Alfresco™ Enterprise

Enterprise 4.1.5

Alfresco Enterprise 4.1.5 Administrator

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 sections:

- **Installing Alfresco** describes how to install Alfresco and components
- **Upgrading Alfresco** describes how to upgrade Alfresco and components
- **Administering Alfresco** describes how to configure, maintain, and manage the system
- **Troubleshooting** describes how to analyze and troubleshoot various scenarios
- **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.

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, text the user types, 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.

Installing

Depending on your system, you can install Alfresco using one of the following methods:

- Using a setup wizard, which contains the required software and components you need for evaluating Alfresco
- Using a standard WAR file to deploy Alfresco in a production environment

Installing Alfresco using setup wizards

This section includes instructions that describe the quickest way to install Alfresco using the setup wizards.

Installing Alfresco Enterprise on Linux

The setup wizard for Linux installs all the software and components that you require for running Alfresco. This setup wizard installs Alfresco and additional software, including a Tomcat application server, PostgreSQL database, JDK, OpenOffice, SWFTools, and ImageMagick.

1. Download the following installation file:

   `alfresco-enterprise-4.1.5-installer-linux-x64.bin`

   This Alfresco setup wizard is for 64-bit Linux systems.
2. Execute the downloaded file using the following commands:

   ```
   chmod 777 the bin file
   sudo ./alfresco-enterprise-4.1.5-installer-linux-x64.bin
   ```

   The setup wizard starts.
3. Select the language that you wish to use for the installation.

   This sets the language to be used for the setup wizard.
4. On the **Setup - Alfresco Enterprise** window, click **Next**.
5. On the **Installation type** window, choose how you want to use the setup wizard.

   There are two types of installation in the setup wizard:

Options	Description
Easy	Easy type installs Alfresco using the default options and configuration. This install type requires you to enter information in only two fields: the Alfresco install location and the administrator password. Choose this route to install Alfresco with the default environment. ✎ If you have previously installed Alfresco and the server is running, when you run this setup wizard again, you may be prompted to enter alternative port numbers for the components and services that you install, for example, for the Tomcat application server, FTP port, and the RMI port.

Options	Description
Advanced	Advanced type installs Alfresco but lets you configure the server ports and service properties. You can also choose which additional components to install.

To complete the **Easy** setup wizard:

a. Select **Easy**, and then click **Next**.

b. On the **Installation folder** window, click **Next** to accept the default location.

c. On the **Admin Password** window, enter a password for the Administrator user (`admin`).

d. Repeat the password, and then click **Next**.

e. Click **Next** through the remaining windows in the setup wizard.

f. Click **Finish** to complete the installation.

   Go to the step for the **Completing the Alfresco Enterprise Setup Wizard** window and launching Alfresco Share.

To complete the **Advanced** setup wizard, select **Advanced** and then click **Next**.

Follow the remaining steps in this task.

6. On the **Select Components** window, select the components that you want to install. Deselect the components that you do not want to install.

   You can select from the following components:

   - Java
   - PostgreSQL
   - SharePoint
   - Web Quick Start
   - OpenOffice

   ✎ You cannot deselect the Alfresco component because it is installed by default.

7. When you have finished selecting the components, click **Next**.
8. On the **Installation folder** window, click **Next** to accept the default location.

   For example, the default location is `/opt/alfresco-4.1.x`.

   Alternatively, click the 📂 icon to choose another location.
9. On the **Database Server Parameters** window, enter a port number for your database.

   Enter a suitable port number or click **Next** to accept the default of 5432.
10. On the **Tomcat Port Configuration** window, enter the following Tomcat configuration parameters:

    a. Web Server Domain

    For example, the default is 127.0.0.1.

    The URL http://127.0.0.1:8080/share is based on the web server domain and the Tomcat port number that you specify on the **Tomcat Port Configuration** window. The default of 127.0.0.1 can be used on this machine to verify that Alfresco is running successfully. However, it is not an externally addressable URL, which means that it is not possible for users on other machines to access this URL. To make sure that other users can access the machine where Alfresco is installed, you need to define and create a publicly addressable name.

b. Tomcat port

   For example, the default is 8080.

c. Tomcat Shutdown port

   For example, the default is 8005.

d. Tomcat SSL Port

   For example, the default is 8443.

e. Tomcat AJP Port

   For example, the default is 8009.

11. On the **Alfresco FTP Port** window, enter a port number for the Alfresco FTP server, and then click **Next**.
12. On the **Alfresco RMI Port** window, enter a port number for the RMI service, and then click **Next**.
13. On the **Admin Password** window, type a password. Repeat the password, and then click **Next**.

    This sets the password for the Alfresco Administrator user account (`admin`).
14. (Optional) If you are installing SharePoint Protocol Support, the **Alfresco SharePoint Port** window displays. Enter a port number, and then click **Next**.
15. (Optional) If you are installing the OpenOffice component, the **OpenOffice Server Port** window displays. Enter a port number on which the OpenOffice server will listen, and then click **Next**.

    If you are installing OpenOffice, the system requires the following additional libraries, which are not supplied with the Alfresco installer.

    - `linux-vdso.so.1`
    - `libXext.so.6`
    - `libm.so.6`
    - `libc.so.6`
    - `libdl.so.2`
    - `libpthread.so.0`
    - `libfreetype.so.6`
    - `libX11.so.6`
    - `libXau.so.6`
    - `ld-linux-x86-64.so.2`
    - `libXdmcp.so.6`

    If these libraries are not already available on your system, you may need to install them. For example, you can use your system's package manager to install the following packages: `libXext`, `glibc`, `freetype`, `libX11`, `libXau`, and `libXdmcp`.
16. On the **Service Startup Configuration** window, you are presented with two options for starting up the Alfresco services.

Options	Description
Manual	Sets the services to be started manually.

Options	Description
Auto	Sets the services to start up automatically when you start your machine.

Select the services start up option, and then click **Next**.

17. On the **Ready to Install** window, click **Next**.

    The **Installing** window displays, showing the progress of the installation.

18. On the **Completing the Alfresco Enterprise Setup Wizard** window, click **Finish**.

    This window shows check boxes that determine whether you will see the Readme file, the Getting Started web page, and also whether to launch Alfresco Share. By default, these options are selected and will launch when you click **Finish**. If you do not want to start Alfresco at this point, deselect the **Launch Alfresco Enterprise Share** check box.

19. Click **OK** to close the Readme.

    The Alfresco server starts and then Alfresco Share launches in your default browser.

    It may take several minutes to start the Alfresco server and to launch Share. Your browser opens and tries to connect to http://127.0.0.1:8080/share.

20. Log on to Alfresco Share as the `admin` user. Enter the password that you specified in the **Admin Password** window.

    The Alfresco server is launched automatically as a service called `alfresco`. This service comprises the following individual services:

    - `postgresql`
    - `Tomcat Server`

    If you did not automatically launch Alfresco at the end of the setup wizard, to start Alfresco, you need to start all the services.

21. Manually start the Alfresco server:

    ```
    service alfresco start
    ```

    To start only the `tomcat` service:

    ```
    service alfresco start tomcat
    ```

22. To fully stop Alfresco, you must stop all the services:

    ```
    service alfresco stop
    ```

Installing Alfresco Enterprise on Windows

The setup wizard for Microsoft Windows installs all the software and components that you require for running Alfresco. This setup wizard installs Alfresco and additional software, including a Tomcat application server, PostgreSQL database, JDK, OpenOffice, SWFTools, and ImageMagick.

1. Download the following installation file:

   ```
   alfresco-enterprise-4.1.5-installer-win-x64.exe
   ```

   The Alfresco setup wizard is for 64-bit Windows systems.

2. Double-click the downloaded file.
3. Select the language that you wish to use for the installation.

   This sets the language to be used for the setup wizard.

4. On the **Setup - Alfresco Enterprise** window, click **Next**.

5. On the **Installation type** window, choose how you want to use the setup wizard.

   There are two types of installation in the setup wizard:

Options	Description
Easy	**Easy** type installs Alfresco using the default options and configuration. This install type requires only two fields: install location and administrator password. Choose this route to install Alfresco with the default environment. If you have previously installed Alfresco and the server is running, when you run this setup wizard again, you may be prompted to enter alternative port numbers for the components and services that you install, for example, for the Tomcat application server, FTP port, and the RMI port.
Advanced	**Advanced** type installs Alfresco but lets you configure the server ports and service properties. You can also choose which additional components to install.

   To complete the **Easy** setup wizard:

   a. Select **Easy**, and then click **Next**.
   b. On the **Installation folder** window, click **Next** to accept the default location.
   c. On the **Admin Password** window, enter a password for the Administrator user (`admin`).
   d. Repeat the password, and then click **Next**.
   e. Click **Next** through the remaining windows in the setup wizard.
   f. Click **Finish** to complete the installation.

      Go to the step for the **Completing the Alfresco Enterprise Setup Wizard** window and launching Alfresco Share.

   To complete the **Advanced** setup wizard, select **Advanced** and then click **Next**.

   Follow the remaining steps in this task.

6. On the **Select Components** window, select the components that you want to install. Deselect the components that you do not want to install.

   You can select from the following components:

   - Java
   - PostgreSQL
   - SharePoint
   - Web Quick Start
   - OpenOffice

   You cannot deselect the Alfresco component because it is installed by default.

7. When you have finished selecting the components, click **Next**.
8. On the **Installation folder** window, click **Next** to accept the default location.

   For example, the default location is `C:\Alfresco`.

