dir.root/backup_lucene_index` from the most recent backup step and rename it to `dir.root/lucene-indexes`.
4. Restart the Alfresco server.

   Upon restarting, Alfresco will detect that the indexes are stale, and incrementally reindex just the content that has changed since the last backup.

   For incremental reindexing to occur properly, leave the `index.recovery.mode` property at its default value of `AUTO`. Setting this property to `FULL` forces a full reindex, even if incremental reindexing is possible, negating any benefits from this procedure. If a full rebuild is required, it is quicker to delete the existing index.

Restoring a MySQL database

This task describes how to perform a restoration of the MySQL database.

1. Open a command prompt.
2. Browse to the location of the file that you created using `mysqldump`.
3. Run the command `mysql -u root -p`.
4. Enter the root password when prompted.

5. Enter `create database alfresco;` to create a database called `alfresco`.
6. Type `exit;` to exit from MySQL.
7. Type `mysql -u root -p alfresco < dumpFile.sql` to load your dump file into the `alfresco` database.
8. Enter the root password when prompted.

Upgrading Alfresco

This section describes the recommended procedure for performing an upgrade. The procedure involves a new installation of the Alfresco binaries and configuration, and an in-place upgrade of a copy of the repository. In-place upgrade of the Alfresco binaries and configuration is not recommended.

Before starting an upgrade:

- Ensure that you have backed up your production environment
- If you have any customizations (for example, AMPs) in your existing Alfresco installation, recompile all Java code against the new version of Alfresco and regression test against the new version of Alfresco
- When you upgrade Alfresco with Oracle, the `alfresco` user needs more privileges than connect and resource. At minimum, the `alfresco` user should have permission to delete objects. A safer option is to give a `sysdba` role for the upgrade process only. After the upgrade, this role should be removed.

You must perform a test upgrade using a backup copy of the repository before attempting to upgrade your production environment. Therefore it is important that your backups are up-to-date.

1. Validate your platform is still on the supported stacks for the new version of Alfresco. See Supported stacks on page 13.
2. Shut down your existing Alfresco instance, including any virtualization servers and File System Receivers. Alfresco runtimes may be left running and upgraded at a later time using this same process.
3. Perform a cold back up of your repository. See Backing up the repository on page 170.
4. Back up any configuration overrides from the `<extension>` directory.

   Do not copy the files. Copy only the override settings so that you will not overwrite the new extension files in the upgraded version.
5. Download and install the new version of the Alfresco WAR in a different directory to the existing installation. See Installing Alfresco on page 16.
6. Validate the new installation to check that it is working correctly. See Validating an upgrade on page 176.
   a. Configure the new installation with a new repository (not the existing one).
   b. Start Alfresco and validate that the system works correctly.
   c. Shut down Alfresco and (optionally) clear the new repository.
7. Restore the backup into the new repository (not the existing one). See Restoring the repository on page 171.
8. Manually apply your backed up configuration override settings into the relevant new override files in the `<extension>` directory. Ensure that you do not overwrite the new extension files. See Configuring an upgrade on page 175.
9. Reinstall all customizations into the new Alfresco instance.

10. Restart the Alfresco server for the configuration changes to take place. Monitor the startup log messages for information on the status of the upgrade. See Starting the Alfresco server on page 53.
11. Remove the old Alfresco installation and repository.

If you are satisfied that the upgrade is successful, remove the old Alfresco installation and repository. This may be done at a later time, once the new installation has been thoroughly validated.

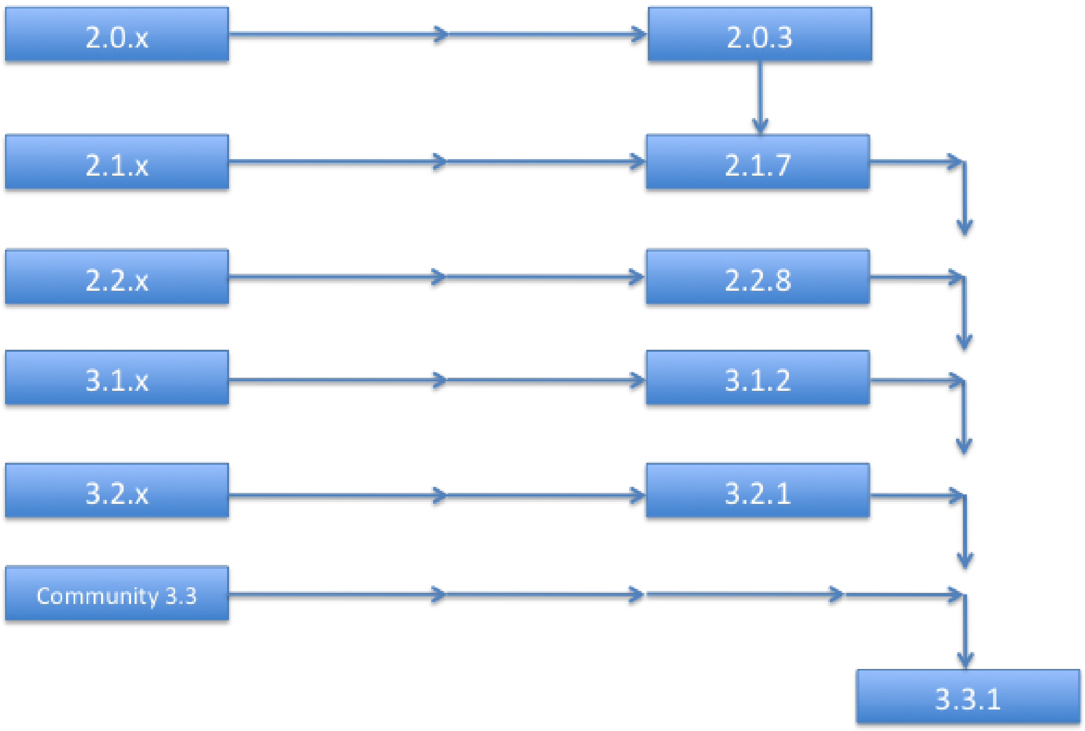
Alfresco upgrade paths

When you upgrade Alfresco, it is recommended that you follow a structured upgrade path between versions.

Alfresco supports upgrading up to two major versions above your existing version. In the table, 3.x, 2.2, 2.1, and 2.0 are all considered to be major version. This means that you can upgrade directly from 2.1 to 3.x; whereas 2.0 would require an intermediate step.

Major version upgrades

To upgrade to the latest version, first apply the latest service pack for your current version, then upgrade to the latest version (including service pack). The exception to this is when you are upgrading from before 2.1, where you must first go to 2.0.3, and then to 2.1.7. The following diagram shows the upgrade paths for major versions:



For example, if your production environment currently runs Alfresco Version 2.2 SP1, you need to upgrade to Version 2.2 SP8 before you can upgrade to Version 3.3 SP1.

Minor version upgrades

From:	To:
2.1 SP7; 2.2 SP8; 3.1 SP2; 3.2 SP1; 3.3	3.3 SP1
2.1 SP7; 2.2 SP7; 3.1 SP2; 3.2 SP1; Community 3.3	3.3
2.1 SP7; 2.2 SP7; 3.1 SP2; 3.2R	3.2 SP1
2.1 SP7; 2.2 SP6; 3.1 SP2; 3.2; Community 3.2	3.2r
2.1 SP7; 2.2 SP5; 3.1 SP2; Community 3.2	3.2
2.1 SP7; 2.2 SP5; 3.1 SP1	3.1 SP2
2.1 SP7; 2.2 SP4; 3.1	3.1 SP1
2.1 SP7; 2.2 SP3; 3.0 SP1; Community	3.1
2.1 SP6; 2.2; 2.2 SP2; 3.0	3.0 SP1
2.1 SP5; 2.2; 2.2 SP1; Community	3.0
Previous service pack, and latest service pack available from previous release.	2.2 - 2.2 SP8
2.1 SP5; 2.1 SP6	2.1 SP7
Previous service pack. For Legacy upgrades, Alfresco recommend engaging consulting.	1.2 - 2.1 SP4

Configuring an upgrade

Before running the server for the first time, check the database connection details and Alfresco data folder locations, and set them according to the environment in which the server is running.

By default, the server creates a data folder for storing content binaries and indexes at a location relative to the caller's location when the server starts. This is appropriate for quick previews and tests, but should be changed when running a server that will be storing long-lived data.

1. Locate the distribution's configuration files and samples.
2. Reapply any configuration changes to the new installation in the `<extension>` directory.
3. Open the `alfresco-global.properties` file.
4. Choose a root location for the storage of content binaries and index files.
5. Adjust the properties to change the database connection details.
6. Note the choice of JDBC driver used with each connection type.
7. Choose a Hibernate schema most appropriate to your database.
8. Save the file.
9. If you have any customizations (AMPs, patches, and so on) in your existing Alfresco installation, do the following:
   a. Recompile all Java code against the new version of Alfresco and regression test against the new version of Alfresco (this is best done prior to the upgrade itself, since it could be a lengthy exercise).
   b. Reinstall all customizations into the new Alfresco instance.
10. Start the server.

The configuration overrides will ensure that the server immediately directs data to the appropriate locations.

Validating an upgrade

Once you have upgraded, follow these steps to validate the new installation.

1. Restart the Alfresco server.

   The configuration overrides ensure the server immediately directs data to the appropriate locations.
2. Monitor the startup log messages for information on the status of the upgrade.
3. Validate the new installation using a blank repository.
4. Configure the new installation with a new repository (not the existing one).
5. Verify the database connection details and Alfresco data folder locations are set according to the environment in which the server is running.
6. Start Alfresco and validate the system works correctly.
7. Shut down Alfresco.
8. When you are certain the new installation is thoroughly validated, remove the old Alfresco installation and repository.

WCM-specific upgrade

This section describes the specific instructions required for upgrading WCM.

To avoid problems with deployment from one version of Alfresco to another, you should plan your upgrade such that an Alfresco instance, and all of its dependent runtimes, are upgraded at the same time.

If you avoid deployment, these components do not all need to be taken offline, upgraded, and brought back online at exactly the same time. You can use a rolling upgrade process to avoid downtime on your site.

When upgrading, virtualization servers are considered to be an integral part of Alfresco and must be upgraded in lock-step with the main Alfresco instance(s).

Upgrading Version 2.2 SP3 to Version 3.1

When upgrading Alfresco 2.2 SP3 to Version 3.1 with WCM, you must modify a property value in `wcm-bootstrap-context.xml` to enable the patch to run on every installation.

1. Open the file `wcm-bootstrap-context.xml`.
2. Locate the following property:

   ```
   <property name="fixesToSchema"><value>9999</value></property>
   ```
3. Change the value `9999` to `${version.schema}`.

   For example:

   ```
   <property name="fixesToSchema"><value>${version.schema}</value></
   property>
   ```

Upgrading an Alfresco runtime

Any Alfresco runtimes in your environment must be upgraded using the same general upgrade process. This process requires downtime, so to accomplish an interruption-free upgrade, configure at least two Alfresco runtimes and upgrade each of them separately (so that at least one remains online at all times).

Upgrading a cluster

If you have configured an Alfresco cluster, you must perform this task when upgrading Alfresco.

1. Shut down all nodes in the cluster.
2. Perform the steps described in the upgrading general process on each node in turn, ensuring that each node starts fully before restarting the next one. See Upgrading Alfresco on page 173.

   You only have to copy the database once as it will be upgraded by the first node to be upgraded. The other nodes will detect it has been upgraded and skip the database upgrade step.

Migrating

This section describes how to perform various migration procedures for Alfresco servers and databases.

Migrating servers

The process of migrating an instance of Alfresco running on one server to another server follows a similar pattern to the backup process, with additional steps to ensure any configuration is also copied over.

The `dir.root` property is usually defined in the `alfresco-global.properties` file.

The `dir.root` is often a directory named `alf_data` within the directory where Alfresco is installed, and will hold both content and full text indexes by default. The `dir.root` location is also reported in the Alfresco logs when the server is started.

Backing up Alfresco Server 1

This task describes how to back up the first server for migration.

1. Stop the application server to ensure that no changes can be made while backing up or restoring.
2. Export the database to `dir.root` (same location as content and indexes).
3. Copy the `configuration` directory to `dir.root`.

   For example:

   ```
   cp -r tomcat/shared/classes/alfresco/extension alf_data
   ```
4. Back up `dir.root`.

Restoring to Server 2

This task describes how to restore a back up of a server to another server.

1. Install a compatible Alfresco server. This is typically an identical version to server 1.

   Do not start the new Alfresco server.
2. Restore `dir.root`. If the path is different on server 2, change the `dir.root` configuration.
3. Rename the new server's configuration directory.

   For example:

   ```
   mv tomcat/shared/classes/alfresco/extension new_ext
   ```
4. Move the configuration directory from `dir.root` to the appropriate location

   For example:

   ```
   mv alf_data/extension tomcat/shared/classes/alfresco
   ```
5. If any configuration references server 1 explicitly, change these references to server 2.

6. Import the database from `dir.root`.
7. Start the Alfresco server.

You should now have a new instance of Alfresco on a second server with identical data. If you wish to migrate the data from one type of database to another, see Migrating from the default database to another database.

Alfresco multi-tenancy

Alfresco supports a single-instance, single-tenant (ST) environment where each tenant (for example, customer, company, or organization) runs a single instance that is installed on one server or across a cluster of servers. You can also run multiple instances of Alfresco on the same server by configuring separate database schemas, separate content stores, separate (non-conflicting) ports, and so on. However, this adds additional complexity in terms of configuration and upgrades.

The multi-tenancy (MT) features enable Alfresco to be configured as a true single-instance, multi-tenant environment. This enables multiple independent tenants to be hosted on a single instance, which can be installed either on a single server or across a cluster of servers. The Alfresco instance is logically partitioned such that it will appear to each tenant that they are accessing a completely separate instance of Alfresco.

Enabling multi-tenancy

You can configure a multi-tenant (MT) environment by renaming three sample MT extension files, and then restarting Alfresco.

1. Locate the following directory:

   `<extension>\mt\`
2. Remove the `.sample` extension from the following MT files:
   a. `mt-admin-context.xml.sample`
   b. `mt-contentstore-context.xml.sample`
   c. `mt-context.xml.sample`

   All three files must be renamed to enable MT.
3. Restart the Alfresco server.

Managing tenants

The default Alfresco administrator user has access to the default environment and can be considered to be a "super tenant". The administrator can manage tenants using the Tenant Administration Console.

1. Log in to Alfresco as the `admin` user and access: `http://localhost:8080/alfresco/faces/jsp/admin/tenantadmin-console.jsp`
2. Perform the following as required:
   a. To list all tenants and show their details, type `show tenants`.
   b. To show details for a single tenant, type `show tenant <tenant domain>`.

      This shows the status (for example, whether it is enabled or disabled) and the root content store directory.
   c. To create a tenant, type `create <tenant domain> <tenant admin password> [<root contentstore dir>]`.

For example, `create zzz.com l3tm31n /usr/tenantstores/zzz`

This creates an empty tenant. By default the tenant will be enabled. It will have an administrator user called `admin@<tenant domain>` with the supplied password. All users that the administrator creates can login using `<username>@<tenant domain>`. The root of the content store directory can be optionally specified, otherwise it defaults to the repository default root content store (as specified by the `dir.contentstore` property). The default workflows are also be bootstrapped.

d. To enable a tenant, type `enable <tenant domain>`.

   This enables the tenant so that it is active and available for new logins.

e. To disable a tenant, type `disable <tenant domain>`.

   This disables the tenant so that it is inactive and prevents tenant login.

Multi-tenancy administration

Once a tenant is created and enabled, the tenant administrator can log into Explorer and access the Administration Console within the context of their tenant domain.

For example, if a tenant/organization called `acme` is created, the tenant administrator can log in as `admin@acme` and create users such as `alice@acme`, and `bob@acme`.

The administration features currently available to the tenant administrator include:

- Manage system users (including user Usages and Quotas)
- Manage user groups
- Category management
- Export and import
- System information
- Node browser

Multi-tenancy export and import

This section describes how a tenant administrator can export and import spaces and tenants using the Tenant Administration Console.

Repository export does not apply to certain areas, such as in-flight workflows. A repository import must be into the same version of Alfresco from which the export was performed.

1. Log in to Alfresco as the `admin` user and access: `http://localhost:8080/alfresco/faces/jsp/admin/tenantadmin-console.jsp`
2. Use the export feature to export a tenant:

   ```
   export <tenant domain> <destination directory>
   ```

   This exports the tenant to a set of repository export files in a given destination directory. Export file names will be suffixed with `<tenant domain>_`.
3. Use the import feature to import a tenant:

   ```
   import <tenant domain> <source directory> [<root contentstore dir>]
   ```

   This creates a tenant by importing the tenant files from the given source directory. The import file names must be suffixed with `<tenant domain>_`.

   If an existing tenant needs to be re-imported, the tenant must be deleted first. To cleanly delete a tenant, the server must be restarted to clear the index threads. The tenant-specific index directories and tenant-specific content directories must also be manually deleted before starting the import.

Multi-tenancy implementation

To implement multi-tenancy, Alfresco has been logically partitioned such that each tenant has access to their own set of tenant-specific stores. These stores are typically routed to their own physical root directory. This also means that indexes are partitioned, since Alfresco maintains an index per store.

All Alfresco-related services are partitioned including node services, security services, workflow services, search and index services, and dictionary services. To support Alfresco Share in a multi-tenant environment, additional partitioned services include site services, activity services, invite services, and AVM services.

The metadata is logically partitioned within the database schema.