Alternatively, click the 📂 icon to choose another location.

9. On the **Database Server Parameters** window, enter a port number for your database.

   Enter a suitable port number or click **Next** to accept the default of 5432.

10. On the **Tomcat Port Configuration** window, enter the following Tomcat configuration parameters, and then click **Next**.

    a. Enter the Web Server domain number.

       For example, the default is 127.0.0.1.

       The URL http://127.0.0.1:8080/share is based on the web server domain and the Tomcat port number that you specify on the **Tomcat Port Configuration** window. The default of 127.0.0.1 can be used on this machine to verify that Alfresco is running successfully. However, it is not an externally addressable URL, which means that it is not possible for users on other machines to access this URL. To make sure that other users can access the machine where Alfresco is installed, you need to define and create a publicly addressable name.

    b. Enter the port number for the Tomcat web application.

       For example, the default is 8080.

    c. Enter the Tomcat Shutdown port number.

       For example, the default is 8005.

    d. Enter the Tomcat SSL port number.

       For example, the default is 8443.

    e. Enter the Tomcat AJP Port number.

       For example, the default is 8009.

11. On the **Alfresco FTP Port** window, enter a port number for the Alfresco FTP server, and then click **Next**.

12. On the **Alfresco RMI Port** window, enter a port number for the RMI service, and then click **Next**.

13. On the **Admin Password** window, enter a password. Repeat the password, and then click **Next**.

    This sets the password for the Alfresco Administrator user account (`admin`).

14. (Optional) If you are installing SharePoint Protocol Support, the **Alfresco SharePoint Port** window displays. Enter a port number, and then click **Next**.

15. (Optional) If you are installing the OpenOffice component, the **OpenOffice Server Port** window displays. Enter a port number on which the OpenOffice server will listen, and then click **Next**.

16. On the **Service Startup Configuration** window, you are presented with two options for starting up the Alfresco services.

Options	Description
Manual	Sets the services to be started manually. Choose this option if you want to start the services yourself.
Auto	Sets the services to start up automatically when you restart the machine.

    Select the services start up option, and then click **Next**.

17. On the **Ready to Install** window, click **Next**.

    The **Installing** window displays, showing the progress of the installation.
18. On the **Completing the Alfresco Enterprise Setup Wizard** window, click **Finish**.

    This window shows check boxes that determine whether you will see the Readme file, the Getting Started web page, and also whether to start the server and launch Alfresco Share. By default, these options are selected and will start when you click **Finish**. If you do not want to start Alfresco at this point, deselect the **Launch Alfresco Enterprise Share** check box.
19. Click **OK** to close the Readme.

    The Alfresco server starts and then Alfresco Share launches in your default browser.

    ⚠ It may take several minutes to start the Alfresco server and to launch Share. Your browser opens and tries to connect to http://127.0.0.1:8080/share.
20. Log on to Alfresco Share as the `admin` user. Enter the password that you specified in the **Admin Password** window.

    The Alfresco server is launched as a Windows service. To manage the server, open the Control Panel **Services** window. The services that will be running for an Alfresco install using the default options are:

    - `alfrescoPostgreSQL`
    - `alfrescoTomcat`

    If you did not automatically launch Alfresco at the end of the installation wizard, to start Alfresco, you need to start all the services. Use the `servicerun start` script in the installation directory or select **All Programs > Alfresco Enterprise > Alfresco Enterprise Service > Start Alfresco Enterprise Service**.
21. To fully stop Alfresco, you must stop all the services. Use the scripts in the installation directory to start or stop the services: `servicerun start` and `servicerun stop`.

Installing Alfresco Enterprise

This section provides information for manually installing Alfresco Enterprise.

Software requirements

The following table lists the required software that must be on your system for manually installing Alfresco.

Component	Recommendation
Java SE Development Kit (JDK)	The Sun Microsystems JDK 6 is required. The `JAVA_HOME` environment variable must be set to the location of the JDK installation.
Application server	Alfresco runs within an application server. Alfresco Enterprise runs within Tomcat but can be installed on other application servers. For information on installing Alfresco with other supported application servers, see Installing Alfresco on JBoss and Installing Alfresco on WebLogic.
Database	Alfresco comes preconfigured with the PostgreSQL database. If you intend to use Alfresco in a production environment, you can use one of the supported databases. For the latest information on supported databases, refer to the Alfresco website. For information on configuring the database settings, refer to Configuring databases.

Component	Recommendation
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. Use the latest (stable) version of OpenOffice.org.
ImageMagick	Alfresco uses ImageMagick to manipulate images for previewing.
Flash Player	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. Use the latest (stable) version of Flash Player for your platform.
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.

Language support

The Alfresco Share interface is supported for use with a number of languages that have been through an Engineering QA and linguistic testing cycle.

Alfresco is supported with the following languages:

- English
- German
- French
- Spanish
- Italian
- Japanese
- Dutch
- Russian
- Norwegian
- Simplified Chinese

You can select the language when you install Alfresco using the setup wizards. Before you install a localized version, ensure that your browser is set up to view the relevant locale. This ensures that the special characters display correctly in your installed instance.

The source-localized files are encoded in ASCII, and the special and accented characters are displayed using escape sequences. The source files have been renamed using the corresponding locale for each language. For example, for the French version, `site-welcome.properties` is called `sitewelcome_ fr.properties`.

Although the Share interface is localized, the following components have not been localized, therefore, any strings originating from these components in Share and Explorer will be displayed in English.

- SharePoint
- Web Quick Start
- OpenOffice

The following files are not localized and the error messages remain in English to ease searching for fixes to issues.

- `content-service.properties`
- `dictionary-messages.properties`

- `jbpm-engine-messages.properties`
- `module-messages.properties`
- `patch-service.properties`
- `repoadmin-interpreter-help.properties`
- `schema-update.properties`
- `system-messages.properties` (partially translated)
- `tenant-interpreter-help.properties`
- `version-service.properties`
- `webclient-config-admin-interpreter-help.properties`
- `workflow-interpreter-help.properties`
- `control.properties` (in remote-api directory)

Production environment checklist

This section provides a check list for validating the architecture on which Alfresco will run and also for validating the production environment prior to installing Alfresco.

Validating the architecture

This section describes the steps required to validate the architecture to ensure that it meets the prerequisites for an Alfresco installation.

1. Check the supported stacks list.

   Validate that your environment is on the supported stacks list on http://www.alfresco.com.

2. Validate and optimize the hardware (I/O subsystems and CPU) settings.

   a. Optimize the following I/O, in this order of priority:

      - I/O to the relational database that Alfresco is configured to use.
      - I/O to the disk subsystem on which the Lucene indexes are stored.
      - I/O to the disk subsystem on which the content is stored.

      I/O is one of the main factors that influence Alfresco performance. In each case, the goal is to minimize the latency (response time) between Alfresco and the storage system, while also maximizing bandwidth. Low latency is particularly important for database I/O, and one rudimentary test of this is to ping the database server from the Alfresco server. Round trip times greater than 1ms indicate a suboptimal network topology or configuration that will adversely impact Alfresco performance. “Jitter” (highly variable round trip times) is also of concern, as that will increase the variability of Alfresco’s performance. The standard deviation for round trip times should be less than 0.1ms.

   b. Ensure that your system has a clock speed of greater than 2.5Ghz.

      For production use, this clock speed will ensure reasonable response times to the end user. Alfresco Enterprise 3.x and later versions have been tested on 64-bit CPU architectures, primarily because it allows the JVM to use more memory (RAM) that the earlier 32-bit CPU architecture.

      CPU clock speed is of particular concern for the Sun UltraSPARC architecture, as some current UltraSPARC based servers ship with CPUs that have clock speeds as low as 900Mhz, well below what is required for adequate Alfresco performance. If you intend to use Sun servers for hosting Alfresco, ensure that all CPUs have a clock speed of at least 2.5Ghz.

        This implies that:

        - An X or M class Sun server is required, with careful CPU selection to ensure 2.5Ghz (or better) clock speed.
        - T class servers should not be used, as they do not support CPUs faster than approximately 2Ghz. Alfresco is unable to provide specific guidance on Sun server classes, models, or configurations, so you should talk with your Sun reseller to confirm that minimum CPU clock speed recommendations will be met.

    c. Ensure that you allocate extra virtual memory on Linux systems. This extra space is required for processes within the Alfresco server that use the fork operation (for example, ImageMagick). Allocating this extra space ensures that Alfresco has sufficient memory to complete fork operations without reserving extra RAM.

3. Validate the database.

    Alfresco does not provide technical support for maintaining or tuning your relational database. Ensure that your project has access to a certified database administrator (DBA) to support your Alfresco installation.

    Regular maintenance and tuning of the Alfresco database is necessary. Specifically, all of the database servers that Alfresco supports require at the very least that some form of index statistics maintenance be performed at frequent, regular intervals to maintian optimal Alfresco performance.

    Index maintenance can have a severe impact on Alfresco performance while in progress, hence it needs to be discussed with your project team and scheduled appropriately.

4. Validate the Operating System.

    a. Ensure that your chosen OS has been officially certified for use with Alfresco (refer to the Supported Stacks list for details).

    b. Alfresco recommends that a 64-bit OS is used. See the Supported Stacks list for information on the exceptions.

5. Validate and tune the JVM.

    Ensure that your chosen JDK-enabled Java Virtual Machine has been officially certified for use with Alfresco (refer to the Supported Stacks list for details).