Logging enables nested diagnostic context (NDC). For a single tenant environment, the log output will show the user name context. For a multi-tenant environment, the log output also shows the tenant context.

Clustering

The MT features have been designed and implemented to work in a clustered configuration.

Cache size

If you wish to support a large number of tenants (for example, greater than 99), then must review and increase the cache size for all of the tenant caches. The cache sizes are by default configured to 100 (including the default domain) in the `<configRoot>/cache-context.xml` file. Change the cache size in the `ehcache-custom-cluster.xml.sample.cluster` file.

Tenant-based caches currently include:

- `webScriptsRegistryCache (RepositoryContainer)`
- `prefixesCache (NamespaceDAOImpl)`
- `urisCache (NamespaceDAOImpl)`
- `compiledModelsCache (DictionaryDAOImpl)`
- `uriToModelsCache (DictionaryDAOImpl)`
- `messagesCache (MessageServiceImpl)`
- `loadedResourceBundlesCache (MessageServiceImpl)`
- `resourceBundleBaseNamesCache (MessageServiceImpl)`

Modules

Alfresco supports the ability to pre-package AMPs (Alfresco Module Packages) into the Alfresco WAR, which are installed into the default domain on start up. In a multi-tenant environment, the module is also installed into each tenant domain when the tenant is created or imported.

Features not currently supported in a multi-tenant environment

The following features and components are not supported in a multi-tenant production environment:

- CIFS
- WCM
- Portlets
- Delete tenant (pending Delete Store)
- LDAP, NTLM and authentication methods other than `alfresco`
- Inbound email
- Content replication

- IMAP
- SharePoint Protocol
- Alfresco Share (for versions prior to Alfresco 3.2)

Managing the content store

The Content Store Selector provides a mechanism to control the store used for the content file associated with a particular content item.

By applying the `cm:storeSelector` aspect and setting its `cm:storeName` property to the name of a selectable store, the content will be automatically moved from its current location to the new store. The store does not, therefore, store content itself, it defines and manages those stores that are available for selection.

This allows storage polices to be implemented to control which underlying physical storage is used, based on your applications needs or business policies. For example, if you have a fast (and expensive) local disk, you can use this to store the files that you want to ensure are served for best performance; however, infrequently used files may be stored on lower cost, slower storage.

Content store selector configuration example

The following example defines two file stores, in addition to the standard default file store. By setting the `cm:storeName` property to either of these new stores or the default store, the content is automatically moved from its existing store to the relevant new store.

A summary of the procedure is as follows:

1. Create a file called `sample-content-store-selector-context.xml` in the `extension` directory.
2. Define two new file stores.
3. Declare a `storeSelectorContentStore` to be the system's primary content store.
4. Declare the mapping between the store name and store instances.
5. Add the extra stores to the list to be handled by the `eagerContentStoreCleaner`.

1. Open the `sample-content-store-selector-context.xml` file.
2. Define the new file stores by adding the following bean definitions:

```
<bean id="firstSharedFileContentStore"
 class="org.alfresco.repo.content.filestore.FileContentStore">
   <constructor-arg>
      <value>${dir.root}/storeA</value>
   </constructor-arg>
</bean>

<bean id="secondSharedFileContentStore"
 class="org.alfresco.repo.content.filestore.FileContentStore">
   <constructor-arg>
      <value>${dir.root}/storeB</value>
   </constructor-arg>
</bean>
```

   This configuration snippet defines two new stores. The physical location is relative to the `dir.root` property defined in the `alfresco-global.properties` file.
3. Declare the `storeSelectorContentStore` to be the primary content store by adding the following bean definition:

```
<bean id="contentService" parent="baseContentService">
   <property name="store">
       <ref bean="storeSelectorContentStore" />
   </property>
</bean>
```

4. Declare the mapping between store names and store instances.

```
<bean id="storeSelectorContentStore"
 parent="storeSelectorContentStoreBase">
       <property name="defaultStoreName">
            <value>default</value>
       </property>
       <property name="storesByName">
           <map>
               <entry key="default">
                   <ref bean="fileContentStore" />
               </entry>
               <entry key="storeA">
                   <ref bean="firstSharedFileContentStore" />
               </entry>
               <entry key="storeB">
                   <ref bean="secondSharedFileContentStore" />
               </entry>
          </map>
       </property>
   </bean>
```

   The list of stores is defined by the `<property name="storesByName">` property. Any stores you want to be available to the `storeSelectorContentStore` should be listed under this property.

Using the new content store

The new content store is set using the `cm:storeName` property.

The `cm:storeName` property can be set in number of ways:

- Manually, by exposing this property so its value can be set by either Explorer or Share
- Running a script action that sets the `cm:storeName` property value within the script
- Using a rule that runs a script action to set the property

The expected behavior is as follows:

- When the `cm:storeSelector` aspect is not present or is removed, the content is copied to a new location in the 'default' store
- When the `cm:storeSelector` aspect is added or changed, the content is copied to the named store
- Under normal circumstances, a trail of content will be left in the stores, just as it would be if the content were being modified. The normal processes to clean up the orphaned content will be followed.

Content Store Selector full configuration example

The following example shows the full definition of creating new stores using the Content Store Selector.

This configuration must be saved an a extension, for example, `<extension>\sample-content-store-selector-context.xml`.

The list of stores available can be set by updating the list under the `<property name="storesByName">` property.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN' 'http://
www.springframework.org/dtd/spring-beans.dtd'>
```

```
<beans>

   <bean id="firstSharedFileContentStore"
 class="org.alfresco.repo.content.filestore.FileContentStore">
      <constructor-arg>
         <value>${dir.root}/storeA</value>
      </constructor-arg>
   </bean>

   <bean id="secondSharedFileContentStore"
 class="org.alfresco.repo.content.filestore.FileContentStore">
      <constructor-arg>
         <value>${dir.root}/storeB</value>
      </constructor-arg>
   </bean>

   <bean id="storeSelectorContentStore" parent="storeSelectorContentStoreBase">
       <property name="defaultStoreName">
            <value>default</value>
       </property>
       <property name="storesByName">
           <map>
               <entry key="default">
                   <ref bean="fileContentStore" />
               </entry>
               <entry key="storeA">
                   <ref bean="firstSharedFileContentStore" />
               </entry>
               <entry key="storeB">
                   <ref bean="secondSharedFileContentStore" />
               </entry>
          </map>
       </property>
   </bean>

<!-- Point the ContentService to the 'selector' store -->
   <bean id="contentService" parent="baseContentService">
      <property name="store">
         <ref bean="storeSelectorContentStore" />
      </property>
   </bean>

   <!-- Add the other stores to the list of stores for cleaning -->
   <bean id="eagerContentStoreCleaner"
 class="org.alfresco.repo.content.cleanup.EagerContentStoreCleaner" init-
method="init">
      <property name="eagerOrphanCleanup" >
         <value>${system.content.eagerOrphanCleanup}</value>
      </property>
      <property name="stores" >
         <list>
            <ref bean="fileContentStore" />
            <ref bean="firstSharedFileContentStore" />
            <ref bean="secondSharedFileContentStore" />
         </list>
      </property>
      <property name="listeners" >
         <ref bean="deletedContentBackupListeners" />
      </property>
   </bean>

</beans>
```

The following example shows the web-client-config-custom.xml file:

```
 <!-- Configuring in the cm:storeSelector aspect -->
   <config evaluator="aspect-name" condition="cm:storeSelector">
```

```
      <property-sheet>
         <show-property name="cm:storeName" />
      </property-sheet>
   </config>
   <config evaluator="string-compare" condition="Action Wizards">
      <aspects>
         <aspect name="cm:storeSelector"/>
      </aspects>
   </config>
   ...
```

Monitoring Alfresco

This section describes how to monitor Alfresco using the Java Management Extension (JMX) and the Hyperic plug in.

JMX monitoring and management extensions

This section describes the JMX-based monitoring and management functionality.

The monitoring and management extensions can be subdivided into three categories:

Read-only monitoring beans
Expose a variety of real-time metrics for monitoring health and throughput of your Alfresco server.

Configuration beans
Provide an easily navigable view of key system configuration for support and diagnostic purposes.

Management beans
Allow control over various subsystems.

For more information on these categories of bean, refer to the reference section JMX bean categories reference on page 213.

Coexistence with other MBeans

If there is an MBean server already running on the Java Virtual Machine (JVM) that Alfresco is running on, Alfresco will export its MBeans to that server. Otherwise, Alfresco will start up its own MBean server. This means that, for example, on Tomcat or WebLogic, the Alfresco beans will compliment those provided by the application server and will be navigable in the same context with a suitable JMX client.

Activating the Sun JMX agent and local JMX connectivity

Using Tomcat and a Sun JVM for the richest possible monitoring experience, you can get Alfresco and Tomcat to share the JVM's own platform MBean server, whose pre-registered MXBeans give a detailed view of the JVM's health, usage and throughput, in areas including class loading, Hotspot compilation, garbage collection and thread activity. Sun's MBean server also provides a convenient 'local' connection method, allowing the Alfresco process to be automatically 'discovered' by a JMX client such as JConsole or Hyperic agent without manual configuration of connection details. For more information on using Hyperic with Alfresco, refer to Installing Alfresco Enterprise plug in for Hyperic on page 185

The Sun JMX agent can also be activated in 'remote' mode (where a connection is made through an RMI lookup). However, since Alfresco is always preconfigured to allow a secure remote JMX connection on any JVM, it is most likely that you will choose to activate the Sun JMX agent in local mode. This will mean the platform MBean Server will be shared by Alfresco and still be available for remote connections through the RMI connector.

- To activate the Sun JMX agent in local mode, you simply need to ensure that the following system property is set:

  `com.sun.management.jmxremote`

  For example, in your Tomcat startup script, you could use the following line:

  ```
  export JAVA_OPTS="${JAVA_OPTS} -Dcom.sun.management.jmxremote"
  ```
- Refer to the Sun documentation for more information on all the possible configuration options.

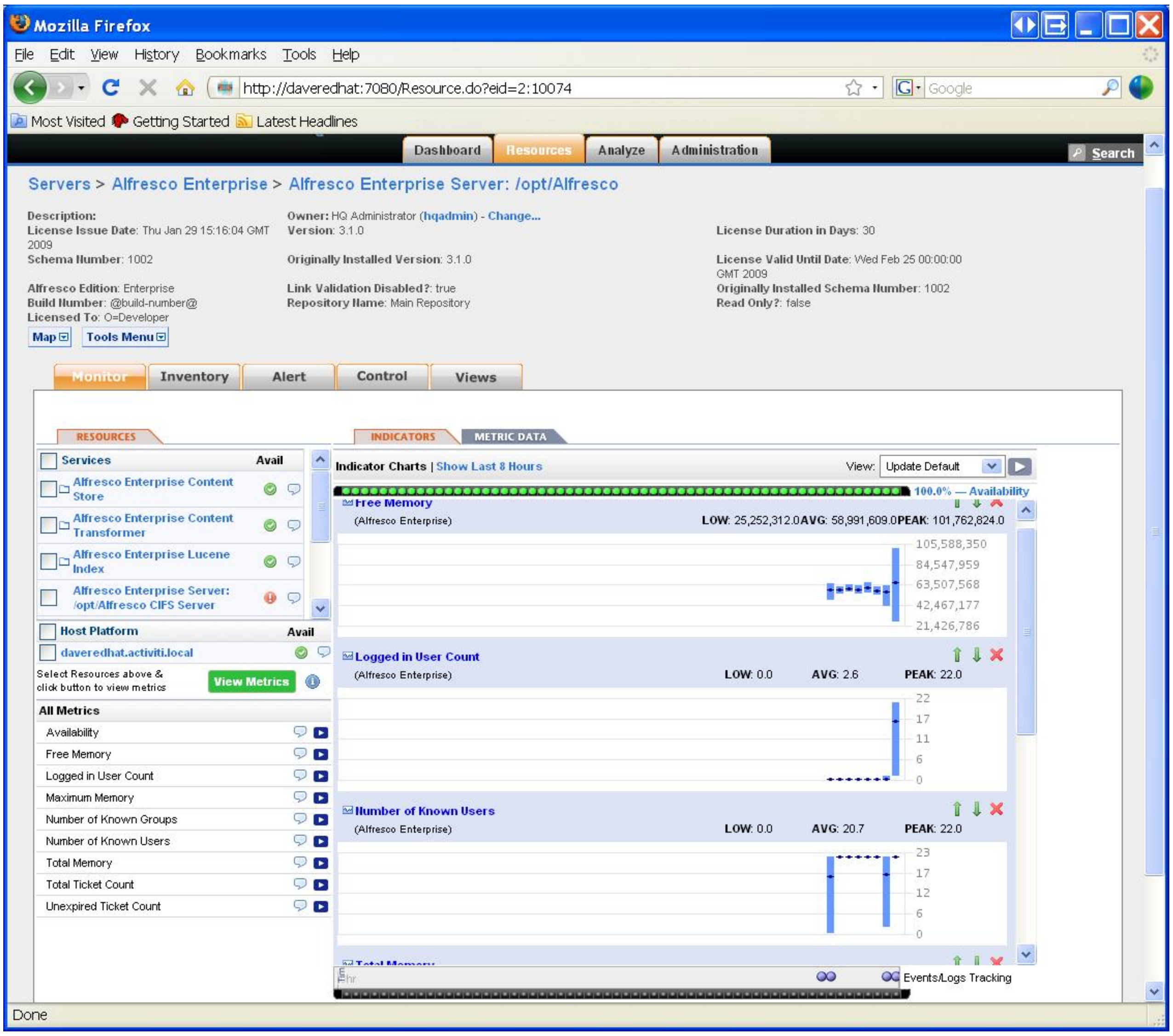
Installing Alfresco Enterprise plug in for Hyperic

Hyperic provides auto-discovery, monitoring of system resources, alerts, charting, and event correlation for problem identification and resolution. The Alfresco Enterprise plug in for Hyperic allows you to auto-discover all Alfresco components and to monitor usage and performance using the Hyperic HQ interface.

Before you install the Alfresco Enterprise plug in for Hyperic, ensure the following:

- Hyperic HQ 4.0 Server is installed and running on your system
- Hyperic HQ Agent is installed on the same machine as Alfresco
- The operating system user running the Hyperic agent and the operating system user running Alfresco both have permissions to read and write each other's file. For example, both users must be running as root, or alternatively, both users must be in the same group, with `umask` set to 2.

The Hyperic installation consists of a server and one or more agents. A Hyperic agent running on the same machine as your Alfresco server can use the Alfresco Enterprise plug in to detect running Alfresco servers on the machine, collecting metrics on availability, performance, and use. The agent sends the inventory and metric data to the Hyperic HQ server, which can be managed using the Hyperic portal.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://daveredhat:7080/Resource.do?eid=2:10074
Most Visited Getting Started Latest Headlines
Dashboard Resources Analyze Administration
Search

Servers > Alfresco Enterprise > Alfresco Enterprise Server: /opt/Alfresco

Description:
License Issue Date: Thu Jan 29 15:16:04 GMT 2009
Schema Number: 1002
Alfresco Edition: Enterprise
Build Number: @build-number@
Licensed To: O=Developer

Owner: HQ Administrator (hqadmin) - Change...
Version: 3.1.0
Originally Installed Version: 3.1.0
Link Validation Disabled?: true
Repository Name: Main Repository

License Duration in Days: 30
License Valid Until Date: Wed Feb 25 00:00:00 GMT 2009
Originally Installed Schema Number: 1002
Read Only?: false

Services	Avail
Alfresco Enterprise Content Store	✓
Alfresco Enterprise Content Transformer	✓
Alfresco Enterprise Lucene Index	✓
Alfresco Enterprise Server: /opt/Alfresco CIFS Server	!

Host Platform	Avail
daveredhat.activiti.local	✓

Select Resources above & click button to view metrics

All Metrics
Availability
Free Memory
Logged in User Count
Maximum Memory
Number of Known Groups
Number of Known Users
Total Memory
Total Ticket Count
Unexpired Ticket Count

Indicator Charts | Show Last 8 Hours View: Update Default

100.0% — Availability

Free Memory (Alfresco Enterprise) LOW: 25,252,312.0 AVG: 58,991,609.0 PEAK: 101,762,824.0

Logged in User Count (Alfresco Enterprise) LOW: 0.0 AVG: 2.6 PEAK: 22.0

Number of Known Users (Alfresco Enterprise) LOW: 0.0 AVG: 20.7 PEAK: 22.0

1. Browse to Alfresco Enterprise Network.
2. Log in using your Network user name and password.
3. Click **Downloads**.
4. Search for and download the following Alfresco installation file:

   `alfresco-enterprise-plugin.xml`
5. Copy the downloaded file to the `hq-plugins` directory in your Hyperic installation.
6. Open your Alfresco startup script.
7. Append the following to the JAVA_OPTS setting:

   (Windows) `-Dalfresco.home=%ALF_HOME% -Dcom.sun.management.jmxremote`

   (Linux) `-Dalfresco.home=${ALF_HOME} -Dcom.sun.management.jmxremote`

   If you installed Alfresco using one of the installation wizards, this is already set for you in the `alfresco.bat` or `alfresco.sh` start up script.
8. Save your start up script.
9. Restart the Hyperic agent using the following command:

   (Linux) `/etc/init.d/hyperic-hq-agent restart`
10. Restart the Alfresco server.

11. In the Hyperic Portal, schedule an auto-discovery. Refer to the Hyperic documentation for details on how to set up auto-discovery.

    The **Alfresco Enterprise Server** is discovered, in addition to all the system resources on the machine.
12. To enable log tracking, click **Resources**, then navigate to the Alfresco Server and select **Inventory**.
13. Under **Configuration Properties**, click **Edit**.
14. Set `server.log_track.enable` to true.
15. Set the `server.log_track.level`, if required.

    Log entries will then be indicated by a blue light at the bottom of the **Monitor** tab.

Auditing Alfresco

Alfresco provides the ability to audit activity. When auditing is enabled, it captures both user and application interaction with the repository at the service layer, without recording all the underlying changes required to achieve this.

There are two implementations of audit available:

- the default audit
- an implementation that accommodates the requirements of Records Management

The default audit records:

- The current authenticated user (including domain)
- The timestamp
- The identifier of the containing transaction
- The key node reference
- The host machine

The implementation of auditing for Records Management can be leveraged regardless of whether or not you have installed the Records Management module.

The Records Management implementation provides features, such as:

- Data is stored in inherently searchable (database-indexed) tables, specific to the type of data
- Any type of data can be produced and passed to the `AuditComponent` (free-form, interceptor-driven, ad-hoc)
- Audit configuration is used to filter, transform, and generate data for eventual storage
- Provides a callback-driven query API

For WCM, web projects automatically capture a full version history of all changes to all assets using the snapshot history, without requiring additional auditing functionality. This does not prevent data from being captured in any part of WCM or Alfresco, but asset metadata history is automatically available in WCM.

Configuring auditing

Auditing is disabled by default. To enable auditing, you need to set auditing in the global properties file.

The new and old auditing implementations are enabled in tandem to allow backwards compatibility with settings in `config/auditConfig.xml`. You can use the new implementation exclusively and configuration by setting a property in the alfresco-global.properties file.

1. Edit the `alfresco-global.properties` file.
2. Add the following setting:

   ```
   # Audit configuration
    audit.enabled=true
   ```

   This setting will enable the default auditing tool.
3. (Optional) To use the Records Management implementation of audit, add the following line:

   `audit.useNewConfig=true`
4. Save the file.
5. Restart the Alfresco server.

Scheduled jobs

Alfresco runs a number of scheduled jobs that assist in the maintenance of a production environment. These jobs are defined in the `<configRoot>/scheduled-jobs-context.xml` file.

Scheduled job	Description
`ftsIndexerTrigger`	Triggers the full text indexing of uploaded or modified documents.
`contentStoreCleanerTrigger`	Launches the `contentStoreCleaner` bean, which identifies, and deletes or purges orphaned content from the content store while the system is running. Content is said to be orphaned when all references to a content binary have been removed from the metadata. By default, this job is triggered at 4:00 am each day. In a clustered environment, this job could be enabled on a headless (non-public) node only, which will improve efficiently.
`nodeServiceCleanupTrigger`	Performs cleanup operations on DM node data, including old deleted nodes and old transactions. In a clustered environment, this job could be enabled on a headless (non-public) node only, which will improve efficiently.
`openOfficeConnectionTesterTrigger`	Maintains the connection with OpenOffice.
`indexBackupTrigger`	Creates a safe backup of the Lucene directories.
`tempFileCleanerTrigger`	Cleans up all Alfresco temporary files that are older than the given number of hours. Subdirectories are also emptied and all directories below the primary temporary subdirectory are removed. The job data must include the `protectHours` property, which is the number of hours to protect a temporary file from deletion since its last modification.
`avmOrphanReaperJob`	Quartz wrapper for OrphanReaper, which is a background thread for reaping nodes that are no longer referenced in the AVM repository. These orphans arise from purge operations.
`avmExpiredContentTrigger`	Searches for expired content in the web project staging area and prompts the last modifier of the content to review it.

Scheduled job	Description
`ehCacheTracerJob`	Collects detailed cache usage statistics and outputs them to the console, depending on how logging has been configured for the server. By default, his job is not activated. To activate this job, uncomment the `scheduler` property.

Creating and managing workflows

This section contains steps for creating a workflow process definition and task model.

What is workflow?

A workflow is a work procedure and workflow steps that represent the activities users must follow in Alfresco to achieve a desired outcome. You can define your own content-oriented workflows. Alfresco provides two different types of workflow: simple workflow and advanced workflow.

Simple workflow

Simple workflow defines content rules for a space. The content rule dictates how the content entering, leaving, or currently residing in the space is managed. Each workflow definition is restricted to a single state.

Advanced workflow

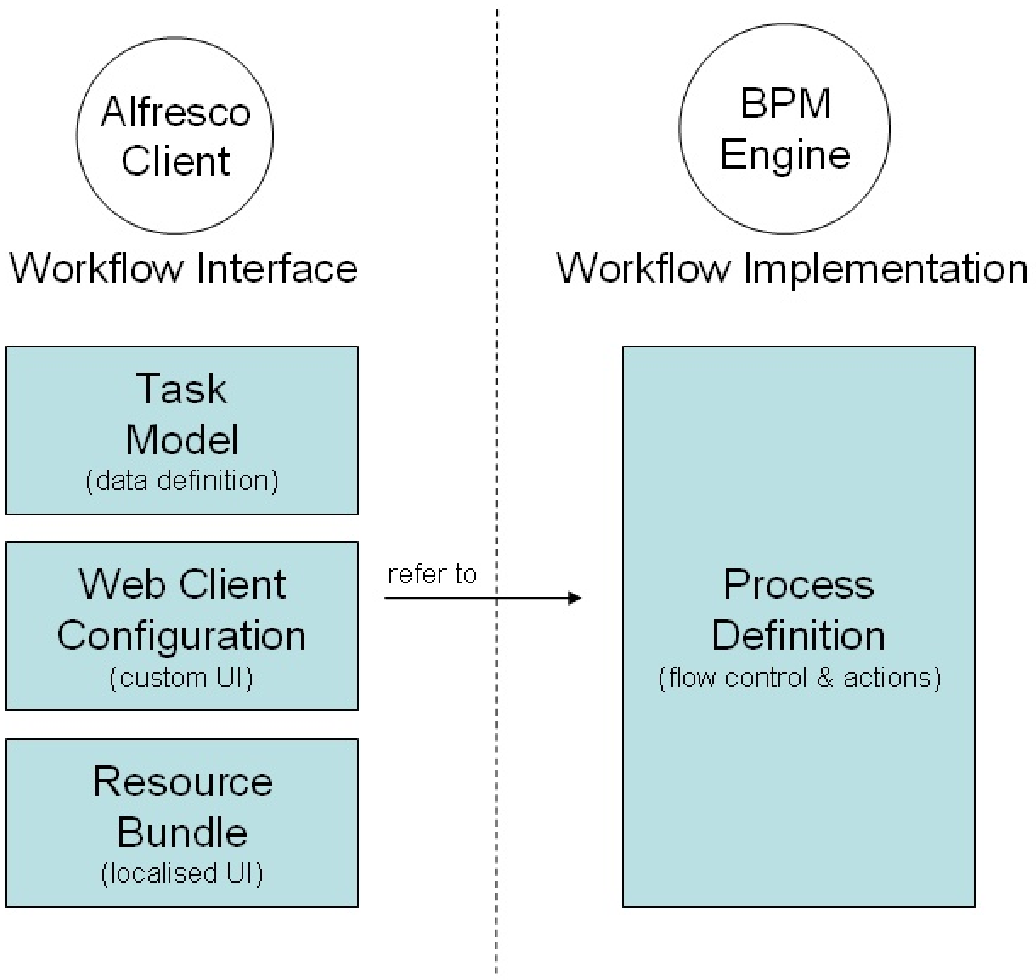
Advanced workflow is any workflow constructed using the Alfresco embedded workflow engine. Alfresco includes two out-of-the-box workflows, which are both basic examples of advanced workflows:

- Adhoc Task (for assigning a task to a colleague)
- Review & Approve (for setting up review and approval of content)

In both examples, the content items are attached to the workflow.

Advanced workflow artifacts

An Alfresco "advanced" workflow is comprised of the following artifacts:


Alfresco Client
Workflow Interface
BPM Engine
Workflow Implementation
Task Model (data definition)
Web Client Configuration (custom UI)
Resource Bundle (localised UI)
refer to
Process Definition (flow control & actions)

Administration features

Three workflow administration features are available. These are:

- Workflow Definition Language
- Workflow Interfaces
- Workflow Tools

Implementation

- Workflows are defined using jPDL process language.
- You can create process definitions manually in XML files or you can create the definitions graphically using the JBoss jBPM Process Designer. These definitions are automatically saved in XML files.
- You can specify the content of the workflow task dialogs that will be displayed in the UI.
- Workflow definitions are easily localizable.
- Optionally, you can use Process Designer to deploy the process definitions to the Alfresco server in real time.

Creating a skeleton process definition

The process definition allows you to:

- Group tasks into task nodes
- Assign task nodes to people using swimlanes (example)

- Define system actions that can occur during the transition from one task node to the next

Process definition methods

A process definition can be either an XML file or a process archive. Both of these can be created using jBPM Process Designer. You can also create the XML file manually.

The Process Designer allows you to switch between diagram mode and XML editor mode. This means that you can initially create a file using diagram mode, then switch to editor mode for viewing and fine tuning.

When you redeploy updated process definitions, you can do so manually or using the Process Designer. In either case, the previously deployed process definition is versioned and kept alive to allow existing "in-progress" workflows to continue executing.

Creating a skeleton process definition manually

The high-level parts of the process definition are shown in the Adhoc task example. The complete process definition is shown in the Process definition for an ad hoc task.

Assume that the person who starts the workflow is **initiator**, and the person who is assigned the workflow is **assignee**.

The initiator can specify the following information through the UI:

- Task description
- Due date (optional)
- Priority
- Request for email notification of task completion (optional)

Setting up JBoss jBPM Process Designer

There are two ways to set up Process Designer:

- Download and install a single zipped file from the Alfresco website. This will install a new, standalone Process Designer.

or

- Do the steps manually. You might do this if you want to embed jBPM Process Designer in your own Eclipse. In this case, you need Eclipse 3.2 (or later).

Installing the Process Designer package

The Process Designer is a zipped that contains the following:

- Eclipse 3.2.1
- jBPM 3.1.2
- jBPM Process Designer 3.0.11
- Eclipse projects for the following workflows: Review & Approve; Ad hoc Task

Unzip the file to any `<install>` folder. There is a ReadMe file in the root of the `<install>` folder with installation and configuration instructions.

Deploying Eclipse

This task describes how to download and install Eclipse.

1. Download **Eclipse SDK 3.2** from http://www.eclipse.org/downloads.

2. Unzip the downloaded file in the required directory.
3. To run Eclipse, follow the advice in the release notes, `readme_eclipse.html`.

The following are good sources for Eclipse information:

- http://www.cs.laurentian.ca/badams/c1047/eclipse-tutorials/index.html
- http://eclipse-tutorial.dev.java.net/http://eclipse-tutorial.dev.java.net/
- http://wiki.eclipse.org/index.php/Eclipse_FAQs

Deploying JBoss jBPM 3.1.2

This task describes how to download and install JBoss jBPM.

Documentation for JBoss jBPM is located at:

- http://docs.jboss.com/jbpm/v3/userguide/
- http://docs/jboss/com/jbpm/v3/gpd/

1. Download JBoss jBPM 3.1.2 from:

   http://www.jboss.com/products/jbpm/downloads
2. Unzip the downloaded file in the required directory.
3. Start Eclipse.
4. Ensure you are in the correct workspace.
5. Click **File** > **Import**.
6. In the Select dialog box, click **General > Existing projects into workspace**.
7. Click **Next**.
8. In the Import Projects dialog box, enable **Select root directory**.
9. Browse to the jBPM root directory, for example, `jbpm-3.1.2`.
10. Enable **Copy projects into workspace**.

Creating a task model

Perform this task to create a task model, which is similar to a content model.

1. Derive your model from the base task model:
   - `<import uri="http://www.alfresco.org/model/bpm/1.0" prefix="bpm"/>`
2. For each `<task>` in the process definition, create a `<type>` in the task model. `<type name>` must be identical to `<task name>`:
   - `<type name="wf:submitAdhocTask">`
   - `<type name="wf:adhocTask">`
   - `<type name="wf:completedAdhocTask">`
3. For each type, describe the properties:
   - `<property name="wf:notifyMe">`
4. For each type, define aspects:
   - `<aspect>bpm:assignee</aspect>`

   You do not need to create an association for `initiator`. `initiator` is a process instance variable that is available in any workflow. `initiator` is a repository node (cm:person) that represents the person who initiated the workflow.

5. Add the details for properties and associations.

Deploying the task model

There are two ways to deploy a task model:

- Using the workflowDeployer bean
- Deploying as a content model

Using the workflowDeployer bean

A workflow task model may be deployed using the workflowDeployer bean.

```
<bean id="myworkflows.workflowBootstrap" parent="workflowDeployer">
   <property name="workflowDefinitions">
      ...
   </property>
   <property name="models">
      <list>
         <-- Task Model associated with above process definition -->
         <value>alfresco/workflow/adhocModel.xml</value>
      </list>
   </property>
</bean>
```

Deploying as a content model

The task model is a content model, so it may be deployed like any other content model.

An example configuration snippet follows:

```
<bean id="adhocWorkflow.dictionaryBootstrap" parent="dictionaryModelBootstrap"
 depends-on="dictionaryBootstrap">
   <property name="models">
      <list>
         <value>alfresco/model/adhocTaskModel.xml</value>
      </list>
   </property>
</bean>
```

Adding behavior to a skeleton process definition

You can add behavior to a skeleton process definition to describe the data flow, task assignments, and repository actions.

Based on the task model, the complete process definition is shown in the following code snippet.

Note:

- The swimlane assignee is mapped to the process variable `bpm_assignee`.
- A task-create event is used to initialize the due date and priority of the ad hoc task.
- The ad hoc task completed transition event is used to call Alfresco JavaScript that sends an email.

```
<?xml version="1.0" encoding="UTF-8"?>

<process-definition xmlns="urn:jbpm.org:jpdl-3.1" name="wf:adhoc">

  <swimlane name="initiator"/>

  <start-state name="start">
    <task name="wf:submitAdhocTask" swimlane="initiator"/>
    <transition name="" to="adhoc"/>
  </start-state>
  <swimlane name="assignee">
    <assignment actor-id="#{bpm_assignee.properties['cm:userName']}"/>
  </swimlane>

  <task-node name="adhoc">
    <task name="wf:adhocTask" swimlane="assignee">
      <event type="task-create">
        <script>
          if (bpm_workflowDueDate != void)
          {
            taskInstance.dueDate = bpm_workflowDueDate;
          }
          if (bpm_workflowPriority != void)
          {
            taskInstance.priority = bpm_workflowPriority;
          }
        </script>
      </event>
    </task>
    <transition name="" to="completed">
      <action class="org.alfresco.repo.workflow.jbpm.AlfrescoJavaScript">
        <script>
          if (wf_notifyMe)
          {
            var mail = actions.create("mail");
            mail.parameters.to = initiator.properties["cm:email"];
            mail.parameters.subject = "Adhoc Task " +
 bpm_workflowDescription;
            mail.parameters.from = bpm_assignee.properties["cm:email"];
            mail.parameters.text = "It's done";
            mail.execute(bpm_package);
          }
        </script>
      </action>
    </transition>
  </task-node>

  <task-node name="completed">
    <task name="wf:completedAdhocTask" swimlane="initiator"/>
    <transition name="" to="end"/>
  </task-node>

  <end-state name="end"/>

</process-definition>
```

Configuring UI workflow dialogs

Dialogs are used in Explorer to present tasks to users.

You can modify default dialogs to:

- Control which task properties display.
- Control which task properties are read-only and mandatory.
- Control how each task property is rendered in the dialog.
- Present a custom dialog for each kind of task in the workflow.

Alfresco Explorer already provides a sophisticated configuration mechanism for controlling dialogs. This same mechanism extends to workflow tasks.

Deploying a process definition

You can deploy a process definition manually or use the Process Designer. For information on the Process Designer, see Setting up JBoss jBPM Process Designer.

For manual deployment, the Alfresco server must be shut down. Process definitions are deployed when Alfresco starts.

Deploying a process definition manually

You can use the `workflowDeployer` bean to deploy process definitions (refer to the following code snippet).

The bean must be in a file in the `<extension>` directory.

```
<bean id="myworkflows.workflowBootstrap" parent="workflowDeployer">
  <property name="workflowDefinitions">
    <list>
      <props>
        <prop key="engineId">jbpm</prop>
        <prop key="location">alfresco/workflow/adhoc_processdefinition.xml</
prop>
        <prop key="mimetype">text/xml</prop>
      </props>
    </list>
  </property>
</bean>
```

Deploying a process definition

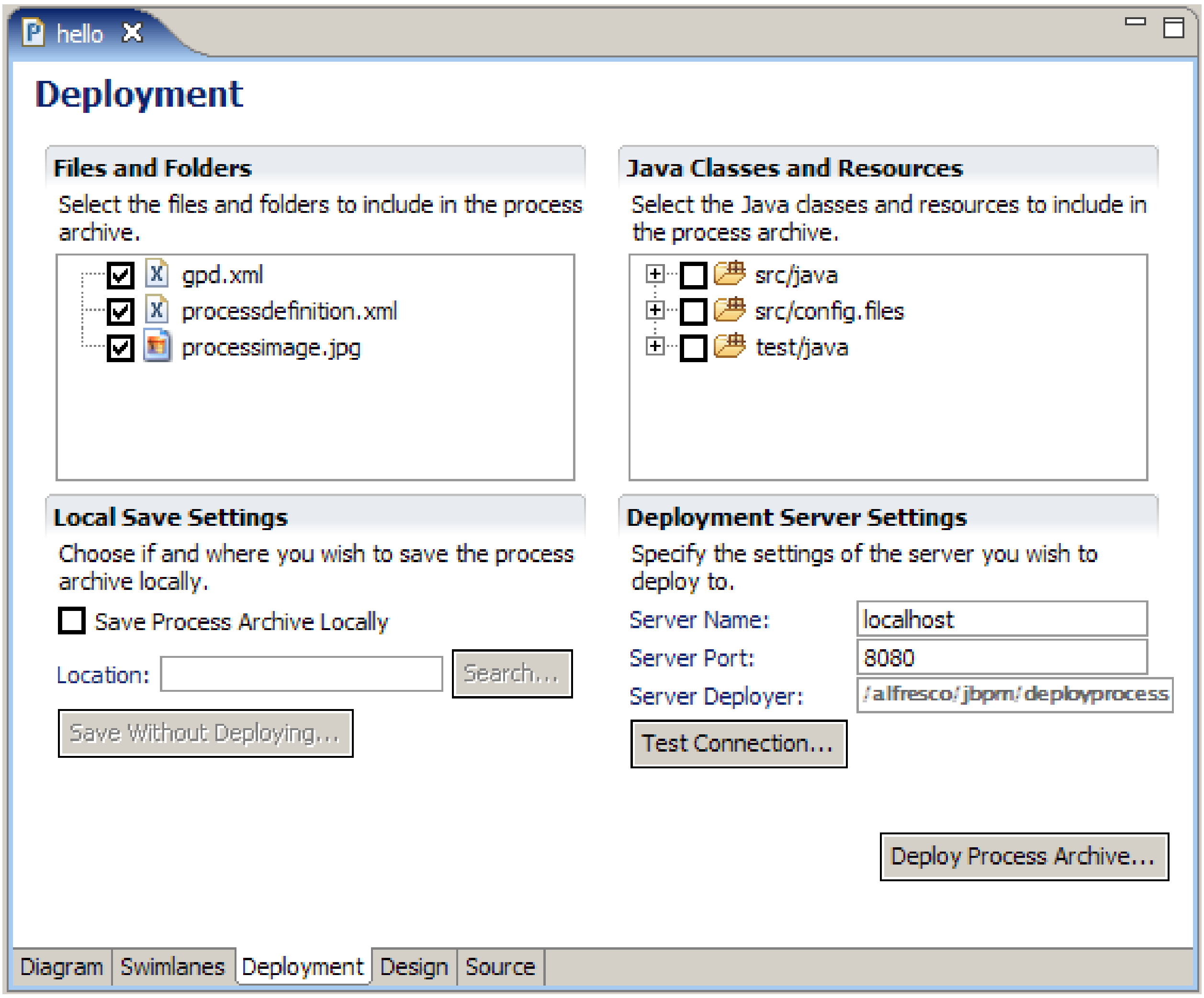
Use the Process Designer to deploy a process definition.

1. Ensure the Alfresco server is running.
2. At the bottom of the Process Designer window, click the **Deployment** tab.
3. In the **Deployment Server Settings** block in the **Server Name** field, enter the name of the machine where Alfresco is installed.
4. In the **Server Port** field, enter the port number assigned to Alfresco (default: 8080)
5. In the **Server Deployer** field, enter `/alfresco/jbpm/deployprocess`.
6. Click **Test Connection**.
7. If everything is okay, click **Deploy Process Archive**.

   You do not need to restart the Alfresco server to activate the newly deployed process definition. Using the jBPM Process Designer allows quick turnaround while testing new process definitions.

The process definition is now available for Alfresco.

The following image shows using the Process Designer to deploy a process definition.



Deployment

Files and Folders

Select the files and folders to include in the process archive.

- ✓ gpd.xml
- ✓ processdefinition.xml
- ✓ processimage.jpg

Java Classes and Resources

Select the Java classes and resources to include in the process archive.

- ☐ src/java
- ☐ src/config.files
- ☐ test/java

Local Save Settings

Choose if and where you wish to save the process archive locally.

☐ Save Process Archive Locally

Location:

Search...

Save Without Deploying...

Deployment Server Settings

Specify the settings of the server you wish to deploy to.

Server Name:	localhost
Server Port:	8080
Server Deployer:	/alfresco/jbpm/deployprocess

Test Connection...

Deploy Process Archive...

Sizing guidelines

Deciding how many servers should be used to run Alfresco is difficult to estimate. This section contains practical experience and some basic assumptions about Alfresco sizing. Each deployment is different and the ideal configuration can only be based on an understanding of the requirements, use cases, and preferred environment.

This section focuses on Alfresco software only and not on dependencies, such as the database or storage solutions.

The sizing guidelines assume that all servers meet the following hardware configuration:

- Processor: 64-bit Intel Xeon 2.8Ghz or better
- Memory: 8GB RAM
- Disk: 100GB (minimum)
- Operating System: 64-bit Red Hat 5 or later

If the preferred deployment environment is not equivalent to this, care must be taken to adjust the anticipated performance figures to match the actual environment.

Sizing in a production environment

Minimal server configurations are based on whether clustering is required. Clustering is often used to improve performance and to help guarantee high-availability and fault-tolerance. This guide does not go discuss specific instructions for configuring a high-availability environment.

Non-clustered

In this configuration, there is one server running Alfresco while another runs the database. Alfresco and the database can co-exist on development environments and small deployments.

Clustered

In this configuration, two or more Alfresco servers share a common database and file-system, each on its own dedicated server for a total minimum of four servers, two running Alfresco, one running the database, while the fourth serves as the shared file server.

Sizing methodology

This section describes the basic methodology for sizing.

The most accurate means of estimating size is not concerned with the number of users but rather the overall load on the environment. This methodology focuses on the Transactions Per Second (TPS). Transaction is defined to mean the number of basic repository operations (create, read, update, delete or CRUD) that the server can typically handle under very high load, while still ensuring adequate responsiveness. Calculations are weighted toward Create operations, as those tend to be the most computationally expensive tasks.

Minimally, a single un-optimized server should be able to handle between 3-7 transactions per second during periods of highest usage (usage peaks). Through JVM tuning, and with network and database optimizations, this number can rise to over 15 transactions per second.

Sizing formula

The general goal is to achieve a certain throughput of documents in a certain amount of time during the periods of highest utilization, therefore the following formula is a starting point for estimating size:

```
# of Servers = Desired Throughput in TPS (DTW) / Average TPS per Server (ATPS)
An example follows:
Desired Throughput per 5-day Week (DTW):
1,000,000 documents
Desired Throughput in TPS assuming 12 hours of daily operation (DT):
DTW / (5 weekdays * 12 hours * 3600 seconds per hour) = 1,000,000 /
 (5*12*3,600)
Total: 4.6 TPS
Number of Servers assuming 3 TPS per server:
DT / ATPS = 4.6 / 3 = 1.5 Servers round up to 2
```

Formula variations

Invariably, someone will still want to map Desired Throughput (DT) to the number of users so we offer the following formula to estimate the value:

```
DT = (# of Users * Average Transactions per User per Day) / (Average Hours in
 Workday * 3,600 Seconds per Hour)
Example:
DT = (500 Users * 300 Transactions per Day) / (8 hours * 3,600)
DT = 5.2 TPS
```

Troubleshooting

This chapter provides information to help you solve issues that may arise when installing and configuring Alfresco.

For additional help, refer to the following:

- Alfresco Enterprise Knowledge Base
- Administration Console in Alfresco Explorer to view various installation and setup information
- Alfresco Installation forum (http://forums.alfresco.com/)

Debugging an Alfresco installation

When developing add-ins, fixing bugs, or changing Alfresco from the source code, it is helpful to debug an instance of Alfresco running on a standard application server. This section outlines the steps needed to configure Alfresco and Eclipse to provide a real-time view of the server and to troubleshoot issues by stepping through the code line by line.

To debug a running Alfresco server, you must connect to the JVM in which Alfresco is running. The following steps configure the JVM to expose an interface for this connection, and then configure Eclipse to connect to and control that JVM.

Configuring the JVM

This task describes how to configure the JVM to expose an interface for connection to the Alfresco server.

Before you start, you must:

- Have a fully installed, configured, and running instance of Alfresco. These steps assume you are using Tomcat on Windows, but the steps are similar for other application servers on other systems.
- Have an IDE installed. These steps describe how to configure Eclipse, which must be installed first (http://www.eclipse.org/downloads)
- Download and install the Alfresco source code from http://wiki.alfresco.com/wiki/Alfresco_SVN_Development_Environment.
- Ensure the source code is the same version as the installed Alfresco server.

1. Verify the Alfresco server is not running.
2. Edit the JVM options used to start the Alfresco Tomcat instance. If you are using the default `alf_start.bat` to start Alfresco, perform the following:
    a. Edit `alfresco.bat` in a text editor.
    b. Find the line: `set JAVA_OPTS=%JAVA_OPTS% -server –X…`
    c. On the next line, add the following on one line:
    ```
    set JAVA_OPTS=%JAVA_OPTS% -server -Xdebug
    -Xrunjdwp:transport=dt_socket,server=y,suspend=n,
    address=8082
    ```
    where address is a port for your system
3. Save the file and close the editor.

Configuring Eclipse

This task describes how to configure Eclipse to connect to and control the JVM.

1. From the Run menu, choose the **Open Debug** dialog.
2. Right-click **Remote Java Application** and select **New**.
3. In the Name box, type `Debug Local Tomcat Alfresco`.
4. Next to Project, click **Browse**, and select **Web Client**. If this is not available as an option, ensure your source code matches that of your server.
5. In Connection Properties, enter the Port number you added to the `alf_start.bat` file.
6. Check **Allow Termination of remote VM** if you want to be able to stop the Alfresco server from the Eclipse console.
7. Click **Apply** to save the configuration.

You have configured Alfresco and Eclipse. Next, you can start the Alfresco server and start the Debug configuration in Eclipse. Eclipse will connect to the Alfresco JVM. From the Java perspective in Eclipse, you can expand the "core" or "web client" packages, open the class files you are interested in, and set breakpoints for the Alfresco server to stop at. From the Debug perspective, you can then interrogate specific variables from the Alfresco server "live", and step through the source code line by line.

Debugging an upgrade

The startup log is important to help Alfresco Support diagnose any issues that might arise as a result of the upgrade.

1. Immediately after starting the Alfresco server, make a copy of the `alfresco.log` file.
2. Make a copy of the temporary files that contain the SQL statements executed by the upgrade.

   The locations of the temporary files are written to the `alfresco.log` file.
3. Submit the log file and SQL temporary files to Alfresco Support.

Setting log levels

The `log4j.properties` file lets you configure logging levels to provide debugging information when troubleshooting. To set up logging policies, you must prepend `log4.logger` to the class name you want to log to, and set the logging level.

1. Find a class definition in the Spring config files, or uncomment an existing configuration for the component.
2. Add the class definition to the `log4j.properties` file.
3. Prepend `log4j.logger` onto the class definition in the log file. For example:

   ```
   log4j.logger.org.springframework.beans.factory.config.PropertyPlaceholderConfigure
   ```

   This example sets the logging level to `debug`. Other logging levels include `all`, `info`, `warn`, `error`, and `fatal`.
4. Restart the server.

   During startup, all the properties display.

Testing and debugging links

The `<configRoot>/log4j.properties` file lets you set the level of logging based on the amount of information you need.

- To enable debugging for the background process that continually checks the links in a web project, remove the comment from the following line:

```
#log4j.logger.org.alfresco.linkvalidation.LinkValidationServiceImpl=debug
```

- To enable debugging for the action to run when performing a link validation check, add the following line:

  ```
  log4j.logger.org.alfresco.linkvalidation.LinkValidationAction=debug
  ```

- To enable debugging for the link validation report dialog, add the following line:

  ```
  log4j.logger.org.alfresco.web.bean.wcm.LinkValidationDialog=debug
  ```

Error messages

This section lists possible issues you may encounter when installing Alfresco and suggests possible remedies.

ImageMagick

Error message on the console:

```
ERROR [AbstractImageMagickContentTransformer]
JMagickContentTransformer not available:
ERROR [AbstractImageMagickContentTransformer]
ImageMagickContentTransformer not available:
Failed to execute command: imconvert ...
```

These issues will not cause the server to fail. Alfresco is reporting that external document transformation engines are not available for use by the server. You can remove the transformation references if they are not required.

`JAVA_HOME`

Make sure the `JAVA_HOME` variable is set correctly for your Java installation.

FTP Socket

Error message on server startup:

```
ERROR [protocol] FTP Socket error
```

```
java.net.BindException: Address already in use:
JVM_Bind at
```

```
java.net.PlainSocketImpl.socketBind(Native Method)
```

Check to see if you have any services running against port 8080 for the Alfresco server or port 21 for the Alfresco FTP integration.

Troubleshooting an upgrade

This section provides help for diagnosing and resolving any issues that might arise as a result of an upgrade.

1. Open the `alfresco.log` file.
2. Make a copy of the `alfresco.log`.
3. In `alfresco.log`, note the locations of the temporary files containing the SQL statements executed during the upgrade, and make a copy of these temporary files.
4. Submit the log file and temporary files to Alfresco Support.

   By in-place upgrading a copy of the repository, rolling back to the previous version in the event of an upgrade failure is quick and painless. The original installation, configuration, and repository are untouched by this process, so it can simply be restarted. This process also allows for the upgrade to be performed any number of times.

Troubleshooting OpenOffice subsystems

This section provides help for troubleshooting the OpenOffice subsystems.

1. Enable the following log4j property to debug:

   ```
   log4j.logger.org.alfresco.enterprise.repo.content=DEBUG
   ```

   The OOoDirect debug entry is: `log4j.logger.org.alfresco.repo.content.transform=DEBUG`.

2. If Tomcat is not shutdown gracefully, the `soffice.bin` process may not be stopped. This can result in errors when starting Tomcat with port 8080 being is use. If this occurs, manually kill the `soffice.bin` process.

3. You may see a failure to connect error message, for example:

   ```
   INFO: ProcessManager implementation is WindowsProcessManager
   org.artofsolving.jodconverter.office.OfficeProcess start
   INFO: starting process with acceptString
    'socket,host=127.0.0.1,port=8101,tcpNoDelay=1'
   and profileDir 'C:\Alfresco\tomcat\temp
   \.jodconverter_socket_host-127.0.0.1_port-8101'
   org.artofsolving.jodconverter.office.OfficeProcess start
   INFO: started process
   ERROR [repo.content.JodConverterSharedInstance] Unable to start
    JodConverter library.
   The following error is shown for informational purposes only.
   org.artofsolving.jodconverter.office.OfficeException: failed to start and
    connect
   ```

   If the OpenOffice process takes more than 30s to fully start up, then Alfresco fails to connect to it. If this occurs, manually kill the `soffice.bin` process before attempting to restart the Jodconverter subsystem.

   The next time that you start OpenOffice, it usually starts fast enough to connect (this is due to operating system caching).

4. If the OpenOffice home location is incorrect, the Jodconverter subsystem will still start, but no OpenOffice process will be running or connected. The error may be reported in the console but not in the `alfresco.log` file.

5. When restarting the Jodconverter subsystem using JMX, you need to set the enabled property to true (this will also stop the JOD subsystem if it is running); then use the **start** operation to start the Jodconverter subsystem with the new property settings.

Troubleshooting the JMX Dumper

This section provides help for troubleshooting the JMX Dumper.

Invoking the JMX Dumper may result in a stack trace in the log file. When you open `jmx-dumper`, it is trying to find a data source defined in the `web.xml` file. (`<res-ref-name>jdbc/dataSource</res-ref-name>`), but this data source is not declared in the `alfresco.xml` file.

To prevent this logging message for appearing, you can configure the data source in the `$CATALINA_BASE/conf/[enginename]/[hostname]/alfresco.xml` file.

Troubleshooting NFS

This section provides help for diagnosing and resolving any issues that might arise when configuring NFS.

User ID and Group ID

An issue with the NFS server (JLAN-11) transposes the GID and UID of the NFS client user, meaning that Unix accounts that have a user-id that differs from the group-id will not gain authentication. The Alfresco-NFS server will deny them access. The user will only see `ls: /mnt/alfresco: Input/output error`. This issue lasted this long presumably because so many Linux distributions create a new group for each new user, unless told otherwise. Though the bug is declared closed, it has yet to filter down to SVN, and `org.alfresco.filesys.server.auth.AlfrescoRpcAuthenticator.authenticateRpcClient(int, RpcPacket)` still reads the GID before the UID.

NFS server port number is not ephemeral

If the `NFSServerPort` property is not given, it defaults to 2049. This is likely to conflict with a native NFS server, if any. The portmapper daemon, when properly used, removes any dependency upon a well known port number, however neither Alfresco nor any native NFS server seem to use this functionality.

Running native and Alfresco NFS servers on the same host

If you wish to run the native server along side the Alfresco NFS server, you cannot depend upon the portmapper, as there is a 50 percent chance that it will retain the native NFS details. When using `nfs-utils-1.0.10` version on Linux, `mount.nfs` will defer to the portmapper for the port-number, version-number, and protocol of the NFS server in question. Only if all three of these are supplied on the command line will the mount command directly contact the Alfresco NFS server. Failing this, `mount.nfs` will fail as it cannot find the server you have described in the portmapper table. You must therefore configure both `MountServerPort` and `NFSServerPort` to known values above 1024. Afterward the following command line should succeed:

```
mount -oport=yourNfsPort,mountport=yourMountPort,proto=tcp
 yourFfsServerName:/alfresco /mnt/alfresco/
```

The `proto` option may be either `tcp` or `udp`. It is desirable to have functionality to resolve the NFS server required by the volume requested, however, the portmapper only searches for the NFS server on the version and protocol.

Troubleshooting CIFS

This section provides help for diagnosing and resolving any issues that might arise when configuring CIFS.

Password Error

Sometimes, when connecting to an instance of Alfresco Share, the login dialog appears several times until finally taking effect. This problem can be caused by the Share connecting to the Windows file server that is running on native SMB/port 445 rather than trying to connect via NetBIOS.

Native SMB collisions

Native SMB can be disabled by adding the following registry key:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetBT\Parameters]
 "SMBDeviceEnabled"=dword:00000000
```

Reboot the system after creating this key. Setting the value to one or deleting the registry key will restore native SMB support.