    For information on configuring and tuning the JVM, refer to Tuning the JVM.

    Alfresco requires an official Sun JDK. Other JVMs (including OpenJDK, Harmony, gcj, JRockit, IBM, HP, and so on) are not supported. Alfresco recommends using a 64-bit Sun JVM if the underlying platform (operating system and hardware) is 64-bit capable.

Validating the environment

The following environment-specific items must be validated prior to installing Alfresco.

An Environment Validation tool is also available that can validate most of the following requirements. This tool is available from the Alfresco Support Portal in **Online Resources > Knowledge base** http://support.alfresco.com.

1. Validate that the host name of the server can be resolved in DNS.

    This is required if Alfresco is going to be configured in a cluster.

    Using an incorrect host name or a host name that no longer resolves to its own IP address can give an internal error, such as `ObjID already in use`. You can get more information about this error in the `log4j.properties` file using the following command:

```
log4j.logger.org.springframework.remoting.rmi.RmiServiceExporter=debug
```

To resolve this error, you can either:

- Correct the host name using the following command:

  ```
  hostname ip-10-58-197-161
  ```

- Specify the correct IP address in the `alfresco-global.properties` file as shown below:

  ```
  alfresco.rmi.services.host=10.20.30.40
  ```

2. Validate that the user Alfresco will run as can open sufficient file descriptors (4096 or more).
3. Validate that the ports on which Alfresco listens are available:

The ports listed in the following table are the defaults. If you are planning to reconfigure Alfresco to use different ports, or wish to enable additional protocols (such as HTTPS, SMTP, IMAP or NFS), update this list with those port numbers.

Protocol	Port number	Notes
FTP	TCP 21	On Unix-like operating systems that offer so-called “privileged ports”, Alfresco will normally be unable to bind to this port, unless it is run as the root user (which is not recommended). In this case, even if this port is available, Alfresco will still fail to bind to it, however for FTP services, this is a non-fatal error. The Alfresco FTP functionality will be disabled in the repository.
SMTP	TCP 25	SMTP is not enabled by default.
SMB/NetBT:	UDP 137,138	
SMB/NetBT:	TCP 139,445	On Unix-like operating systems that offer so#called “privileged ports”, Alfresco will normally be unable to bind to this port, unless it is run as the root user (which is not recommended). In this case, even if this port is available, Alfresco will still fail to bind to it, however for CIFS services, this is a non-fatal error. The Alfresco CIFS functionality will be disabled in the repository.
IMAP	TCP 143	IMAP is not enabled by default.
SharePoint Protocol	TCP 7070	This port is only required if you install support for the SharePoint Protocol.
Tomcat Administration	TCP 8005	

Protocol	Port number	Notes
HTTP	TCP 8080	
RMI	TCP 50500	

4. Validate that the installed JVM is Sun version 1.6.
5. Validate that the directory in which the JVM is installed does not contain spaces.
6. Validate that the directory in which Alfresco is installed does not contain spaces.
7. Validate that the directory Alfresco will use for the repository (typically called `alf_data`) is both readable and writeable by the operating system user that the Alfresco process will run as.
8. Validate that you can connect to the database as the Alfresco database user, from the Alfresco server.

   Ensure that you install the database vendor's client tools on the Alfresco server.
9. Validate that the character encoding for the Alfresco database is UTF-8.
10. (MySQL only) Validate that the storage engine for the Alfresco database is InnoDB.
11. Validate that the following third-party software is installed and the correct versions:
    a. OpenOffice v3.1 or newer
    b. ImageMagick v6.2 or newer
12. (RHEL and Solaris only) Validate that OpenOffice is able to run in headless mode.

Alfresco Enterprise installation files

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.

The setup wizards install all the components you need for running Alfresco and ensure that you have all the recommended software installed and that configurations are set. When you install Alfresco using the setup wizards, it runs within an instance of the Tomcat application server.

If you wish to install Alfresco within an existing Tomcat or another application server, use the Alfresco WAR file. If you use the WAR file to install Alfresco, you must install the required additional components manually.

The following sections help you to determine what files to download and install.

Alfresco setup wizards

The setup wizards provide a full Alfresco install, which you can use if no Alfresco component is installed on your production environment system.

Description	File name
Setup wizard for **Windows**	`alfresco-enterprise-4.1.5-installer-win-x64.exe` (64 bit) The Alfresco setup wizard for Windows is for 64-bit systems. It is not suitable for use on 32-bit environments.

Description	File name
Setup wizard for **Linux**	`alfresco-enterprise-4.1.5-installer-linux-x64.bin` (64 bit) The Alfresco setup wizard for Linux is for 64-bit systems. It is not suitable for use on 32-bit environments. ✎ The Linux executable file is a graphical installer, but you can also run this file to install Alfresco using text mode. Text mode is a keyboard-based installation method. Run the command with the `--mode text` option.

Alfresco WAR installation file

Description	File name
Alfresco WAR files for manual install into existing application servers or for upgrades to existing Alfresco installations. This file also contains the Module Management tool and JCR benchmarking tool. Includes the sample extension files, such as `alfresco-global.properties`.	`alfresco-enterprise-4.1.5.zip`

Enterprise EAR file

Description	File name
Enterprise EAR file includes the sample extension files, such as `alfresco-global.properties`, and also contains the alfresco-enterprise.ear file and myfaces1_1-websphere-shared-lib.zip.	`alfresco-enterprise-ear-4.1.5.zip`

Solr search

Description	File name
Solr search installation file	`alfresco-enterprise-solr-4.1.5.zip`

File System Transfer Receiver

Description	File name
FSTR installation file	`alfresco-enterprise-file-transfer-receiver-4.1.5.zip`

SharePoint Protocol Support

Description	File name
Microsoft SharePoint Protocol Support functionality	`alfresco-enterprise-spp-4.1.5.zip`

Alfresco WCM

Description	File name
Web Quick Start bundle	`alfresco-enterprise-wcmqs-4.1.5.zip`
Alfresco Web Editor	`alfresco-enterprise-webeditor-4.1.5.zip`

Description	File name
Deployment receiver installation file for Windows	`alfresco-enterprise-deployment-4.1.5-win.exe`
Deployment receiver installation file for Linux	`alfresco-enterprise-deployment-4.1.5-linux.bin`
Forms developer kit	`alfresco-enterprise-fdk-4.1.5.zip`

Alfresco Records Management

Description	File name
Records Management zip, which includes the required core and Share AMPs	`alfresco-rm-2.0.2-171.zip`

Alfresco SDK AND APIs

Description	File name
Alfresco Software Development Kit, including the source files	`alfresco-enterprise-sdk-4.1.5.zip`

Alfresco Web Service client

Description	File name
WSDL-based API providing standard remote access to the Alfresco repository	`alfresco-web-service-client-4.15.zip`

Downloading Enterprise installation files

This section describes the location of the Enterprise installation files that are available for download.

1. Browse to the Alfresco Support Portal at http://support.alfresco.com.
2. On the **Sign in** page, enter your registered email address and password, and then click **Sign in**.
3. Click **Online Resources**.
4. Click **Downloads**.

   The **Downloads** page initially shows the release notes for all the Alfresco versions. Click **More...** for the full list of release notes available for each version.
5. On the left navigation menu, select the directory for the Alfresco version you require.

   The list of files displays on the **Downloads** page.
6. Click the link for the file you want to download.
7. Click **OK** to download the file to your local machine.

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

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, refer to the **Supported Platforms** page at http://www.alfresco.com/services/subscription/supported-platforms/.

Installing Alfresco on Tomcat

For more complex Alfresco installations or if you wish to use an existing Tomcat application server, you can use the Web Archive (WAR) bundle to install Alfresco on any platform. For this type of installation, you must ensure that the required software is installed on the machine.

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

Configuring Alfresco as a Windows service

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

1. Open a command prompt.
2. To install Alfresco as a Windows service, enter the following commands:

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

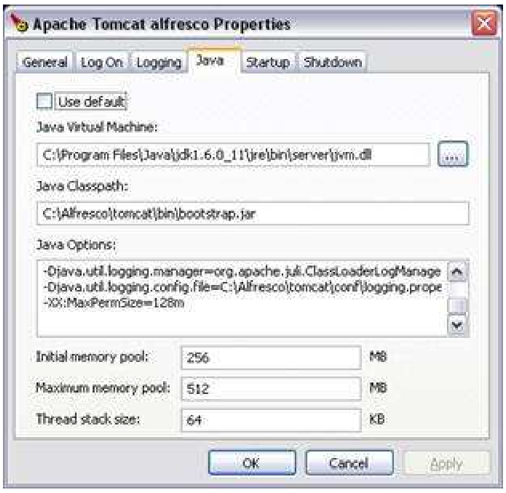
   For Tomcat 6:

   ```
   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
   ```

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 Tomcat application server

This section describes how to install an instance of Tomcat manually and modify it to use the correct directory structure and files for Alfresco.

The installation directory for Tomcat is referred to as `<TOMCAT-HOME>`.

These instructions recommend that you name the required directories as `shared/classes` and `shared/lib` because these are the path names used within full Alfresco installations. You can substitute alternative names for these directories.