The SMBDeviceEnabled registry key does not seem to be recognized on Vista and Windows 2008. In order to stop native SMB being used when the Alfresco CIFS server is being run under

Vista or Windows 2008, the firewall rules can be updated to block inbound connections on port 445.

To set up the Windows firewall on the system running the Alfresco CIFS server:

1. Run the Windows Firewall with Advanced Security application:
   - (Vista) go to **Control Panels > Administrative Tools**
   - (Windows 2008) go to **Start > Administrative Tools**
2. Click on the **Inbound Rules** item in the left hand column.
3. Scroll down to the **File and Printer Sharing** rules and enable the following:
   - File And Printer Sharing (`NB-Datagram-In`), File And Printer Sharing (`NB-Name-In`) and File And Printer Sharing (`NB-Session-In`)
   - The File And Printer Sharing (`SMB-In`) rule should be disabled, this blocks the native SMB/port 445 traffic
   - Other File And Printer Sharing (...) rules are not required and can be left as is

Troubleshooting LDAP duplicate person entries

It is possible that you may find duplicate person entries using LDAP.

Duplicate person entries may occur because:

- A configuration error in the LDAP import: each import creates a new person. The query to find the existing person does not match the person you expect. Therefore, this usually means the `UID` attribute is not correct or consistent over configurations. There is a task to simplify the configuration and make this less likely.
- Before the first LDAP import completes, a person is created automatically as the user logs in. The LDAP import will not see the auto created person - a duplicate will be present after the import completes. One way to avoid this is to disable the auto creation of people.
- There are simultaneous logins for the same person which auto creates the person details

Duplicate person entries can now be handled in a number of ways:

- Find the best match, use it, and then leave all entries as they are
- Find the best match, and then fix the other UIDs to be unique by appending a GUID
- Find the best match and delete all others

The following method describes how to delete duplicates that can no longer be removed using the UI:

1. Create a new Javascript in **Data Dictionary > Javascripts** with the following:
   ```
   people.deletePerson("UserNameToDelete");
   ```
2. Browse to **Company Home > More Actions > View Details**.
3. On the details screen, select **Run Action > Execute Script**, and then select the script that you created.

OpenLDAP tips

This section shows a sample configuration file.

There are a number of things to note:

- The maximum number of results returned has been increased from the default of 500 that even applies to paged results. See the OpenLDAP documentation on limits. If you have more than 500 users or groups this would be an issue.

- Digest authentication has been configured to map from a user ID to the corresponding distinguished name. See the example data.
- Passwords are in clear text (so that any authentication mechanism can be used). It is possible they can be in the correct hashed form for the MD5 digest to work.

```
See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
include  /usr/local/etc/openldap/schema/core.schema
include  /usr/local/etc/openldap/schema/cosine.schema
include  /usr/local/etc/openldap/schema/inetorgperson.schema

# Define global ACLs to disable default read access.

# Do not enable referrals until AFTER you have a working directory
# service AND an understanding of referrals.
#referral  ldap://root.openldap.org

pidfile   /usr/local/var/run/slapd.pid
argsfile   /usr/local/var/run/slapd.args

# Load dynamic backend modules:
# modulepath /usr/local/libexec/openldap
# moduleload back_bdb.la
# moduleload back_ldap.la
# moduleload back_ldbm.la
# moduleload back_passwd.la
# moduleload back_shell.la

# Sample security restrictions
# Require integrity protection (prevent hijacking)
# Require 112-bit (3DES or better) encryption for updates
# Require 63-bit encryption for simple bind
# security ssf=1 update_ssf=112 simple_bind=64

# Sample access control policy:
# Root DSE: allow anyone to read it
# Subschema (sub)entry DSE: allow anyone to read it
# Other DSEs:
#  Allow self write access
#  Allow authenticated users read access
#  Allow anonymous users to authenticate
# Directives needed to implement policy:
# access to dn.base="" by * read
# access to dn.base="cn=Subschema" by * read
# access to *
# by self write
# by users read
# by anonymous auth
#
# if no access controls are present, the default policy
# allows anyone and everyone to read anything but restricts
# updates to rootdn.  (e.g., "access to * by * read")
#
# rootdn can always read and write EVERYTHING!

#######################################################################
# BDB database definitions
#######################################################################

database  bdb
suffix  "dc=company,dc=com"
rootdn  "cn=Manager,dc=company,dc=com"
# Cleartext passwords, especially for the rootdn, should
# be avoid.  See slappasswd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
# This is secret ....
```

```
rootpw          {SSHA}u9AUUYOSVX6idlXcwyYOAG6G84oHFpvG
# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
directory  /usr/local/var/openldap-data
# Indices to maintain
index  objectClass  eq

# Clear text to allow hashing
password-hash  {CLEARTEXT}

# SASL mappings for md5 digest authentication
# Extract the user id and use as the search key

authz-regexp
   uid=([^,]*),cn=digest-md5,cn=auth
   ldap:///dc=company,dc=com??one?(uid=$1)

authz-regexp
   uid=([^,]*),cn=company.com,cn=digest-md5,cn=auth
   ldap:///dc=company,dc=com??one?(uid=$1)

# Tweaks to increase the result set size and max query time

sizelimit 50000
timelimit 3600
```

The following is a very simple example LDIF file that defines People and Groups Organizational units and some example users and groups.

```
# Initial directory contents
dn: dc=company,dc=com
dc: company
objectClass: top
objectClass: domain

dn: ou=People,dc=company,dc=com
ou: People
objectClass: top
objectClass: organizationalUnit

dn: ou=Groups,dc=company,dc=com
ou: Groups
objectClass: top
objectClass: organizationalUnit

dn: uid=fullname,ou=People,dc=company,dc=com
objectclass: inetOrgPerson
sn: Name
cn: Full Name
userPassword: inClearText
telephoneNumber: 1234567890
uid: fullname
givenName: Full
mail: full.name@company.com
o: Company Software Inc.

dn: uid=walrus,ou=People,dc=company,dc=com
objectclass: inetOrgPerson
sn: Rus
cn: Wal Rus
userPassword: inClearText
telephoneNumber: 1234567890
uid: walrus
givenName: Wal
mail: wal.rus@company.com
o: Company Software Inc.
```

```
dn: cn=Group One,ou=Groups,dc=company,dc=com
objectclass: groupOfNames
cn: Group One
member: uid=fullname,ou=People,dc=company,dc=com

dn: cn=Group Two,ou=Groups,dc=company,dc=com
objectclass: groupOfNames
cn: Group Two
member: cn=Group One,ou=Groups,dc=company,dc=com
member: uid=walrus,ou=People,dc=company,dc=com
```

Active Directory tips

This section describes the tips for using Active Directory with the LDAP synchronization.

- You may need to give special permissions in the Active Directory to the account that you are using to do the LDAP bind (as configured in ldap.synchronization.java.naming.security.principal). To do this, open Active Directory Users and Computers, right click on the domain, and select "Delegate Control..." Click "Next", then select the user that you are using for the LDAP bind and click "Next". The permission that they will need is on the next screen "Read all inetOrgPerson information."
- The example URL in ldap.authentication.java.naming.provider.url does not use SSL. SSL is recommended for production systems. You'll need to switch the port from 389 (below, non-SSL) to 636 for SSL.
- It is often helpful to screen out non-user accounts and disabled accounts. The default user queries in the ldap-ad subsystem type do this by checking bit fields on the userAccountControl attribute. For example:

  ```
  userAccountControl:1.2.840.113556.1.4.803:=512
  ```

Troubleshooting SMTP inbound email using StartTLS

For StartTLS support to work for inbound email, you must configure SSL for Java.

To identify whether you are having this problem, enable `DEBUG` logging for the class `org.subethamail` in your `log4j.properties` file.

```
startTLS() failed: no cipher suites in common
```

The following process outlines one methodology for creation of a self-signed certificate. However, this may differ between JVM vendors, so consult your JVM documentation for more information.

1. Create a suitable key and certificate:

   ```
   keytool -genkey -keystore mySrvKeystore -keyalg RSA
   ```

2. Add the following somewhere in your Tomcat configuration. In RHEL 5, this file would be located at `/etc/tomcat5/tomcat5.conf`. For example:

   ```
   JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStore=mySrvKeystore -
   Djavax.net.ssl.keyStorePassword=123456"
   ```

This methodology explains how to create a self-signed certificate only. SSL vendors can provide certificates signed by an authority and may be more suitable for production use.

Reference

This chapter provides additional useful information for Alfresco administrators.

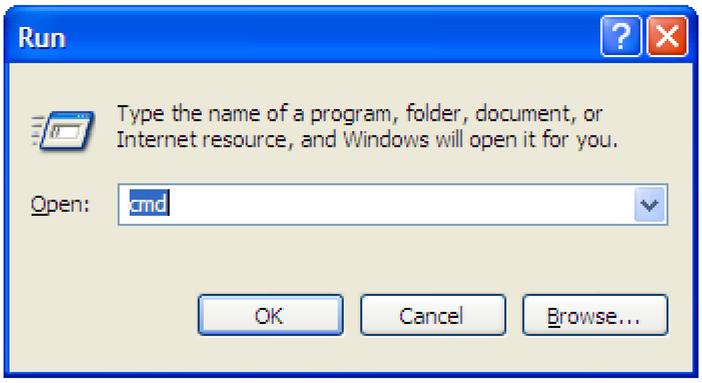
Frequently occurring tasks

This section describes tasks that are frequently used or referred to in this guide.

Opening a Windows command prompt

You may need to run and edit scripts in a command prompt when installing on a Windows-based system.

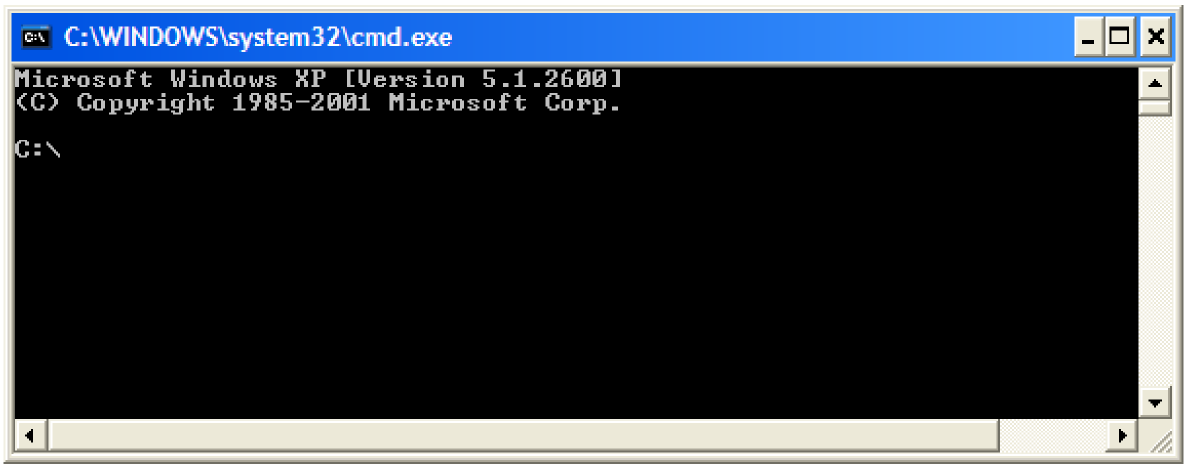
1. On the Windows task bar, click **Start > Run**.
2. In the **Run** dialog box, type `cmd`.



Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
cmd
OK
Cancel
Browse...

3. Click **OK**.

   The **Run** dialog box closes and a command prompt opens.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\
```

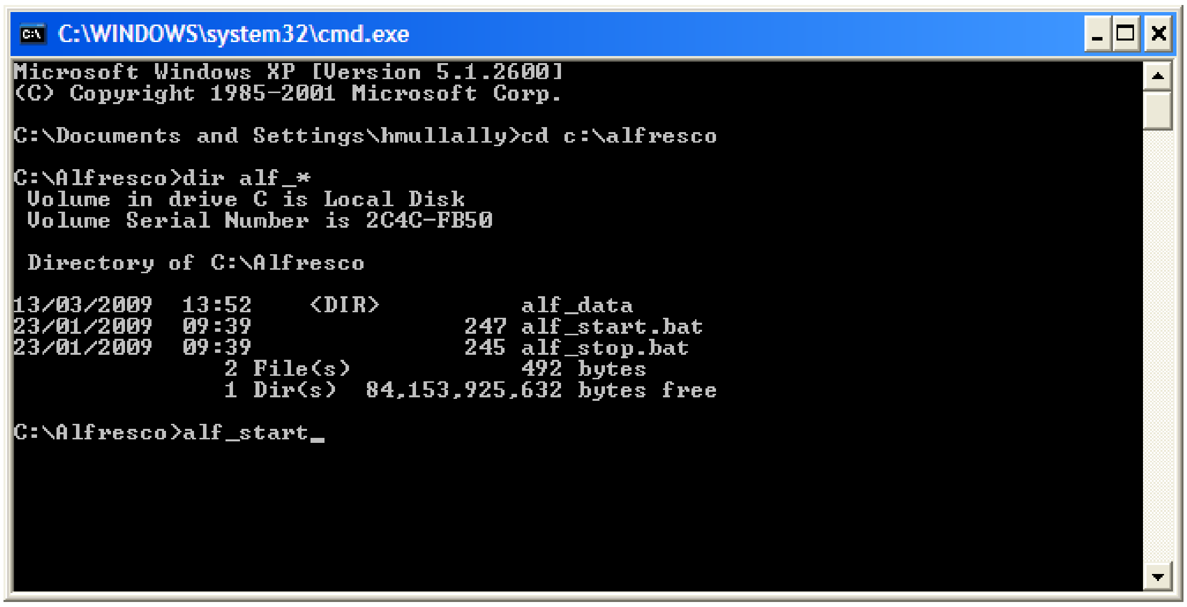
Running Windows batch files

When you have installed Alfresco on a Windows-based system, you may prefer to run Alfresco from a batch file. A batch file is a Windows file with a `.bat` extension.

1. In Windows Explorer, browse to `C:\Alfresco`.
2. Double-click a file name with a `.bat` extension.

   For example, to start Alfresco, double-click the file name `alf_start.bat`.
3. Alternatively, in a command prompt, type `cd c:\alfresco`, and press ENTER.

4. To check that you are in the correct directory, type `dir alf_*`, and look for `alf_start.bat`.
5. Type `alf_start`.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\hmullally>cd c:\alfresco

C:\Alfresco>dir alf_*
 Volume in drive C is Local Disk
 Volume Serial Number is 2C4C-FB50

 Directory of C:\Alfresco

13/03/2009  13:52    <DIR>          alf_data
23/01/2009  09:39               247 alf_start.bat
23/01/2009  09:39               245 alf_stop.bat
               2 File(s)            492 bytes
               1 Dir(s)  84,153,925,632 bytes free

C:\Alfresco>alf_start_
```

The command prompt is closed on normal completion, or if the program is terminated by a command error. If the command prompt closes before you have time to see the error that caused the program to terminate, you can run the batch program by opening a command prompt yourself.

Adding folder paths to the Windows path variable

You may need to add folder paths to the Windows `path` variable when installing on a Windows-based system.

1. On the Windows desktop, right-click **My Computer**.
2. In the pop-up menu, click **Properties**.
3. In the **System Properties** window, click the **Advanced** tab, and then click **Environment Variables**.
4. In the **System Variables** window, highlight **Path**, and click **Edit**.
5. In the **Edit System Variables** window, insert the cursor at the end of the **Variable** value field.
6. If the last character is not a semi-colon (;), add one.
7. After the final semi-colon, type the full path to the file you want to find.

   For example: `path C:\jdk`
8. Click **OK** in each open window.

   The new path will be used the next time a command prompt is opened, or a service is started.

Changing the default shell (Unix/Linux/Solaris) for shell scripts

When you run Alfresco on the Unix, Linux, or Solaris operating systems, the default shell is `sh`. You can edit the `alfresco.sh` file to change to your preferred shell.

1. Open the `alfresco.sh` file.

These steps also apply to any shell script, for example: `apply_amps.sh` or `deploy_start.sh`.

2. Edit the shell command to specify your preferred shell.

   For example, change the `#!/bin/sh` line to `#!/bin/bash`.
3. Save the `alfresco.sh` file.

Setting file limits for Linux

When running Alfresco on Red Hat Linux, if you encounter a "Too many open files" error message, you must increase the file limits setting.

These steps assumes that Alfresco is running as the root `alfresco` user.