1. Download Tomcat from http://tomcat.apache.org.

   See the Alfresco Supported Platforms page for the correct version to download.
2. Install Tomcat following the instructions included in the release.
3. Create the directories required for an Alfresco installation:
   a. Create the `shared/classes` directory.
   b. Create the `shared/lib` directory.
4. Edit the `<TOMCAT-HOME>/conf/catalina.properties` file.
5. Change the value of the `shared.loader=` property to the following:

   ```
   shared.loader=${catalina.home}/shared/classes,${catalina.home}/shared/
   lib/*.jar
   ```

   If you have used alternative names for the directories, you must specify these names in the `shared.loader` property.
6. Save the `catalina.properties` file.
7. Copy the JDBC drivers for the database you are using to:

   ```
   lib/
   ```
8. Edit the `<TOMCAT_HOME>/conf/server.xml` file.
9. Set attributes of HTTP connectors.

   By default, Tomcat uses ISO-8859-1 character encoding when decoding URLs that are received from a browser. This may cause problems when creating, uploading, and renaming files with international characters.

   By default, Tomcat uses an 8K header buffer size, which may not be large enough for Kerberos and NTLM authentication protocols.

   Locate the `Connector` sections, and then add the `URIEncoding="UTF-8"` and `maxHttpHeaderSize="32768"` properties.

   ```
   <Connector port="8080" protocol="HTTP/1.1" URIEncoding="UTF-8"
    connectionTimeout="20000" redirectPort="8443" maxHttpHeaderSize="32768"/
   >
   ```
10. Save the `server.xml` file.
11. There is an issue with Alfresco Share document downloads on Tomcat with https (SSL) for Internet Explorer versions 7 and 8. On IE7 and IE8, you will see an error message if you try to download a document from Alfresco Share in Tomcat with https (SSL) enabled. To resolve this issue:
    a. Edit the `<TOMCAT-HOME>/conf/context.xml` file.
    b. Add the following line to the `context` element:

    ```
    <Valve className="org.apache.catalina.authenticator.SSLAuthenticator"
     securePagesWithPragma="false" />
    ```
    c. Save the `<TOMCAT-HOME>/conf/context.xml` file.

Installing the Alfresco WAR

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

The Alfresco WAR file is a bundle file containing the required WAR files, in addition to the additional commands, configuration files, and licenses for a manual installation.

1. Browse to the Alfresco Enterprise download area.
2. Download the following file:

   alfresco-enterprise-4.1.5.zip
3. Specify a location for the download and extract the file.

   You see the following directory structure:

```
bin
licenses
web-server
```

The WAR bundle also contains the following file:

```
README.txt
```

The following files are contained within the suggested subdirectories for within the Tomcat application server.

/bin

File name	Description
alfresco-bm.jar	The JCR Benchmarking toolkit.
alfresco-mmt.jar	The Alfresco Module Management Tool (MMT).
apply_amps.bat	Windows batch file for Tomcat application server installs, used to apply all AMP files in the <installLocation> directory.
apply_amps.sh	Linux script file for Tomcat application server installs, used to apply all AMP files in the <installLocation> directory.
clean_tomcat.bat	Windows batch file for cleaning out temporary application server files from previous installations.
clean_tomcat.sh	Linux script for cleaning out temporary application server files from previous installations.
Win32NetBIOS.dll	Required for CIFS.
Win32NetBIOSx64.dll	Required for CIFS on 64-bit Windows.
Win32Utils.dll	Required for CIFS.
Win32Utilsx64.dll	Required for CIFS on 64-bit Windows.

The /licenses directory contains the following structure:

```
3rd-party
```

This directory contains the third-party license files.

The web-server directory contains the following structure:

```
endorsed
lib
shared
webapps
```

/endorsed

This directory contains the Java libraries that should be copied to your application server's endorsed directory (for example, tomcat/endorsed). Some Alfresco features require Xalan and its dependencies to be in the endorsed directory because they use XSLT features that are not available in the xsltc implementation built into the JDK.

File name	Description
serializer.jar	Serializer classes of Xalan-Java.

File name	Description
`xalan.jar`	XSLT processor for transforming XML documents into HTML, text, or other XML document types.

`/lib`

File name	Description
`postgresql-version.jdbc4`	PostgreSQL database JDBC connector file.

`/shared`

File name	Description
`/classes/alfresco-global.properties.sample`	The global properties file, which is used for Alfresco configuration properties.
`/classes/alfresco`	Contains the Alfresco directory structure for the configuration override files, including the `extension` and `web-extension` directories.

`/webapps`

File name	Description
`alfresco.war`	The Alfresco WAR file.
`share.war`	The Alfresco Share WAR file.

4. Move the `alfresco.war` file and `share.war` files to the appropriate location for your application server.

   For example, for Tomcat, move the `.war` files to the `<TOMCAT_HOME>/webapps` directory.
5. Edit the `/shared/classes/alfresco-global.properties.sample` file with your configuration settings.
6. Save the file without the `.sample` extension.
7. Move the `alfresco-global.properties` file to `<classpathRoot>`.

   For example, `<TOMCAT_HOME>/shared/classes`.

If you deployed previous versions of Alfresco, you must remove any temporary files created by your application server. Use the `clean_tomcat.bat` or `clean_tomcat.sh` command.

Deploying Share into a separate Tomcat instance

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. 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 `share-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`
   c. Save the file without the `.sample` extension.
7. 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.

Ensure that JBoss is installed. Review the Supported Platforms page on the Support Portal for more information.

These steps assume that you know the path of the JBoss directory, which is represented as `<JBOSS_HOME>`.

1. Download JBoss EAP.

   For information on the correct version to download for your version of Alfresco see Supported stacks.
2. JBoss EAP has two installation options for Web Service support:
   - WSNative (selected by default)
   - WSCXF

   Make sure that you install JBoss with the WSNative option.
3. To prevent interference with the Web Service stacks used by Alfresco, follow the instructions in the RedHat Support article:

   How to remove JBossWS from JBoss EAP 5.x
4. Download the following file from the Support Portal.

   `alfresco-enterprise-4.1.5.zip`
5. Create a temporary directory in which to uncompress the file.

   When you uncompress the `alfresco-enterprise-4.1.5.zip` file, the directory structure contains the files that you need for installing Alfresco.
6. Copy the `alfresco-global.properties.sample` file from the temporary directory to the `<JBOSS_HOME>/server/default/conf` directory.

   The `alfresco-global.properties.sample` file is located in the `<TEMP_DIR>/web-server/shared/classes` directory.
7. Edit the parameters in the `alfresco-global.properties.sample` file to suit your environment.
8. Save the `alfresco-global.properties.sample` file without the `.sample` extension.
9. Navigate to the `<TEMP_DIR>/web-server/webapps` directory.
10. Copy or move the `alfresco.war` and `share.war` files to the `<JBOSS_HOME>/server/default/deploy` directory.
11. Follow the instructions for configuring JBoss for Alfresco.

Configuring JBoss for Alfresco

This section describes how to configure an Alfresco installation on JBoss.

These steps assume that you know the path of the JBoss directory, which is represented as `<JBOSS_HOME>`.

1. Configure the database. This example assumes you are using a PostgreSQL database.
   a. Copy the PostgreSQL driver JAR to `<JBOSS_HOME>/server/default/lib`.
   b. Follow the instructions for configuring the PostgreSQL database.
2. Configure UTF-8 support.
   a. Edit the following files:
      - `<JBOSS_HOME>/server/default/deploy/jbossweb.sar/server.xml`
      - `<JBOSS_HOME>/server/all/deploy/jbossweb.sar/server.xml`

   b. Add `URIEncoding="UTF-8"` to the following section:

```
<Connector protocol="HTTP/1.1" port="8080" URIEncoding="UTF-8"
 address="${jboss.bind.address}"
                     connectionTimeout="20000" redirectPort="8443" />

<!-- A AJP 1.3 Connector on port 8009 -->
<Connector protocol="AJP/1.3" port="8009" URIEncoding="UTF-8"
 address="${jboss.bind.address}" redirectPort="8443" />
```

3. Configure logging.
   a. Edit the `<JBOSS_HOME>/server/default/conf/jboss-log4j.xml` file to reduce the huge debug log output.

      For example:

```
<root>
     <priority value="INFO" />
     <appender-ref ref="CONSOLE"/>
     <appender-ref ref="FILE"/>
  </root>
```

   b. (Optional) The logging configuration that is set reduces the debug output but there will still be a large volume of output sent to the console. To reduce it further, add the following to the `<JBOSS_HOME>/server/default/conf/jboss-log4j.xml` file:

```
 <category name="org.jboss.logging.Log4jService$URLWatchTimerTask">
     <priority value="INFO"/>
  </category>
  <category name="org.jboss.system.server.Server">
     <priority value="INFO"/>
  </category>
  <category name="org.jboss">
     <priority value="WARN"/>
  </category>
  <category name="net">
     <priority value="WARN"/>
  </category>
  <category name="org.alfresco">
     <priority value="WARN"/>
  </category>
  <category name="org.alfresco.repo.policy">
     <priority value="WARN"/>
  </category>
  <category name="org.springframework">
     <priority value="WARN"/>
  </category>
  <category name="org.hibernate">
     <priority value="WARN"/>
  </category>
  <category name="org.hibernate.cache.ReadWriteCache">
     <priority value="ERROR"/>
  </category>
  <category name="org.hibernate.cache.EhCacheProvider">
     <priority value="ERROR"/>
  </category>
  <category
 name="org.hibernate.engine.StatefulPersistenceContext.ProxyWarnLog">
     <priority value="ERROR"/>
  </category>
  <category name="org.apache.myfaces">
     <priority value="ERROR"/>
  </category>
  <category name="org.jbpm.jpdl.xml.JpdlXmlReader">
     <priority value="ERROR"/>
  </category>
```

4. Configure Hibernate.
   a. Edit the <JBOSS_HOME>/server/default/deployers/ejb3.deployer/META-INF/jpa-deployers-jboss-beans.xml file, and then change the hibernate.bytecode.provider=javassist line to hibernate.bytecode.provider=cglib.