1. Edit the following file:

   `/etc/security/limits.conf`
2. Add the following settings:

   ```
   alfresco soft nofile 4096
   alfresco hard nofile 65536
   ```

   This sets the normal number of file handles available to the `alfresco` user to be 4096. This is known as the soft limit.
3. As the `alfresco` user, set a system-level setting for Linux, up to the hard limit, using the following command:

   ```
   ulimit -n 8192
   ```

Properties that can be edited in a JMX client

This section contains a summary of the properties that can be viewed and changed in a JMX client.

`alfresco.authentication.allowGuestLogin`
Specifies whether to allow guest access to Alfresco.

`alfresco.authentication.authenticateCIFS`
A Boolean that when true enables Alfresco-internal authentication for the CIFS server. When false and no other members of the authentication chain support CIFS authentication, the CIFS server will be disabled.

`ntlm.authentication.mapUnknownUserToGuest`
Specifies whether unknown users are automatically logged on as the Alfresco guest user during Single Sign-On (SSO).

`ntlm.authentication.sso.enabled`
A Boolean that when true enables NTLM based Single Sign On (SSO) functionality in the Web clients. When false and no other members of the authentication chain support SSO, password-based login will be used.

`authentication.chain`
Specifies the authentication chain.

`synchronization.autoCreatePeopleOnLogin`
Specifies whether to create a user with default properties when a user is successfully authenticated, who does not yet exist in Alfresco, and was not returned by a differential sync (if enabled with the property above). The default is true. Setting this to false allows you to restrict Alfresco to a subset of those users who could be authenticated by LDAP; only those created by synchronization are allowed to log in. You can control the set of users in this more restricted set by overriding the user query properties of the LDAP authentication subsystem

`synchronization.import.cron`
Specifies a cron expression defining when the scheduled synchronization job should run, by default at midnight every day.

`synchronization.loggingInterval`
Specifies the number of user or group entries the synchronization subsystem will process before logging progress at INFO level. If you have the following default entry in log4j.properties:

`log4j.logger.org.alfresco.repo.security.sync=info`. The default is 100.

`synchronization.syncOnStartup`
Specifies whether to trigger a differential sync when the subsystem starts up. The default is true. This ensures that when user registries are first configured, the bulk of the synchronization work is done on server startup, rather than on the first login.

`synchronization.syncWhenMissingPeopleLogIn`
Specifies whether to trigger a differential sync when a user is successfully authenticated who does not yet exist in Alfresco. The default is true.

`synchronization.synchronizeChangesOnly`
Specifies if the scheduled synchronization job is run in differential mode. The default is false, which means that the scheduled sync job is run in full mode. Regardless of this setting a differential sync may still be triggered when a user is successfully authenticated who does not yet exist in Alfresco.

`synchronization.workerThreads`
Specifies the number of worker threads. For example, 2.

`cifs.WINS.autoDetectEnabled`
When true causes the cifs.WINS.primary and cifs.WINS.secondary properties to be ignored.

`cifs.WINS.primary`
Specifies a primary WINS server with which to register the server name.

`cifs.WINS.secondary`
Specifies a secondary WINS server with which to register the server name.

`cifs.bindto`
Specifies the network adapter to which to bind. If not specified, the server will bind to all available adapters/addresses.

`cifs.disableNIO`
Disables the new NIO-based CIFS server code and reverts to using the older socket based code.

`cifs.disableNativeCode`
When true, switches off the use of any JNI calls and JNI-based CIFS implementations.

`cifs.domain`
An optional property. When not empty, specifies the domain or workgroup to which the server belongs. This defaults to the domain/workgroup of the server, if not specified.

`cifs.enabled`
Enables or disables the CIFS server.

`cifs.hostannounce`
Enables announcement of the CIFS server to the local domain/workgroup so that it shows up in Network Places/Network Neighborhood.

`cifs.ipv6.enabled`
Enables the use of IP v6 in addition to IP v4 for native SMB. When true, the server will listen for incoming connections on IPv6 and IPv4 sockets.

`cifs.netBIOSSMB.datagramPort`
Controls the NetBIOS datagram port. The default is 138.

`cifs.netBIOSSMB.namePort`
Controls the NetBIOS name server port on which to listen. The default is 137.

`cifs.netBIOSSMB.sessionPort`
Controls the NetBIOS session port on which to listen for incoming session requests. The default is 139.

`cifs.serverName`
Specifies the host name for the Alfresco CIFS server. This can be a maximum of 16 characters and must be unique on the network. The special token {localname} can be used in place of the local server's host name and a unique name can be generated by prepending/appending to it.

`cifs.sessionTimeout`
Specifies the CIFS session timeout value in seconds. The default session timeout is 15 minutes. If no I/O occurs on the session within this time then the session will be closed by the server. Windows clients send keep-alive requests, usually within 15 minutes.

`cifs.tcpipSMB.port`
Controls the port used to listen for the SMB over TCP/IP protocol (or native SMB), supported by Win2000 and above clients. The default port is 445.

`cifs.urlfile.prefix`
An absolute URL against which all desktop actions and URL files resolve their folder URL. The special token {localname} can be used in place of the local server's host name.

`filesystem.acl.global.defaultAccessLevel`
Specifies the default access level. Directly names the access control level (None, Read or Write) that applies to requests that are not in scope of any other access control. Note that it is not valid to use the value None without defining other access controls.

`filesystem.acl.global.domainAccessControls`
Specifies the set of access controls with domain scope. This is a composite property whose value should be a comma-separated list of domain names. To define the access level for one of the listed domains, use the property filesystem.acl.global.domainAccessControls.value.Domain.accessType.

`filesystem.acl.global.protocolAccessControls`
Specifies the set of access controls with protocol scope. This is a composite property whose value should be a comma-separated list of access control names.

`filesystem.acl.global.userAccessControls`
Specifies the set of access controls with user scope. This is a composite property whose value should be a comma-separated list of user names.

`filesystem.domainMappings`
Specifies the domain mapping rules that are used when the client does not supply its domain in the NTLM request.

`filesystem.name`
Specifies the name given to the repository file system mount exposed through the CIFS server. For example, Alfresco.

`ftp.enabled`
Enables or disables the FTP server.

`ftp.ipv6.enabled`
Enables or disables the IPv6 FTP server.

`ftp.port`
Specifies the port that the FTP server listens for incoming connections on. Defaults to port 21.

`nfs.enabled`
Enables or disables the NFS server.

`nfs.user.mappings`
A composite property that configures the user ID/group ID to the Alfresco user name mappings that are used by the current RPC authentication implementation.

`nfs.user.mappings.default.gid`
The Group Identifier (GID) for NFS user mappings.

`nfs.user.mappings.default.uid`
The User Identifier (UID) for NFS user mappings.

`imap.config.home.folderPath`
Specifies the default locations for the IMAP mount point. For example, `Imap Home`.

`imap.config.home.rootPath`
Specifies the default location for the IMAP mount point. For example, `/${spaces.company_home.childname}`.

`imap.config.home.store`
Specifies the default location for the IMAP mount point. For example, `${spaces.store}`.

`imap.config.ignore.extraction`
Defines whether or not attachments are extracted.

`imap.config.server.mountPoints`
Defines whether or not attachments are extracted.

`imap.config.server.mountPoints`
Defines whether or not attachments are extracted.

`imap.config.server.mountPoints`
Defines whether or not attachments are extracted.

`imap.config.server.mountPoints`
Specifies the list of mount points. For example, `AlfrescoIMAP`.

`imap.server.enabled`
Enables or disables the IMAP server. This is set to false, by default.

`imap.server.host`
Specifies the host for the IMAP server.

`imap.server.port`
Specifies the port number for the IMAP server. For example, 143.

`imap.config.server.mountPoints.value.AlfrescoIMAP.modeName`
Specifies the `AlfrescoIMAP` mount point access mode name. For example, `MIXED`.

`imap.config.server.mountPoints.default.rootPath`
Specifies the root path for the mount point.

`imap.config.server.mountPoints.value.AlfrescoIMAP.mountPointName`
Specifies the mount point name.

`imap.config.server.mountPoints.default.store`
Specifies the default store for the mount point.

`server.allowedusers`
A comma-separated list of users who are allowed to log in. Leave empty if all users are allowed to log in.

`server.maxusers`
The maximum number of users who are allowed to log in or -1 if there is no limit.

`server.transaction.allow-writes`
A Boolean property that when true indicates that the repository will allow write operations (provided that the license is valid). When false the repository is in read-only mode.

`img.dyn`
Points to the directory containing the ImageMagick shared library (Unix) or DLL files (Windows). For example, (Windows) `img.dyn=${img.root}`; (Linux) `img.dyn=${img.root}/lib`.

`img.exe`
Points to the ImageMagick executable file name.

`img.root`
Points to the ImageMagick root directory.

`swf.exe`
Points to the SWF Tools executable file name.

`wcm-deployment-receiver.poll.delay`
Specifies how long to wait before polling. For example, 5000.

`wcm-deployment-receiver.rmi.service.port`
Specifies the port number for the RMI service. For example, 44101

JMX bean categories reference

This reference section provides detailed information on the individual bean types exported by Alfresco.

The heading for each bean type provides the JMX object naming scheme, where possible. Each section lists the individual properties for the bean type.

JMX read-only monitoring beans

This section contains the list of read-only monitoring beans.

Alfresco:Name=Authority

Exposes key metrics relating to the authority service:

NumberOfGroups
The number of groups known to the Authority Service.

NumberOfUsers
The number of users known to the Authority Service.

Alfresco:Name=ConnectionPool

Allows monitoring of the Apache Commons DBCP database connection pool and its configuration. It exposes the following properties:

DefaultTransactionIsolation
The JDBC code number for the transaction isolation level, corresponding to those in the `java.sql.Connection` class. The special value of -1 indicates that the database's default transaction isolation level is in use and this is the most common setting. For the Microsoft SQL Server JDBC driver, the special value of 4096 indicates snapshot isolation.

DriverClassName
The fully-qualified name of the JDBC driver class.

InitialSize
The number of connections opened when the pool is initialized.

MaxActive
The maximum number of connections in the pool.

MaxIdle
The maximum number of connections that are not in use kept open.

MaxWait
The maximum number of milliseconds to wait for a connection to be returned before throwing an exception (when connections are unavailable) or -1 to wait indefinitely.

MinEvictableIdleTimeMillis
The minimum number of milliseconds that a connection may sit idle before it is eligible for eviction.

MinIdle
The minimum number of connections in the pool.

NumActive
The number connections in use; a useful monitoring metric.

NumIdle
The number of connections that are not in use; another useful monitoring metric.

Url
The JDBC URL to the database connection.

Username
The name used to authenticate with the database.

RemoveAbandoned
A Boolean that when true indicates that a connection is considered abandoned and eligible for removal if it has been idle longer than the `RemoveAbandonedTimeout`.

RemoveAbandonedTimeout
The time in seconds before an abandoned connection can be removed.

TestOnBorrow
A boolean that when true indicates that connections will be validated before being borrowed from the pool.

TestOnReturn
A boolean that when true indicates that connections will be validated before being returned to the pool.

TestWhileIdle
A boolean that when true indicates that connections will be validated whilst they are idle.

TimeBetweenEvictionRunsMillis
The number of milliseconds to sleep between eviction runs, when greater than zero.

ValidationQuery
The SQL query that will be used to validate connections before returning them.

Alfresco:Name=ContentStore,Type=*,Root=*

Allows monitoring of each of Alfresco content stores. When `Type=FileContentStore`, the Root attribute of the name holds the file system path to the store. The following properties are exposed:

TotalSize
The total size in bytes.

WriteSupported
Stated whether the store currently allow write operations.

Alfresco:Name=ContentTransformer,Type=*

Exposes key information about the transformation utilities relied upon by Alfresco. Currently, there are two instances:

- `Alfresco:Name=ContentTransformer,Type=ImageMagick`
- `Alfresco:Name=ContentTransformer,Type=pdf2swf`

The following properties are exposed:

Available
A boolean that when true indicates that the utility is actually installed correctly and was found when the Alfresco server started up.

VersionString
The version information returned by the utility, if it was found to be available.

Alfresco:Name=DatabaseInformation

Exposes metadata about the database itself.

DatabaseMajorVersion
The database version number.

DatabaseMinorVersion
The database version number.

DatabaseProductName
The database product name.

DatabaseProductVersion
The database product version.

DriverMajorVersion
The driver major version number.

DriverMinorVersion
The driver minor version number.

DriverName
Product name of the JDBC driver.

DriverVersion
The driver version number.

JDBCMajorVersion
The major version number of the JDBC specification supported by the driver.

JDBCMinorVersion
The minor version number of the JDBC specification supported by the driver.

StoresLowerCaseIdentifiers

StoresLowerCaseQuotedIdentifiers

StoresMixedCaseIdentifiers

StoresMixedCaseQuotedIdentifiers

StoresUpperCaseIdentifiers

StoresUpperCaseQuotedIdentifiers

URL
The JDBC URL of the database connection.

UserName
The name used to authenticate with the database.

Alfresco:Name=Hibernate

An instance of the `StatisticsService` class provided by Hibernate, allowing access to an extensive set of Hibernate-related metrics.

Alfresco:Name=LicenseDescriptor

Exposes the parameters of the Alfresco Enterprise license.

Days
The number of days of usage that the license allows from its issue date, if the license is time limited.

HeartBeatDisabled
A boolean that when true indicates that the license permits the usage of the Alfresco server with its heartbeat functionality disabled (involving the automatic submission of basic repository statistics to Alfresco).

Holder
The person or entity to which the license was issued.

Issued
The date and time on which the license was issued.

Issuer
Who issued the license (always Alfresco).

RemainingDays
The number of days of usage that the license allows from today, if the license is time limited.

Subject
The product edition to which the license applies.

ValidUntil
The date on which the license will expire, if the license is time limited.

Alfresco:Name=LuceneIndexes,Index=*

Allows monitoring of each searchable index. The Index attribute of the name holds the relative path to the index under `alf_data/lucene-indexes` and the following properties are exposed:

ActualSize
The size of the index in bytes.

EntryStatus
A composite table containing the current status of each entry in the index (double-click the value in JConsole to expand it and view its rows). Each row in the table has a key of the format `<ENTRY TYPE>-<ENTRY STATE>`, for example, `DELTA-COMMITTED` and a value containing the number of entries with that type and state.

EventCounts
A composite table containing the names and counts of significant events that have occurred on the index since the server was started (double-click the value in JConsole to expand it and view its rows). Examples of event names are `CommittedTransactions`, `MergedDeletions` and `MergedIndexes`.

NumberOfDocuments
The number of documents in the index.

NumberOfFields
The number of fields known to the index.

NumberOfIndexedFields
The number of these fields that are indexed.

UsedSize
The size of the index directory in bytes. A large discrepancy from the value of `ActualSize` may indicate that there are unused data files.

Alfresco:Name=ModuleService

Allows monitoring of installed modules.

AllModules
A composite table containing the details of all modules currently installed. Double-click the value in JConsole to expand it and use the **Composite Navigation** arrows to navigate through each module.

Alfresco:Name=OpenOffice

Exposes information about the OpenOffice server used for document conversions. In addition to the property below, this bean has a property corresponding to each registry key in the `org.openoffice.Setup` sub-tree of the OpenOffice configuration registry, providing useful metadata about the particular flavor of OpenOffice that is installed. For example, `ooName` provides the product name, for example, `"OpenOffice.org"` and `ooSetupVersionAboutBox` provides its version, for example, "3.0.0".

available
A Boolean that when true indicates that a connection was successfully established to the OpenOffice server.

Alfresco:Name=PatchService

Allows monitoring of installed patches.

AppliedPatches
A composite table containing the details of all patches currently installed. Double-click the value in JConsole to expand it and use the "Composite Navigation" arrows to navigate through each patch.

Alfresco:Name=RepositoryDescriptor,Type=*

Exposes metadata about the Alfresco repository. Currently, there are two instances of this bean:

Alfresco:Name=RepositoryDescriptor,Type=Installed
Exposes information about the initial repository installation, before any patches or upgrades were installed. Of most relevance to patch and upgrade scenarios.

Alfresco:Name=RepositoryDescriptor,Type=Server
Exposes information about the current server version, as contained in the Alfresco war file. This instance should be used to determine the current properties of the server.

Both expose the following properties:

Edition
The Alfresco edition, for example, "Enterprise".

Id
The repository unique ID. This property is only available from the Installed descriptor.

Name
The repository name.

Schema
The schema version number.

Version
The full version string, including build number, for example, "3.1.0 (stable r1234)".

VersionBuild
The build number.

VersionLabel
An optional label given to the build, such as "dev" or "stable".

VersionMajor
The first component of the version number.

VersionMinor
The second component of the version number.

VersionNumber
The full version number, composed from major, minor and revision numbers.

VersionRevision
The third component of the version number.

Alfresco:Name=Runtime

Exposes basic properties about the memory available to the JVM. Note that a Sun JVM exposes much more detailed information through its platform MX Beans.

FreeMemory
The amount of free memory in bytes.

MaxMemory
The maximum amount of memory that the JVM will attempt to use in bytes.

TotalMemory
The total amount of memory in use in bytes.

Alfresco:Name=Schedule,Group=*,Type=*,Trigger=*

Allows monitoring of the individual triggers, i.e. scheduled jobs, running in the Quartz scheduler. The attributes of the object name have the following meaning:

Group
The name of the schedule group that owns the trigger. Typically DEFAULT.

Type
The type of trigger, typically MonitoredCronTrigger or MonitoredSimpleTrigger. Triggers of different types have different properties, as you will see below.

Trigger
The name of the trigger itself. Must be unique within the group.

All instances have the following properties:

CalendarName
The name of the scheduling Calendar associated with the trigger, or null if there is not one.

Description
An optional textual description of the trigger.

EndTime
The time after which the trigger will stop repeating, if set.

FinalFireTime
The time at which the last execution of the trigger is scheduled, if applicable.

Group
The name of the schedule group that owns the trigger.

JobGroup
The name of the schedule group that owns the job executed by the trigger.

JobName
The name of the job executed by the trigger.

MayFireAgain
A Boolean that when true indicates that it is possible for the trigger to fire again.

Name
The name of the trigger.

NextFireTime
The next time at which the trigger will fire.

PreviousFireTime
The previous time at which the trigger fired.

Priority
A numeric priority that decides which trigger is executed before another in the event of a 'tie' in their scheduled times.

StartTime
The time at which the trigger should start.

State
The current state of the trigger.

Volatile
A Boolean that when true indicates that the trigger will not be remembered when the JVM is restarted.

When Type=MonitoredCronTrigger, the following additional properties are available:

CronExpression
A unix-like expression, using the same syntax as the cron command, that expresses when the job should be scheduled.

TimeZone
The name of the time zone to be used to interpret times.

When Type=MonitoredSimpleTrigger the following additional properties are available:

RepeatCount
The number of times the job should repeat, after which it will be removed from the schedule. A value of -1 means repeat indefinitely.

RepeatInterval
The time interval in milliseconds between job executions.

TimesTriggered
The number of times the job has been run.

Alfresco:Name=SystemProperties

A dynamic MBean exposing all the system properties of the JVM. The set of standard system properties is documented on the Apache website.

JMX configuration beans

This section contains the list of configuration beans. Alfresco introduces an innovative way to manage the configuration of the individual Spring beans that compose the server. This feature is available for security and authentication configuration, which can be particularly complex to manage given the possibility of multiple-chained authentication services and authentication components, each with their own DAOs and other supporting services.

To help with the management of such configuration, the key properties of key authentication bean classes are annotated with a special `@Managed` annotation, that causes them to be exposed automatically through dynamic MBeans under the `Alfresco:Type=Configuration` naming tree. This means that the key beans that make up your authentication chain will become visible to a JMX client, no matter how they are named and wired together.

The current set of authentication classes that have this facility include:

- Authentication Components, including chained, JAAS, LDAP and NTLM components
- Authentication Services, including chained and unchained
- Authentication DAOs
- `LDAPInitialDirContextFactories`, encapsulating the parameters of the LDAP server
- `LDAPPersonExportSource`, controlling the synchronization of person information with an LDAP server

In JConsole, the view of a server with a particularly complex authentication configuration that shows all the authentication classes are visible under the Alfresco`:Type=Configuration` naming tree and navigable with JConsole. These beans provide a read-only view of the configuration.

JMX editable management beans

This section contains the list of editable management beans.

Alfresco:Name=FileServerConfig

Allows management and monitoring of the various file servers.

Read-only properties:

CIFSServerAddress
Not implemented.

CIFSServerName
The CIFS server name, if available.

Editable Properties:

These are not cluster-aware. If more than one file server is running (for example, load-balanced FTP) then changes will need to be applied to each machine. Some consoles (for example, JManage) may provide basic facilities for accessing each machine in an application cluster.

CIFSServerEnabled
A Boolean that when true indicates that the CIFS server is enabled and functioning.

FTPServerEnabled
A Boolean that when true indicates that the FTP server is enabled and functioning.

NFSServerEnabled
A Boolean that when true indicates that the NFS server is enabled and functioning.

Alfresco:Name=Log4jHierarchy

An instance of the HierarchyDynamicMBean class provided with log4j that allows adjustments to be made to the level of detail included in the Alfresco server's logs. Note that it is possible to run Alfresco using JDK logging instead of log4j, in which case this bean will not be available.

Read-only properties:

The bean has a property for each logger known to log4j, whose name is the logger name, usually corresponding to a Java class or package name, and whose value is the object name of another MBean that allows management of that logger (see `#log4j:logger=*`). Despite how it might seem, these properties are read-only and editing them has no effect.

Editable properties:

There is one special editable property and note again that it is not cluster aware.

threshold
Controls the server-wide logging threshold. Its value must be the name of one of the log4j logging levels. Any messages logged with a priority lower than this threshold will be filtered from the logs. The default value is ALL, which means no messages are filtered, and the highest level of filtering is OFF which turns off logging altogether (not recommended).

Operations with Impact:

addLoggerMBean
This adds an additional logger to the hierarchy, meaning that the bean will be given an additional read-only property for that logger and a new MBean will be registered in the `#log4j:logger=*` tree, allowing management of that logger. Is is not normally necessary to use this operation, because the Alfresco server pre-registers all loggers initialized during startup. However, there may be a chance that the logger you are interested in was not initialized at this point, in which case you will have to use this operation. The operation requires the fully qualified name of the logger as an argument and if successful returns the object name of the newly registered MBean for managing that logger.

For example, if in Java class `org.alfresco.repo.admin.patch.PatchExecuter` the logger is initialized as follows:

```
private static Log logger = LogFactory.getLog(PatchExecuter.class);
```

Then the logger name would be `org.alfresco.repo.admin.patch.PatchExecuter`.

## log4j:logger=*

An instance of the LoggerDynamicMBean class provided with log4j that allows adjustments to be made to the level of detail included in the logs from an individual logger. Note that it is possible to run Alfresco using JDK logging instead of log4j, in which case this bean will not be available.

Read-only properties:

name
The logger name

Editable properties:

There is one special editable property and note again that it is not cluster aware.

priority

The name of the minimum log4j logging level of messages from this logger to include in the logs. For example, a value of ERROR would mean that messages logged at lower levels such as WARN and INFO would not be included.

Alfresco:Name=VirtServerRegistry,Type=VirtServerRegistry

This is used directly by the Alfresco Virtualization Server.

Search syntax

The following sections describe the Alfresco search syntax.

Search for a single term

Single terms are tokenized before the search according to the appropriate data dictionary definition(s).

If you do not specify a field, it will search in the content and properties. This is a shortcut for searching all properties of type content.

```
banana
TEXT:banana
```

Both of these queries will find any nodes with the word "banana" in any property of type `d:content`.

If the appropriate data dictionary definition(s) for the field supports both FTS and untokenized search, then FTS search will be used. FTS will include synonyms if the analyzer generates them. Terms cannot contain whitespace.

Search for a phrase

Phrases are enclosed in double quotes. Any embedded quotes may be escaped using `"\"`. If no field is specified then the default TEXT field will be used, as with searches for a single term.

The whole phrase will be tokenized before the search according to the appropriate data dictionary definition(s).

```
"big yellow banana"
```

Search for an exact term

To search for an exact term, prefix the term with "=". This ensures that the term will not be tokenized, therefore you can search for stop words.

If both FTS and ID base search are supported for a specified or implied property, then exact matching will be used where possible.

```
=running
```

Will match "running" but will not be tokenized. If you are using stemming it may not match anything.

For the `cm:name` filed, which is in the index as both tokenized and untokized, it will use the untokenized field. For example, `=part` will only match the exact term "part". If you use `=part*` it will match additional terms, like "partners". If there is no untokenized field in the index, it will fall back to use the tokenized field, and then, with stemming/plurals, it would match.

Search for term expansion

To force tokenization and term expansion, prefix the term with "~".

For a property with both ID and FTS indexes, where the ID index is the default, force the use of the FTS index.

```
~running
```

Search for conjunctions

Single terms, phrases, and so on can be combined using "AND" in upper, lower, or mixed case.

If not otherwise specified, by default search fragments will be ANDed together.

```
big yellow banana
big AND yellow AND banana
TEXT:big TEXT:yellow TEXT:banana
TEXT:big and TEXT:yellow and TEXT:banana
```

All these queries search for nodes that contain the terms "big", "yellow", and "banana" in any content.

Search for disjunctions

Single terms, phrases, and so on can be combined using "OR" in upper, lower, or mixed case.

```
big OR yellow OR banana
TEXT:big OR TEXT:yellow OR TEXT:banana
```

Search for negation

Single terms, phrases, and so on can be combined using "NOT" in upper, lower, or mixed case, or prefixed with "!" or "-".

```
yellow NOT banana
yellow !banana
yellow -banana
NOT yellow banana
-yellow banana
!yellow banana
```

Search for optional, mandatory, and excluded elements of a query

Sometimes AND and OR are not enough. If you want to find documents that must contain the term "car", score those with the term "red" higher, but do not match those just containing "red".

Operator	Description
"|"	The field, phrase, group is optional; a match increases the score.
"+"	The field, phrase, group is mandatory (Note: this differs from Google - see "=")
"-", "!"	The field, phrase, group must not match.

The following example finds documents that contain the term "car", score those with the term "red" higher, but does not match those just containing "red":

```
+car |red
```

At least one element of a query must match (or not match) for there to be any results.

All AND and OR constructs can be expressed with these operators.

Search for fields

Search specific fields rather than the default. Terms, phrases, etc. can all be preceded by a field. If not the default field TEXT is used.

```
field:term
field:"phrase"
=field:exact
~field:expand
```

Fields fall into three types: property fields, special fields, and fields for data types.

Property fields evaluate the search term against a particular property, special fields are described in the following table, and data type fields evaluate the search term against all properties of the given type.

Description	Type	Example
Fully qualified property	Property	{http://www.alfresco.org/model/content/1.0}name:apple
Fully qualified property	Property	@{http://www.alfresco.org/model/content/1.0}name:apple
CMIS style property	Property	cm_name:apple
Prefix style property	Property	cm:name:apple
Prefix style property	Property	@cm:name:apple
TEXT	Special	TEXT:apple
ID	Special	ID:"NodeRef"
ISROOT	Special	ISROOT:T
TX	Special	TX:"TX"
PARENT	Special	PARENT:"NodeRef"
PRIMARYPARENT	Special	PRIMARYPARENT:"NodeRef"
QNAME	Special	QNAME:"app:company_home"
CLASS	Special	CLASS:"qname"
EXACTCLASS	Special	EXACTCLASS:"qname"
TYPE	Special	TYPE:"qname"
EXACTTYPE	Special	EXACTTYPE:"qname"
ASPECT	Special	ASPECT:"qname"
EXACTASPECT	Special	EXACTASPECT:"qname"
ALL	Special	ALL:"text"
ISUNSET	Special	ISUNSET:"property-qname"
ISNULL	Special	ISNULL:"property-qname"
ISNOTNULL	Special	ISNOTNULL:"property-qname"
Fully qualified data type	Data Type	{http://www.alfresco.org/model/dictionary/1.0}content:apple

Description	Type	Example
prefixed data type	Data Type	d:content:apple

Search for wildcards

Wildcards are supported in terms, phrases, and exact phrases using "*" to match zero, one, or more characters and "?" to match a single character. The "*" wildcard character may appear on its own and implies Google-style. The "anywhere after" wildcard pattern can be combined with the "=" prefix for identifier based pattern matching.

The following will all find the term apple.

```
TEXT:app?e
TEXT:app*
TEXT:*pple
appl?
*ple
=*ple
"ap*le"
"***le"
"?????"
```

Search for ranges

Inclusive ranges can be specified in Google-style. There is an extended syntax for more complex ranges. Unbounded ranges can be defined using MIN and MAX for numeric and date types and "\u0000" and "\FFFF" for text (anything that is invalid).

Lucene	Google	Description	Example

`[#1 TO #2]`	`#1..#2`	The range #1 to #2 inclusive
		`#1 <= x <= #2`
			`0..5`
			`[0 TO 5]`

`<#1 TO #2]`		The range #1 to #2 including #2 but not #1. `#1 < x <= #2`	`<0 TO 5]`
`[#1 TO #2>`		The range #1 to #2 including #1 but not #2. `#1 <= x < #2`	`[0 TO 5>`
`<#1 TO #2>`		The range #1 to #2 exclusive. `#1 < x < #2`	`<0 TO 5>`

```
TEXT:apple..banana
my:int:[0 TO 10]
my:float:2.5..3.5
my:float:0..MAX
mt:text:[l TO "\uFFFF"]
```

Search for fuzzy matching

Fuzzy matching is not currently implemented. The default Lucene implementation is Levenshtein Distance, which is expensive to evaluate.

Postfix terms with "~float"

```
apple~0.8
```

Search for proximity

Google-style proximity is supported.

To specify proximity for fields, use grouping.

```
big * apple
TEXT:(big * apple)
big *(3) apple
TEXT:(big *(3) apple)
```

Search for boosts

Query time boosts allow matches on certain parts of the query to influence the score more than others.

All query elements can be boosted: terms, phrases, exact terms, expanded terms, proximity (only in filed groups), ranges, and groups.

```
term^2.4
"phrase"^3
term~0.8^4
=term^3
~term^4
cm:name:(big * yellow)^4
1..2^2
[1 TO 2]^2
yellow AND (car OR bus)^3
```

Search for grouping

Groupings of terms are made using "(" and ")". Groupings of all query elements are supported in general. Groupings are also supported after a field - field group.

The query elements in field groups all apply to the same field and cannot include a field.

```
(big OR large) AND banana
title:((big OR large) AND banana)
```

Search for spans and positions

Spans and positions are not currently implemented. Positions will depend on tokenization.

Anything more detailed than one *(2) two are arbitrarily dependent on the tokenization. An identifier and pattern matching, or dual FTS and ID tokenization, may well be the answer in these cases.

```
term[^] - start
term[$] - end
term[position]
```

These are of possible use but excluded for now. Lucene surround extensions:

```
and(terms etc)
99w(terms etc)
97n(terms etc)
```

Escaping characters

Any character may be escaped using the backslash "\" in terms, IDs (field identifiers), and phrases. Java unicode escape sequences are supported. Whitespace can be escaped in terms and IDs.

For example:

```
cm:my\ content:my\ name
```

Mixed FTS ID behavior

This relates to the priority defined on properties in the data dictionary, which can be both tokenized or untokenized.

Explicit priority is set by prefixing the query with "=" for identifier pattern matches.

The tilde "~" can be used to force tokenization.

Search for order precedence

Operator precedence is SQL-like (not Java-like). When there is more than one logical operator in a statement, and they are not explicitly grouped using parentheses, `NOT` is evaluated first, then `AND`, and finally `OR`.

The following shows the operator precedence from highest to lowest:

```
"
[, ], <, >
()
~ (prefix and postfix), =
^
+, |, -
NOT,
AND
OR
```

`AND` and `OR` can be combined with `+`, `|`, `-` with the following meanings:

AND (no prefix is the same as +)	Explanation
`big AND dog`	big and dog must occur
`+big AND +dog`	big and dog must occur
`big AND +dog`	big and dog must occur
`+big AND dog`	big and dog must occur
`big AND |dog`	big must occur and dog should occur
`|big AND dog`	big should occur and dog must occur
`|big AND |dog`	both big and dog should occur, and at least one must match
`big AND -dog`	big must occur and dog must not occur
`-big AND dog`	big must not occur and dog must occur
`-big AND -dog`	both big and dog must not occur
`|big AND -dog`	big should occur and dog must not occur

OR (no prefix is the same as +)	Explanation
`dog OR wolf`	dog and wolf should occur, and at least one must match
`+dog OR +wolf`	dog and wolf should occur, and at least one must match
`dog OR +wolf`	dog and wolf should occur, and at least one must match
`+dog OR wolf`	dog and wolf should occur, and at least one must match

OR (no prefix is the same as +)	Explanation
`dog OR |wolf`	dog and wolf should occur, and at least one must match
`|dog OR wolf`	dog and wolf should occur, and at least one must match
`|dog OR |wolf`	dog and wolf should occur, and at least one must match
`dog OR -wolf`	dog should occur and wolf should not occur, one of the clauses must be valid for any result
`-dog OR wolf`	dog should not occur and wolf should occur, one of the clauses must be valid for any result
`-dog OR -wolf`	dog and wolf should not occur, one of the clauses must be valid for any result

Forms reference

This reference contains detailed information for forms controls and the configuration syntax.

Form controls

Controls are represented by a Freemarker template snippet, and each field has a control and an optional set of parameters.

The following controls are available.

association.ftl

The `association` control is used to allow objects in the repository to be picked and ultimately associated with the node being edited. The control uses the JavaScript `Alfresco.ObjectPicker` component to allow the user to browse the repository and pick objects.
The following parameters are available:

- `compactMode`: Determines whether the picker will be shown in compact mode
- `showTargetLink`: Determines whether a link to the document details page will be rendered to content items

category.ftl

The `category` control is used to allow the user to select categories for the node being edited. The control uses the JavaScript `Alfresco.ObjectPicker` component to allow the user to browse the category hierarchy.
The following parameters are available:

- `compactMode`: Determines whether the picker will be shown in compact mode

checkbox.ftl

The `checkbox` control renders a standard HTML check box control.
The following parameters are available:

- `styleClass`: Allows a custom CSS class to be applied to the check box

date.ftl

The `date` control renders a date field allowing free form entry of dates, as well as a calendar widget allowing dates to be selected visually. If appropriate a time field is also rendered.
The following parameters are available:

- `showTime`: Determines whether the time entry field should be displayed

encoding.ftl

The `encoding` control renders a selectable list of encodings.
The following parameters are available:

- `property`: The name of a content property to retrieve the current encoding from; if omitted the `field.value` value is used
- `styleClass`: Allows a custom CSS class to be applied to the select list

invisible.ftl

The `invisible` control renders nothing at all; it can be used when a form definition needs to be requested and returned but not displayed. This control has no parameters.

mimetype.ftl

The `mimetype` control renders a selectable list of mime types.
The following parameters are available:

- `property`: The name of a content property to retrieve the current mime type from, if omitted the field.value value is used
- `styleClass`: Allows a custom CSS class to be applied to the select list

period.ftl

The `period` control renders a selectable list of periods and an expression entry field.
The following parameters are available:

- `dataTypeParameters`: A JSON object representing the period definitions to show in the list

selectone.ftl

The `selectone` control renders a standard HTML select list.
The following parameters are available:

- `options`: A comma separated list of options to display, for example `"First,Second,Third"`. If a value for an option also needs to be specified, use the `"First|1,Second|2,Third|3"` format.
- `size`: The size of the list, that is, how many options are always visible
- `styleClass`: Allows a custom CSS class to be applied to the select list

size.ftl

The `size` control renders a read only human readable representation of the content size.
The following parameters are available:

- `property`: The name of a content property to retrieve the current content size from; if omitted the `field.value` value is used

textarea.ftl

The `textarea` control renders a standard HTML text area field.
The following parameters are available:

- `rows`: The number of rows the text area will have
- `columns`: The number of columns the text area will have
- `styleClass`: Allows a custom CSS class to be applied to the text area

textfield.ftl

The `textfield` control renders a standard HTML text field.
The following parameters are available:

- `styleClass`: Allows a custom CSS class to be applied to the text field
- `maxLength`: Defines the maximum number of characters the user can enter
- `size`: Defines the size of of the text field

Forms configuration syntax

The `share-config-custom.xml` file uses an XML configuration syntax.