      For example:

      ```
      <entry>
        <key>hibernate.bytecode.provider</key>
        <value>cglib</value>
      </entry>
      ```

5. Open the <JBOSS_HOME>/bin/run.conf file.
   a. Set the conf directory in the global classpath to ensure that the Alfresco extensions and web-extensions directories are available:

      Add the JBOSS_CLASSPATH= setting, specifying the full path of your conf directory, for example:

      ```
      JBOSS_CLASSPATH="/opt/app_servers/jboss_ga/server/default/conf"
      ```

      An example for this setting on Windows is:

      ```
      set JBOSS_CLASSPATH=d:\app_servers\jboss_ga\server\default\conf
      ```

   b. Ensure that the <JBOSS_HOME>/bin/run.conf file specifies appropriate JVM memory parameters in JAVA_OPTS.

      For example, the following are minimums:

      ```
      -Xms128m -Xmx1024m -XX:MaxPermSize=256m
      ```

   c. Enable JMX monitoring and automatic discovery of Alfresco by ensuring that JAVA_OPTS contains the following parameters:

      ```
      -Dcom.sun.management.jmxremote -Dalfresco.home=.
      ```

   d. For running Alfresco on Windows 2003 Server with OpenOffice 3.2, ensure that the JAVA_OPTS contains the following parameters:

      ```
      -Djboss.server.temp.dir.overrideJavaTmpDir=true
      ```

   e. Save the run.conf file.
6. Edit the filtered packages list.
   a. Open the <JBOSS_HOME>/server/default/deployers/jbossweb.deployer/META-INF/war-deployers-jboss-beans.xml file, and then locate the filteredPackages property of the WarClassLoaderDeployer bean.
   b. Open the <extension>\war-deployers-jboss-beans.xml.fragment.sample file.

This sample file contains a `WarClassLoaderDeployer` bean definition fragment for use when configuring JBoss.

c. Copy the full `WarClassLoaderDeployer` bean from the sample file.

d. In the `war-deployers-jboss-beans.xml` file, paste the sample bean fragment over the `WarClassLoaderDeployer` bean.

   The bean definition contains the `filteredPackages` list that is required for Alfresco. This list is on one line and must not contain any breaks.

e. Save the `<JBOSS_HOME>/server/default/deployers/jbossweb.deployer/META-INF/war-deployers-jboss-beans.xml` file.

This setting avoids collisions between the JVM bootstrap classes and those embedded in the war.

7. Execute the `<JBOSS_HOME>/bin/run.sh` file.

   By default JBoss will only listen on the `localhost` network adapter, rather than the adapter with a real IP address connected to the outside world. To override this, start JBoss with the `-b addr` option, specifying the IP address of the network adapter you want to listen on or `0.0.0.0` to listen on all adapters. For example:

```
run.sh -b 0.0.0.0
```

   The following warning message will appear in the log but can be ignored, since Alfresco disables the faces RI with a special parameter in web.xml:

   [STDOUT] 16:59:43,814 ERROR [shared_impl.config.MyfacesConfig] Both MyFaces and the RI are on your classpath. Please make sure to use only one of the two JSF-implementations.

8. Start the Alfresco server.

Configuring JBoss logging for Alfresco using Simple Logging Facade for Java (SLF4J)

This section describes how to configure JBoss logging for Alfresco using SLF4J.

JBOSS 5.1 uses Simple Logging Facade for Java (SLF4J) as the default logging library. Alfresco uses both log4j and slf4 and by default all the Alfresco logs go to `alfresco.log`. Configure JBoss logging for Alfresco using SLF4J if you want logs for all your applications to go to `server.log` and if you also use an external tool to analyze the generated logs. Use this configuration to ensure that the logs are formatted properly.

1. Download the appropriate version of the `slf4j-x.x.x.zip` file or `slf4j-x.x.x.tar.gz` file from SLF4J.
2. Remove the following JAR files from `<JBOSS_HOME>/server/default/deploy/alfresco.war/WEB-INF/lib/`.
   - `commons-logging.jar`
   - `log4j.jar`
   - `log4j-over-slf4j.jar` (if it is present)
3. Ensure that the following JAR files exist. Any missing files can be found in the archive you downloaded in step 1.
   - `slf4j-api.jar`
   - `slf4j-log4j12.jar`
   - `jcl-over-slf4j.jar`
4. Repeat steps 2 and 3 for the `share.war` file.

5. Configure your log level in the following file `<JBOSS_HOME>/server/default/conf/jboss-log4j.xml`.

   For more information, refer to Configuring JBoss for Alfresco.

6. Start the Alfresco server.

   You may see the following errors during the startup:

   ```
   2011-10-12 10:47:21,505 ERROR [STDERR] (main) SLF4J: Class path contains
    multiple SLF4J bindings.
   2011-10-12 10:47:21,505 ERROR [STDERR] (main) SLF4J: Found binding in
    [vfszip:/usr/local/jeap51/jboss-
   as/common/lib/slf4j-jboss-logging.jar/org/slf4j/impl/
   StaticLoggerBinder.class]
   2011-10-12 10:47:21,505 ERROR [STDERR] (main) SLF4J: Found binding in
    [vfszip:/usr/local/jeap51/jboss
   -as/server/W51J51I1/deploy/alfresco.war/WEB-INF/lib/slf4j-
   log4j12-1.5.11.jar/org/slf4j/impl/StaticLoggerBinder.class]
   2011-10-12 10:47:21,505 ERROR [STDERR] (main) SLF4J: See http://
   www.slf4j.org/codes.html#multiple_bindings for an explanation.
   ```

   To avoid these errors, remove the `<JBOSS_HOME>/common/lib/slf4j-jboss-logging.jar` file.

Configuring Solr with JBoss running on Alfresco

This section describes how to configure Solr to communicate with Alfresco deployed on JBoss. The steps describe how to allow Solr to communicate with Alfresco deployed on JBoss 5.1 EAP.

Solr must be deployed on a separate Tomcat instance.

- Configure Solr using the following instructions: Configuring Solr.

Ensure that Alfresco is installed on JBoss using the instructions described in the section Installing Alfresco on JBoss.

These steps assume that you know the path of the JBoss directory, which is represented as `<JBOSS_HOME>`.

1. Create a file called `tomcat-users.xml` in the `<JBOSS_HOME>/server/default/conf` directory.

2. Enter the following content in the `tomcat-users.xml` file:

   ```
   <?xml version='1.0' encoding='utf-8'?>
   <!--
     Licensed to the Apache Software Foundation (ASF) under one or more
     contributor license agreements.  See the NOTICE file distributed with
     this work for additional information regarding copyright ownership.
     The ASF licenses this file to You under the Apache License, Version 2.0
     (the "License"); you may not use this file except in compliance with
     the License.  You may obtain a copy of the License at

         http://www.apache.org/licenses/LICENSE-2.0

     Unless required by applicable law or agreed to in writing, software
     distributed under the License is distributed on an "AS IS" BASIS,
     WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
    implied.
     See the License for the specific language governing permissions and
     limitations under the License.
   -->
   <tomcat-users>
     <user username="CN=Alfresco Repository Client, OU=Unknown, O=Alfresco
    Software Ltd., L=Maidenhead, ST=UK, C=GB" roles="repoclient"
    password="null"/>
   </tomcat-users>
   ```

3. Create a folder called `<JBOSS_HOME>/server/default/keystore` and then copy all of the files from`<alfresco.war>/WEB-INF/classes/alfresco/keystore` to the new folder.
4. Configure the SSL connector by adding the following to the `<JBOSS_HOME>/server/default/deploy/jbossweb.sar/server.xml` file (should be a child of the `<Service>` tag):

   ```
      <Connector port="8443"
           protocol="HTTP/1.1" SSLEnabled="true" maxThreads="150"
    scheme="https"
           keystoreFile="<JBOSS_HOME>/server/default/keystore/ssl.keystore"
           keystorePass="kT9X6oe68t" keystoreType="JCEKS" secure="true"
           connectionTimeout="240000" clientAuth="false" sslProtocol="TLS"
           truststoreFile="<JBOSS_HOME>/server/default/keystore/
   ssl.truststore"
           truststorePass="kT9X6oe68t" truststoreType="JCEKS"
           address="${jboss.bind.address}"/>
   ```

5. Configure the JBoss realm by adding the following to the `<JBOSS_HOME>/server/default/deploy/jbossweb.sar/server.xml` file (should be a child of the `<Host>` tag):

   ```
   <Realm className="org.apache.catalina.realm.MemoryRealm"
          pathname="<JBOSS_HOME>/server/default/conf/tomcat-users.xml" />
   ```

6. Edit the `<JBOSS_HOME>/server/default/conf/alfresco-global.properties` file by adding following properties:

   ```
   dir.keystore=<JBOSS_HOME>/server/default/keystore
   index.subsystem.name=solr
   solr.host=<host_of_tomcat_inctance_where_solr_is_running>
   solr.port=8080
   solr.port.ssl=8443
   ```

7. Start the Alfresco server.

You may see a message on the JBoss console similar to the following:

```
12:23:15,713 WARN  [JSSESocketFactory] SSL renegotiation is disabled, closing
 connection
```

You may find that Solr search and/or the Solr tracking is not working.