The XML syntax is described as follows:

default-controls
The type element defines what control to use, by default, for each type defined in the Alfresco content model. The name attribute contains the prefix form of the data type, for example `d:text`. The template attribute specifies the path to the template snippet to use to represent the field. If the path value should be a relative path, it is relative from the `alfresco` package. If the path value is absolute, it is looked up relative to the `alfresco/web-extension/site-webscripts` package, normally found in the application server shared classes location. The `control-param` element provides a mechanism to pass parameters to control templates, meaning that control templates can be re-used.

constraint-handlers
The constraint element defines what JavaScript function to use to check that fields with constraints are valid before being submitted. The `id` attribute is the unique identifier given to the model constraint in the Alfresco content model, for example `LIST`. The `validation-handler` attribute represents the name of a JavaScript function that gets called when the field value needs to be validated. The `event` attribute defines what event will cause the validation handler to get called. This will be a standard DOM event, that is, `keyup`, `blur`, and so on. The validation handler called usually has a default message to display when validation fails, the `message` and `message-id` attributes provide a way to override this message. However, the validation messages are not shown (the **Submit** button is enabled/disabled).

dependencies
The `dependencies` element defines the list of JavaScript and CSS files required by any custom controls being used in the application. In order for valid XHTML code to be generated, the dependencies need to be known ahead of time so the relevant links can be generated in the HTML head section. The `src` attribute of both the JavaScript and CSS elements contains the path to the resource, the path should be an absolute path from the root of the web application (but not including the web application context).

form
The `form` element represents a form to display. If the form element exists within a config element that provides an evaluator and condition, the form will only be found if the item being requested matches the condition. If the form element exists within a config element without an evaluator and condition, the form is always found. The optional `id` attribute allows an identifier to be associated with the form, thus allowing multiple forms to be defined for the same item. The `submission-url` allows the action attribute of the generated form to be overridden so that the contents of the form can be submitted to any arbitrary URL.

view-form
The `view-form` element allows the default template that auto generates the form UI to be overridden. The `template` attribute specifies the path to the template to be used when the form is in view mode. The value is usually an absolute path, which is relative to the `alfresco/web-extension/site-webscripts` package, normally found in the application server shared classes location. If this element is present, the `field-visibility` element is effectively ignored and therefore does not have to be present.

edit-form
The `edit-form` element allows the default template that auto generates the form UI to be overridden. The `template` attribute specifies the path to the template to be used when the form is in edit mode. The value is usually an absolute path, which is relative to the `alfresco/web-extension/site-webscripts` package, normally found in the application server shared classes location. If this element is present, the `field-visibility` element is effectively ignored and therefore does not have to be present.

create-form

The `create-form` element allows the default template that auto generates the form UI to be overridden. The `template` attribute specifies the path to the template to be used when the form is in create mode. The value is usually an absolute path, which is relative to the `alfresco/web-extension/site-webscripts` package, normally found in the application server shared classes location. If this element is present, the `field-visibility` element is effectively ignored and therefore does not have to be present.

field-visibility

The `field-visibility` element defines which fields are going to appear on the form, unless a custom template is used.

show

The `show` element specifies a field that should appear on the form. The `id` attribute represents the unique identifier for a field, for example, `cm:name`. The optional `for-mode` attribute indicates when the field should appear. Valid values for the attribute are `view`, `edit`, and `create`. If the attribute is not specified, the field will appear in all modes. If present, the field will only appear for the modes listed. For example, to only show a field in view and edit modes, the `for-mode` attribute would contain `view,edit`.
There are fields that may be optional for an item, and by default they may not be returned by the server. The `force` attribute can be used to indicate to the form service that it should do everything it can to find and return a definition for the field. An example might be a property defined on an aspect, if the aspect is not applied to the node, a field definition for the property will not be returned If force is `true`, it would indicate that server needs to try and find the property on an aspect in the content model.

hide

The `hide` element normally comes into play when multiple configuration files are combined as it can be used to hide fields previously configured to be shown. The `id` attribute represents the unique identifier for a field, for example `cm:name` that should not be displayed. The optional `for-mode` attribute indicates in which modes the field should not appear. Valid values for the attribute are view, edit, and `create`. If the attribute is not specified, the field will never appear. If present, the field will be hidden for the modes listed. For example, to hide a field in view and edit modes, the `for-mode` attribute would contain `view,edit`.
The algorithm for determining whether a particular field will be shown or hidden works, as follows:

1. If there is no `field-visibility` configuration (show or hide tags) then all fields are visible in all modes.
2. If there are one or more hide tags then the specified field(s) will be hidden in the specified modes. All other fields remain visible as before.
3. As soon as a single `show` tag appears in the configuration XML, this is taken as a signal that all field visibility is to be manually configured. At that point, all fields default to hidden and only those explicitly configured to be shown (with a `show` tag) will be shown.
4. Show and hide rules will be applied in sequence, with later rules potentially invalidating previous rules.
5. Show or hide rules, which only apply for specified modes, have an implicit element. For example, `<show id="name" for-mode="view"/>` would show the name field in view mode and by implication, hide it in other modes.

appearance

The optional `appearance` element controls the look and feel of the controls that make up the form. Unlike the `field-visibility` element, this element will be processed and the information available to custom templates defined with the `view-form`, `edit-form` and `create-form` elements, it is up to those templates whether they use the available data.
The configuration of what fields are present and how they appear has been separated to provide the maximum flexibility, and although it maybe slightly more verbose, the separation allows the appearance to be defined for fields that are not explicitly mentioned within the `field-visibility` element.

set

The optional `set` element provides the basis of creating groups of fields. The `id` attribute gives the set a unique identifier that other set definitions and fields can refer to. The `parent` attribute allows sets to be nested, and the value should reference a valid set definition, previously defined. The `appearance` attribute specifies how the set will be rendered. Currently, the only supported and allowed values are `fieldset` and `panel`. If an `appearance` attribute is not supplied, the set will not be rendered. The `label` and `label-id` attributes provide the title for the set when it is rendered. If neither are supplied, the set identifier is used.
A default set with an identidier of `""` (empty string) is always present, and any fields without an explicit set membership automatically belong to the default set. The default set will be displayed with a label of `Default`.

field

The `field` element allows most aspects of a field's appearance to be controlled from the label to the control that should be used. The only mandatory attribute is `id`, which specifies the field to be customized. However, the field identifier does not have to be present within the `field-visibility` element.
The `label` and `label-id` attributes define the label to be used for the form. If neither attribute is present, the field label returned from the Form Service is used. The `description` and `description-id` attributes are used to display a tool tip for the field. If neither is present, the description returned from the Form Service is used (this could also be empty).
The `read-only` attribute indicates to the form UI generation template that the field should never be shown in an editable form. Finally, the optional `set` attribute contains the identifier of a previously defined set. If the attribute is omitted, the field belongs to the default set.

control

The `control` element allows the control being used for the field to be configured or customized. If present, the `template` attribute specifies the path to the template snippet to use to represent the field overriding the `default-control` template. If the path value is relative, it is relative from the `alfresco` package. If the path value is absolute, it is looked up relative to the `<web-extension>/site-webscripts` package, normally found in the application server shared classes location.
The `control-param` sub-elements provide a mechanism to pass parameters to control templates. This template could either be the one defined locally or the template defined in the `default-control` element for the data type of the field.

constraint-handlers

The `constraint` sub-elements define the JavaScript function to use for checking that fields with constraints are valid before being submitted. The main purpose of this element is to allow aspects of the constraint to be overridden for a particular field. Each attribute effectively overrides the equivalent attribute.

Tips for getting the most out of Alfresco

1. Allow sufficient time to plan your project and identify the most optimal path for you.

2. Benchmark the system you want to use to ensure you can tune it for best performance and high availability before you go live.
3. Ensure customizations occur using the `<extensions>` directory and/or `AMP` files to help smooth upgrade and debugging processes.
4. Discover more about FreeMarker templates. You can create custom views for your spaces, and email templates to fit your organization, among other things.
5. Discover more about web scripts. This requires some, but not extensive, technical knowledge, and is very powerful.
6. Use a space template to create reusable components and enable business processes.
7. Leverage the CIFS interface to easily integrate with existing applications using drag and drop.
8. For Microsoft shops, Microsoft Office integration makes adoption of Alfresco seamless.
9. Email integration provides simple and safe way to store emails inside the Alfresco repository.
10. Coordinate with Alfresco on short-term consulting. This allows you and/or your System Integrator to work with Alfresco on architecture and planning.
11. Take advantage of the support for multiple security protocols, which makes it suitable for large enterprises.
12. Use Alfresco Network, a member-only (Enterprise subscription) benefit that provides a customer portal, further documentation, and a Knowledge Base.
13. Take advantage of Alfresco training. Get the knowledge and information you need to make your implementation successful.

Common mistakes made by Alfresco administrators

1. Not copying the Enterprise license. Administrators often forget to do this before the trial period expires, resulting in a system that goes into read-only mode.
2. Not keeping extended configurations and customizations separate in the shared directory. Do not put them in the configuration root. If you do, you will lose them during upgrades.
3. Not ensuring that the database driver (especially with MySQL post 2.1 release) is copied to the application server `lib` directory when installing.
4. Not testing the backup strategy.
5. Making changes to the system without testing them thoroughly on a test and pre-production machine first.
6. Failing to set the `dir.root` property to an absolute path location.
7. Not fully shutting down a running instance of Alfresco, so the next time you try and start it, Alfresco says: `Address already in use: JVM_Bind:8080` (especially on Linux).

Eight shortcuts every Alfresco administrator should know

1. Make sure you use a transactional database.
2. Keep your Lucene indexes on your fastest local disk.
3. Version only what and when you need to.
4. If you find yourself constantly creating the same space hierarchy as well as rules and aspects to them, consider creating a Space template instead.
5. Refer to Customizing Alfresco Explorer on page 137 and Customizing Alfresco Share on page 121 for some easy customizations.

6. Increase the database connection pool size for large numbers of concurrent users or sessions.
7. Use the System Information to view system properties, such as schema and server versions.
8. Use the Node Browser (searchable by node reference, xpath, or lucene) to view all properties, parent and child nodes, aspects applied, permissions, and associations.

Glossary

ACP
ACP (Alfresco Content Package) files hold exported information produced when using the Export feature.

alf_data
Directory containing binary content and Lucene indexes.

alfresco-global.properties
The alfresco-global.properties file contains the customizations for extending Alfresco. The standard global properties file that is supplied with the installers and bundles contains settings for the location of the content and index data, the database connection properties, the location of third-party software, and database driver and Hibernate properties.

Alfresco WAR
The Alfresco Web application Archive (WAR) file is for deployment in existing application servers.

AMP
AMP (Alfresco Module Package) is a collection of code, XML, images, CSS, that collectively extend the functionality or data provided by the standard Alfresco repository. An AMP file can contain as little as a set of custom templates or a new category. It can contain a custom model and associated user interface customizations. It could contain a complete new set of functionality, for example records management.

AVM
Alfresco Advanced Versioning Manager (AVM) is an advanced store implementation designed to support the version control requirements of large websites and web applications.

breadcrumb
A navigation link that allows you to jump to any part of the breadcrumb path.

<classpathRoot>
The <classpathRoot> is the directory whose contents are automatically added to the start of your application server's classpath. The location of this directory varies depending on your application server.

<configRoot>
The `<configRoot>` directory is where the default configuration files are stored. Where possible, you should not edit these files but you can edit the overrides in the `<extension>` directory. For example, for Tomcat, `<configRoot>` is: `<TOMCAT_HOME>/webapps/alfresco/WEB-INF/`

<configRootShare>
The `<configRootShare>` directory is where the default configuration files for Share are stored. Where possible, you should not edit these files but you can edit the Share override files in the `<web-extension>` directory. For example, for Tomcat, `<configRootShare>` is `<TOMCAT_HOME>/webapps/share/WEB-INF`

CIFS
Microsoft Common Internet File System (CIFS) is a network file system for sharing files across the Internet.

content
Files or documents made of two main elements: the content itself and information about the content (metadata). For example, documents, video, audio, images, XML, and HTML.

dashboard
The Alfresco dashboard is an interactive user interface that presents and organizes information to the user.

dashlet
A dashlet is an application that appears in the Alfresco dashboard that presents information to the user. Users can organize dashlets into different layouts and set keyboard short cuts for each dashlet.

dir.root
The `dir.root` property is specified in the `alfresco-global.properties` file. It points to the directory `alf_data`, which contains the content and the Lucene indexes.

dir.indexes
This folder contains all Lucene indexes and deltas against those indexes.

Enterprise Content Management (ECM)
Enterprise Content Management (ECM) is a set of technologies used to capture, store, preserve and deliver content and documents and content related to organizational processes. ECM tools and strategies allow the management of an organization's unstructured information, wherever that information exists.

Quoted from: http://en.wikipedia.org/wiki/Enterprise_content_management

Enterprise Edition
The Enterprise Edition is production-ready open source. It is the stress-tested, certified build that is supported by Alfresco Software. It is recommended for corporations, governments, and other organizations looking for a production-ready open source ECM solution, with the primary benefit of being a stable, reliable, certified, supported application with warranty and indemnity, with the support of Alfresco and its certified partners.

<extension>
The `<extension>` directory is where you store files that extend and override the Alfresco default files. When Alfresco is installed, there are sample files in this directory. Many of these files have a `.sample` suffix, which must be removed to activate the file.

For example: for Tomcat, `<extension>` is:`<TOMCAT_HOME>/shared/classes/alfresco/extension/`

Hibernate
Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. Hibernate solves Object-Relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions.

ImageMagick
ImageMagick is a software suite to create, edit, and compose bitmap images. It can read, convert and write images in a large variety of formats. Images can be cropped, colors can be changed, various effects can be applied, images can be rotated and combined, and text, lines, polygons, ellipses and Bézier curves can be added to images and stretched and rotated.

Quoted from: http://www.imagemagick.org/script/index.php

Java Content Repository (JCR) API
Java Content Repository API is a standard Java API (as defined by JSR-170) for accessing content repositories. Alfresco provides support for JCR level 1 and level 2 giving standardized read and write access.

Java Management Extension (JMX) interface
The JMX interface allows you to access Alfresco through a standard console that supports JMX remoting (JSR-160). Example consoles include, JConsole, MC4J, and JManage.

JVM
Java Virtual Machine

Lucene
Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform.

Quoted from: http://lucene.apache.org/java/docs/index.html

repository
The repository is the combination of the content, the indexes, and the database.

sandbox
An environment for testing, experimenting, or using as a working area. In WCM, a sandbox is an area where users can make changes to web content. In the sandbox, a user can add, edit, and delete both folders and files.

site
A site is a collaborative area for a unit of work or a project.

Spring
Spring is an open-source application framework for Java/JEE. The Alfresco repository uses the Spring Framework as the core foundation of its architecture.

store
A store is a logical partition within the repository, grouped for a particular automated use. Each store contains a hierarchy of nodes with one root node. Two main stores in Alfresco are the Document Management (DM) and the Advanced Versioning Manager (AVM) stores. AVM stores are used for Web Content Management. Each store is identified by a content store reference. This reference consists of a store protocol (such as archive or workspace) and a store id (such as SpaceStore or User).

WAR
The Alfresco Web application ARchive (WAR) file is for deployment in existing application servers.

Web Content Management (WCM)
Web Content Management is an Alfresco product for the rapid deployment of web content, allowing users to create, develop, and maintain content for websites.

WebDAV
Web-based Distributed Authoring and Versioning. A protocol that allows users to edit and manage files on remote web servers.

<web-extension>
The `<web-extension>` directory is where you store files that extend and override the Alfresco default files for Alfresco Share. When Alfresco is installed, there are sample files in this directory. Many of the files have a `.sample` suffix, which must be removed to activate the file.

For example: for Tomcat, `<web-extension>` is:`<TOMCAT_HOME>/shared/classes/alfresco/web-extension/`

workflow
A workflow is a work procedure and workflow steps that represent the activities users must follow in order to achieve the desired outcome. Alfresco provides two different types of workflow: simple and advanced. Simple workflow defines content rules for a space. Advanced workflow provides two out-of-the-box workflows (Review and Approve; Adhoc Task).