In this situation, use the following steps:

- Add the `allowUnsafeLegacyRenegotiation="true"` option to the JBoss SSL connector.
- Add the `-Dsun.security.ssl.allowUnsafeRenegotiation=true` option to `JAVA_OPTS`.

Installing Alfresco on WebLogic

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

Before you start:

- Install OpenOffice and ensure that the `ooo.exe` property is set in the `alfresco-global.properties` file
- Create an `alfresco` database and user with appropriate permissions
- Install WebLogic 10.3.3 (11g rel 1) without creating any domains or servers

Certain components of Alfresco require access to the EAR file contents as files. These instructions require that you expand the `.ear` into exploded format, as described in the WebLogic documentation. The Alfresco WebLogic deployment solution makes use of a Filter Classloader, configured in the `weblogic-application.xml` file, to ensure that the unmodified contents of the Alfresco web module will run in WebLogic.

1. Browse to the Alfresco Support Portal.

   http://support.alfresco.com

2. Download and extract the `alfresco-enterprise-ear-4.1.5.zip` file.
3. Obtain the license (`.lic`) file.
4. Create a directory in the WebLogic user's home directory to host the exploded EAR file and copy the `alfresco-enterprise-4.1.5.ear` file to that directory.
5. Run the following commands in the new directory to explode the EAR file:
   a. `mkdir alfresco`
   b. `cd alfresco`
   c. `jar xvf ../alfresco-enterprise-4.1.5.ear`
   d. `mv alfresco.war alfresco.war.tmp`
   e. `mv share.war share.war.tmp`
   f. `mkdir alfresco.war`
   g. `mkdir share.war`
   h. `cd alfresco.war`
   i. `jar xvf ../alfresco.war.tmp`
   j. `cd ../share.war`
   k. `jar xvf ../share.war.tmp`
6. Open the WebLogic Configuration Wizard.

   For example, on Unix, use the following command to create a new domain, `alf_domain`:

   ```
   <Weblogic_HOME>/common/bin/config.sh
   ```
7. Create a directory for the license file.

   For example, in Linux, use the following command:

   ```
   mkdir -p <Weblogic_HOME>/user_projects/domains/alf_domain/alfresco/
   extension/license
   ```
8. Move the license `.lic` file into the `license` directory.
9. 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 global properties configuration.
10. Add the following line to the `alfresco-global.properties` file.

    ```
    db.pool.statements.enable=false
    ```

    This property setting is required to make the DBCP connection pool work on WebLogic.
11. Configure the Oracle JDBC OCI driver.
    a. Download the appropriate Oracle Instant Client package for your operating system from Oracle Database Instant Client.
    b. Unzip the package to a local directory.

       For example: `/data/instantclient_11_2`.
    c. Optionally, delete the previously used JDBC driver (for example, `ojdbc6.jar`) from WebLogic.
    d. Add the following line to the `alfresco-global.properties` file.

       ```
       db.url=jdbc:oracle:oci:@${db.host}:1521:${db.name}
       ```
    e. Locate the `setDomainEnv.sh` file. This file is located in the `<Weblogic_HOME>/user_projects/domains/<Domain_name>/bin directory.`

       For example: `/opt/weblogic/user_projects/domains/alf_domain/bin`.

f. Add the following lines to the `setDomainEnv.sh` file:

```
LD_LIBRARY_PATH="/data/instantclient_11_2"
export LD_LIBRARY_PATH
EXT_PRE_CLASSPATH=$LD_LIBRARY_PATH/ojdbc6.jar
export EXT_PRE_CLASSPATH
```

12. In the `setDomainEnv.sh` file, edit all of the lines prefixed with `MEM_MAX_PERM_SIZE` to increase the PermGen space.

```
MEM_MAX_PERM_SIZE="-XX:MaxPermSize=256m"
MEM_MAX_PERM_SIZE_64BIT="-XX:MaxPermSize=512m"
MEM_MAX_PERM_SIZE_32BIT="-XX:MaxPermSize=256m"
```

This setting may need to be increased further in accordance with the number of deployed applications. You may see different combinations of these lines, depending on whether you have installed on a 64 or 32-bit platform.

13. In the `setDomainEnv.sh` file, set the heap size parameter appropriately. The following setting is a recommendation:

```
WLS_MEM_ARGS="-Xmx2048m"
```

14. After the JAVA_OPTIONS parameter, to disable servicing of Platform MbeanServer with the WLS security infrastructure (in the `setDomainEnv.sh` file), use the `-Dweblogic.disableMBeanAuthorization` system property:

```
JAVA_OPTIONS=”${JAVA_OPTIONS} -Dweblogic.disableMBeanAuthorization=true”
```

This modification will allow Alfresco MBeans to co-exist with WebLogic MBeans.

15. The following extra edit is required in the `setDomainEnv.sh` file, to ensure that `alf_domain` is in the global classpath.

```
WL_HOME="/opt/weblogic/wlserver_10.3"
export WL_HOME

PRE_CLASSPATH="/opt/weblogic/user_projects/domains/alf_domain"
export PRE_CLASSPATH
```

16. Edit the `<Weblogic Home>/user_projects/domains/alf_domain/config/config.xml` file and add the following before the end of the `</security-configuration>` section:

```
<enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-credentials>
```

17. Open the `<Weblogic Home>/wlserver_10.3/common/nodemanager/nodemanager.properties` file, and then edit the settings so that the PermGen settings are passed on to the Alfresco server by the node manager.

```
StartScriptEnabled=true
```

18. Start the domain admin server.

For example:

```
<Weblogic_HOME>/user_projects/domains/alf_domain/startWebLogic.sh
```

19. Open a web browser and log in to the admin server (for example, at `http://localhost:7001/console`) with the credentials that you specified while installing Weblogic.

20. 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.

21. To enable Alfresco JMX functionality with Platform MBean server:
    a. In the left pane of Administration Console, click the domain name link, for example `alf_domain`.
    b. In the **Settings for alf_domain** section, select the **Configuration** tab, and then the **General** tab.
    c. On the **General** tab, expand the **Advanced** options group.
    d. Enable the **Platform MBean Server Enabled** option.
    e. Ensure that the **Platform MBean Server Used** option is enabled.
    f. Click **Save** on the **General** tab.
    g. Restart the domain admin server.
22. In the left pane of the Administration Console, click **Deployments**.
23. In the right pane, click **Install**.
24. In the **Install Application Assistant**, locate and select the directory of your exploded EAR file.

    Use the location that you created in step 5. It should contain the `alfresco.war` and `share.war`.
25. Click **Next**.
26. Check **Install this deployment as an application** radio button, and click **Next**.
27. Click **Finish**.
28. Click **Activate Changes**.
29. Using the **Change Center** panel, restart `AlfrescoServer`.
30. 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 >
```

Configuring Solr with Alfresco running on WebLogic

These instructions describe how to configure Solr to communicate with Alfresco deployed on WebLogic 11g Rel1 (10.3.5).

Solr must be deployed on a separate Tomcat instance.

- Configure Solr using the following instructions: Configuring Solr.

The SSL certificate provided with your Alfresco installation will not work on WebLogic. You need to generate a new SSL certificate for Solr to work correctly. For more information, see the instructions in the Generating Secure Keys for Solr Communication topic.

Ensure that Alfresco is installed on WebLogic using the instructions described in the section Installing Alfresco on WebLogic.

1. Edit the `<Weblogic_HOME>/user_projects/domains/alf_domain/alfresco-global.properties` file, and add the following properties:

   ```
   dir.keystore=<Weblogic_HOME>/user_projects/domains/alf_domain/keystore
   index.subsystem.name=solr
   solr.host=<SOLR_HOST>
   solr.port=8080
   solr.port.ssl=8443
   ```
2. Create and populate a keystore directory for the Alfresco and Solr servers.
   1. Create a folder called `<Weblogic_HOME>/user_projects/domains/alf_domain/keystore`.

      At this stage, the keystore directory will just be a template, containing standard keys that are incompatible with Weblogic.
   2. Copy all the files from `<alfresco.war>/WEB-INF/classes/alfresco/keystore` to this new folder.

      To secure the installation, you must follow the steps to generate new keys as explained in the Generating Secure Keys for Solr Communication section.
3. Open the WebLogic Admin Console:
   1. Go to **Environment – Servers – AlfrescoServer – Configuration – General**.
   2. Select the **SSL Listen Port Enabled** checkbox and then enter **8443** in the **SSL Listen Port** field.
   3. Click **Save**.
   4. On the **Keystores** tab, click **Change** and then select the **Custom Identity and Custom Trust** value in drop down menu.
   5. Click **Save**.
   6. In the **Identity** section, enter following parameter values:

      ```
      Custom Identity Keystore:
      <Weblogic_HOME>/user_projects/domains/alf_domain/keystore/ssl.keystore
      Custom Identity Keystore Type:   JCEKS
      Custom Identity Keystore Passphrase:   kT9X6oe68t
      Confirm Custom Identity Keystore Passphrase:   kT9X6oe68t
      ```
   7. In the **Trust** section provide following parameters:

      ```
      Custom Trust Keystore: <Weblogic_HOME>/user_projects/domains/
      alf_domain/keystore/ssl.truststore
      Custom Trust Keystore Type:  JCEKS
      Custom Trust Keystore Passphrase:   kT9X6oe68t
      Confirm Custom Trust Keystore Passphrase:   kT9X6oe68t
      ```
   8. Click **Save**.

i. Select the **SSL** tab and then enter the following fields:

```
Private Key Alias:   ssl.repo
Private Key Passphrase:   kT9X6oe68t
Confirm Private Key Passphrase:  kT9X6oe68t
```

j. Click **Save**.

k. Expand the Advanced link and then enter the following fields:

```
Two Way Client Cert Behavior: Client Certs Requested But Not Enforced
```

l. Click **Save**.

4. Test that Alfresco can now be accessed over SSL.

   For example, enter `https://localhost:8443/alfresco`.

5. In the WebLogic Admin Console, go to **Security Realms – myrealm – Providers – Authentication – DefaultIdentityAsserter**.

   a. In Available Types: select X.509 and move it to the Chosen: list.

   b. Click **Save**.

   c. Select the **Provider Specific** tab and fill following parameters as below:

   ```
   Default User Name Mapper Attribute Delimiter: , (Comma)
   Default User Name Mapper Attribute Type: CN
   Use Default User Name Mapper: true (check the checkbox).
   ```

   d. Click **Save**.

6. Restart AdminServer and AlfrescoServer.

7. In the WebLogic Admin Console, go to **Security Realms – myrealm – Users and Groups - Users**.

8. Click **New**.

9. In Create a New User page fill following parameters as below:

   ```
   Name:   Alfresco Repository
   Client     Password:   kT9X6oe68t
   Confirm Password:    kT9X6oe68t
   ```

10. Click **OK**.

11. To complete the installation, it is necessary to secure the two-way communication between Alfresco and Solr by generating your own keys. For details, see the Generating Secure Keys for Solr Communication topic.

12. Restart AlfrescoServer.

Enabling publishing to YouTube with Alfresco (Lucene enabled) deployed on WebLogic

This section describes how to enable publishing to YouTube for Alfresco deployed on WebLogic and using the Lucene search engine.

To publish on YouTube using Alfresco deployed on Weblogic and configured with Lucene, you need to disable hostname verification. To disable this verification:

1. Open the Weblogic Administration Console.
2. In the **Change Center** of the Administration Console, click **Lock & Edit**.
3. In the left pane of the Administration Console, expand **Environment**.
4. Select **Servers**.
5. Click the name of the server for which you want to disable host name verification.
6. Navigate to **Configuration** > **SSL**.

7. Click **Advanced** at the bottom of the page.
8. Set the **Hostname Verification** field to **None**.
9. In the **Change Center** of the Administration Console, click **Activate Changes**.
10. Click **Save**.
11. Restart your server.

Enabling Google Docs with WebLogic

This section describes how to enable Google Docs for an instance of Alfresco running on WebLogic.

The steps required to complete this task vary depending upon the embedded search engine: Solr or Lucene.

Enabling Google Docs with Alfresco (Lucene enabled) deployed on WebLogic

This section describes how to enable Google Docs for Alfresco deployed within WebLogic and using the Lucene search engine.

To enable Google Docs for Alfresco running on WebLogic and using Lucene as the search engine, you need to add the Google certificate manually using the Keytool Java utility.

1. To download the certificate:

   a. Download OpenSSL (for Windows) from OpenSSL.

      OpenSSL is pre-installed in the Linux and OS operating systems. To verify that you have OpenSSL installed, run the following command: `openssl version`.

   b. Run the following command: `openssl s_client -connect docs.google.com:443 -showcerts`.

      The PEM-format certificate outputs are displayed.

   c. From the output displayed, copy the certificate for the wildcard *.google.com domain to your clipboard including the `----BEGIN CERTIFICATE----` and `----END CERTIFICATE----` parts.

      Several certificates are displayed. The certificate you need to copy is the first one that is displayed and has the value `CN=*.google.com`.

   d. Paste the PEM-format certificate text into a new file.

      For example, `google.com.pem`.

   e. Convert the PEM-format certificate into DER format using OpenSSL.

      For example, `openssl x509 -in google.com.pem -out google.com.der -outform DER`.

2. Add the `google.com.der` certificate to your truststore using the Keytool Java utility.

   ```
   keytool -import -file google.com.der -keystore ssl.truststore -storetype
    JCEKS -alias google.com
   ```

3. Restart the server.

Enabling Google Docs with Alfresco (Solr enabled) deployed on WebLogic

This section describes how to enable Google Docs with Alfresco deployed on WebLogic.

To enable Google Docs with WebLogic, disable host name verification in the WebLogic Server Administration Console.

1. In the Change Center, click **Lock and Edit**.

   For more instructions, see Use the Change Center.

2. Expand **Environment** and select **Servers**.

3. Click the name of the server for which you want to disable host name verification.
4. Select **Configuration > SSL**, and click **Advanced** at the bottom of the page.
5. Set the **Hostname Verification** field to **None**.

   Oracle recommends leaving host name verification on in production environments.
6. Click **Save**.
7. Restart your application server.

Installing Alfresco on WebSphere

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

See the Support Portal for the currently required Fix Pack level. It is important that both the application server and JDK fix pack components are applied.

Alfresco does not support CIFS Kerberos authentication on WebSphere. This is because Alfresco relies on Sun JDK internal classes.

1. Download the Enterprise EAR file `alfresco-enterprise-ear-4.1.5.zip` from the Support Portal and extract it to an empty directory.

   This embeds Alfresco Explorer and Share, plus the necessary WebSphere configuration to use the myfaces1_1 shared library with parent-last loading order.
2. Create a Myfaces v1.1 shared library.

   Because neither of the versions of JSF that ship with WebSphere 7 are compatible with Alfresco, you must define a new isolated shared library in WebSphere that contains a compatible implementation. This is documented in the Configuring JavaServer Faces implementation section of the WebSphere 7 manual. The Alfresco Enterprise `.ear` file embeds an appropriate shared library definition in `META-INF/ibmconfig`, so it is only necessary to prepare WebSphere.

   Copy and extract the `myfaces1_1-websphere-shared-lib-version.zip` file to the root WebSphere installation directory. This creates a `myfaces1_1` directory containing all the `.jars` required by the `myfaces1_1` shared library on WebSphere. For example, on Windows:

   ```
   cd /d "C:\Program Files\IBM\WebSphere\AppServer"
   java\bin\jar xvf myfaces1_1-websphere-shared-lib.zip
   ```
3. Configure Share to point to the WebSphere default HTTP port 9080 (or another number that you wish to specify).

   a. Locate the `/web-server/classpath/alfresco/web-extension/share-config-custom.xml.sample` file from the extracted `alfresco-enterprise-ear-4.1.5.zip` file.

   b. Copy the `share-config-custom.xml.sample` file to `$WAS_INSTALL_ROOT/lib/alfresco/web-extension/share-config-custom.xml` (For example, `C:\Program Files\IBM\WebSphere\AppServer\lib\alfresco\web-extension\share-config-custom.xml`).

   c. Uncomment this section by removing the begin comment `<--` and end comment `-->` lines surrounding this section.

   ```
    <config evaluator="string-compare" condition="Remote">
        <remote>
           <endpoint>
              <id>alfresco-noauth</id>
              <name>Alfresco - unauthenticated access</name>
              <description>Access to Alfresco Repository WebScripts that
   do not require authentication</description>
   ```

```
            <connector-id>alfresco</connector-id>
            <endpoint-url>http://localhost:8080/alfresco/s</endpoint-
url>
            <identity>none</identity>
         </endpoint>

         <endpoint>
            <id>alfresco</id>
            <name>Alfresco - user access</name>
            <description>Access to Alfresco Repository WebScripts that
 require user authentication</description>
            <connector-id>alfresco</connector-id>
            <endpoint-url>http://localhost:8080/alfresco/s</endpoint-
url>
            <identity>user</identity>
         </endpoint>

         <endpoint>
            <id>alfresco-feed</id>
            <name>Alfresco Feed</name>
            <description>Alfresco Feed - supports basic HTTP
 authentication via the EndPointProxyServlet</description>
            <connector-id>http</connector-id>
            <endpoint-url>http://localhost:8080/alfresco/s</endpoint-
url>
            <basic-auth>true</basic-auth>
            <identity>user</identity>
         </endpoint>
      </remote>
   </config>
```

d. Edit the file, replacing all instances of 8080 with 9080 (or the port number that you specify) and all instances of `yourserver` with localhost (or a different host running Alfresco).

e. In certain environments, an HTTP request originating from Flash cannot be authenticated using an existing session. For these cases, it is useful to disable the Flash-based uploader for Share Document Libraries.

   To disable the Flash uploader, add the following lines to the Document Library config section:

```
<!-- Document Library config section -->
   <config evaluator="string-compare" condition="DocumentLibrary"
 replace="true">
      <!--
         File upload configuration
      -->
      <file-upload>
         <adobe-flash-enabled>false</adobe-flash-enabled>
      </file-upload>
   </config>
```

f. Save the file.

4. Install a license.

   If you have been issued with a `.lic` license file for this version of Alfresco, copy it to a `$WAS_INSTALL_ROOT/lib/alfresco/extension/license` directory (for example, `C:\Program Files\IBM\WebSphere\AppServer\lib\alfresco\extension\license\mylicense.lic`).

5. (Optional) Enable WCM.

   You need the WCM bootstrap context on the class path. Obtain the `wcm-bootstrap-context.xml` file and copy it to the `$WAS_INSTALL_ROOT/lib/alfresco/extension` directory (for example, `C:\Program Files\IBM\WebSphere\AppServer\lib\alfresco\extension`).

6. Define the environment information using the extension classpath mechanism and the `alfresco-global.properties` file.
    a. Locate the `/web-server/classpath/alfresco-global.properties.sample` file from the extracted `alfresco-enterprise-ear-4.1.5.zip` file.
    b. Copy the `alfresco-global.properties.sample` file to `$WAS_INSTALL_ROOT/lib`, removing the `.sample` extension.

       For example, `C:\Program Files\IBM\WebSphere\AppServer\lib\alfresco-global.properties`.
    c. Disable the `mbean` server lookup by adding the following property `mbean.server.locateExistingServerIfPossible=false`.
    d. Uncomment and edit the lines appropriate for your database type.
7. Copy the JDBC driver jar to the `${WAS_INSTALL_ROOT}/lib` directory.

   For example, `C:\ProgramFiles\IBM\WebSphere\AppServer\lib`.
8. Install the EAR file.
    a. Log on to the WebSphere Administrative console.

       For example, http://localhost:9060/ibm/console/.
    b. Navigate to **Application servers > server1 > Process definition > Java Virtual Machine**, and then set the Maximum heap size to 2048 MB.
    c. Navigate to **Application servers > server1 > Container Settings >Web Container Settings >Web container transport chains**.
        - Click **HttpQueueInboundDefault (where port is 9080)** .
        - Click **HTTP inbound channel (HTTP_2) Custom properties**.
        - Create a new property with the name `CookiesConfigureNoCache` and set the value to false.
    d. Navigate to **Applications > New Application > New Enterprise Application**.
    e. Browse to `alfresco-enterprise-4.1.5.ear` on the local file system, and then click **Next**.
    f. Select **Detailed - Show all installation options and parameters**, and then click **Next**.
    g. Jump to the **Map resource references to resources** step, which is highlighted with a (**+**).
    h. Under **Target Resource JNDI Name**, type `jdbc/dummy`, and then click **Next**.

       This step is recommended to force Alfresco to use its built in DBCP connection pool rather than a WebSphere data source.
    i. Jump to the **Map environment entries for Web modules** step.
    j. For `properties/dir.root`, specify an absolute file system path where you would like Alfresco file data to be stored.

       For example, `C:\alf_data`.
    k. Leave the Hibernate properties blank, unless you want to override the default behavior, where they will be auto-detected.
    l. Click **Next** and **Finish**.
    m. Save your profile changes to the master repository.
    n. Restart the WebSphere server.

       Alfresco starts with the WebSphere server.

9. Remove the SQL warning messages from log file.

   WebSphere shows warnings in the log file, similar to the following:

   ```
   [12/7/10 17:24:42:206 EET] 0000003a JDBCException W
    org.hibernate.util.JDBCExceptionReporter logWarnings SQL Warning: 4474,
    SQLState: 01000
   [12/7/10 17:24:42:208 EET] 0000003a JDBCException W
    org.hibernate.util.JDBCExceptionReporter logWarnings [jcc][t4][10217]
   [10310][4.8.87]
   Connection read-only mode is not enforceable after the connection has
    been established.
   To enforce a read only connection, set the read-only data source or
    connection property. ERRORCODE=4474, SQLSTATE=01000
   ```

   The current driver implementation will display these warnings, however, they have no impact on the operation of Alfresco. You can either choose to ignore these warnings, or you can configure the logging to stop them displaying.

   a. Open WebSphere Administrative console.
   b. Navigate to **Troubleshooting > Logs** and **trace- Server - Change Log Detail Levels**.
   c. Search for the `org.hibernate.util.*` component.
   d. Set `org.hibernate.util.JDBCExceptionReporter` class logger - Messages and Trace Levels to `severe` or `fatal`.

10. Log in to Alfresco:

    - Alfresco Share at `http://localhost:9080/share`
    - Alfresco Explorer at `http://localhost:9080/alfresco`

Configuring Solr with Alfresco running on WebSphere

These steps describe how to allow Solr to communicate with Alfresco deployed on WebSphere 7.0.

Solr must be deployed on a separate Tomcat instance.

- Configure Solr using the following instructions: Configuring Solr.
- Ensure that Alfresco is installed on WebSphere using the instructions described in the section Installing Alfresco on WebSphere .

Complete the following steps to allow Solr to communicate with Alfresco deployed on WebSphere 7.0.

1. Copy the `sunjce_provider.jar` file within the Oracle JDK directory to the `$WAS_INSTALL_ROOT/java/jre/lib/ext` folder.
2. Edit `$WAS_INSTALL_ROOT/lib/alfresco-global.properties` by adding following properties to it:

   ```
   dir.keystore=<WAS_INSTALL_ROOT>/keystore
   index.subsystem.name=solr
   solr.host=<host_of_tomcat_instance_where_solr_is_running>
   solr.port=8080
   solr.port.ssl=8443
   ```

3. Create a folder called `$WAS_INSTALL_ROOT/keystore` and then copy all of the files from `<alfresco.war>/WEB-INF/classes/alfresco/keystore` to the new folder.
4. In the Administration Console, go to **Security – SSL certificate and key management – Key stores and certificates**, and then select **New**.
5. On the opened page, enter the following parameters, and then select **OK**:

   ```
   Name: AlfrescoKeyStore
   ```

```
Path: <WAS_INSTALL_ROOT>/keystore/ssl.keystore
Password: kT9X6oe68t
Confirm password: kT9X6oe68t
Type: JCEKS
```

6. Save the changes to the master configuration.
7. Create another key store using following parameters:

```
Name: AlfrescoTrustStore
Path: <WAS_INSTALL_ROOT>/keystore/ssl.truststore
Password: kT9X6oe68t
Confirm password: kT9X6oe68t
Type: JCEKS
```

8. Save the changes to the master configuration.
9. Go to **Security – SSL certificate and key management – SSL configurations – NodeDefaultSSLSettings**.
10. Change the **Trust Store Name** parameter value to `AlfrescoTrustStore`.
11. Change the **Key Store Name** parameter value to `AlfrescoKeyStore`.
12. Select **Get certificate aliases**, and then select **OK**.
13. Save the changes to the master configuration.

Configuring Enterprise to Cloud Sync with Alfresco running on WebSphere

This section describe how to configure Enterprise to Cloud Sync for Alfresco deployed on WebSphere 7.0.

- Ensure that Alfresco is installed on WebSphere using the instructions described here Installing Alfresco on WebSphere.
- Ensure that the Alfresco server is not running. Add the cloud-sync license to the correct location. This location is usually `C:\Alfresco\tomcat\shared\classes\alfresco\extension\license`.
- Start alfresco server.

  Complete the following steps to configure Enterprise to Cloud Sync for Alfresco deployed on WebSphere 7.0.

1. Log into the WebSphere Administration console.
2. In the Administration Console, go to **Security – SSL certificate and key management – Configuration settings**, select **Manage endpoint security configurations**.
3. Select the appropriate outbound configuration to get to your server for example, (cell): **SwSt4-AS1-119Node01Cell:(node):SwSt4-AS1-119Node01 management scope**.
4. Under **Related Items**, click **Key stores and certificates AlfrescoTrustStore**.
5. Under **Additional Properties** section, click **Signer certificates**. Select **Retrieve From Port** button.
6. Enter your Enterprise Cloud Sync host name in the **host name** field, for example, `a.alfresco.me`.
7. Enter your Enterprise Cloud Sync SSL port value in the **Port** field for example, `443`.
8. Enter your Enterprise Cloud Sync certificate name in the **Alias** field for example, `a.alfresco.me_cert`.
9. Click **Retrieve Signer Information**.
10. Verify that the certificate information is for a certificate that you can trust.
11. Click **Apply** and **Save**.
12. Synchronize the content again.

Enabling Google Docs with WebSphere

This section describes how to enable Google Docs for an instance of Alfresco running on WebSphere.

The steps required to complete this task vary depending upon the embedded search engine: Solr or Lucene.

If you initially enable Solr on Alfresco and then switch to Lucene, you will already have **AlfrescoTrustStore** in the certificates. To enable Google Docs in this scenario, use the instructions for Google Docs with Alfresco (Solr enabled).

Enabling Google Docs with Alfresco (Lucene enabled) deployed on WebSphere

This section describes how to enable Google Docs for Alfresco deployed within WebSphere and using the Lucene search engine.

1. Download the **Root 1 - Equifax Secure Certificate Authority** certificate from Geotrust.
2. Download the **Root 2 VeriSign Class 3 Public Primary CA** certificate from Download Primary PCA Root Certificates.
3. Export the **www.google.com** certificate from Google account authorization. In Internet Explorer 8, this is done as follows:
   a. Navigate to **Internet Explorer** in the **Start** menu and right click on it.
   b. Select **Run as administrator**.

      You have started Internet Explorer with administrator privileges, this will enable you to export certificates.
   c. Go to Google account authorization.
   d. Click on the padlock icon (to the right of the address bar) and then **View certificates**.
   e. Click **Copy to File** under the **Details** tab.
   f. Select **DER encoded binary X.509 (.cer)** and specify a path.
4. Open the IBM Websphere console in a browser and navigate to **Security > SSL certificate and key management > Key stores and certificates > NodeDefaultTrustStore > Signer certificates** .
5. Add the three certificates you downloaded in the first three steps.
6. Restart the server.

Enabling Google Docs with Alfresco (Solr enabled) deployed on WebSphere

This section describes how to enable Google Docs with Alfresco deployed on WebSphere and using the Solr search engine.

1. Log into the WebSphere administrative console.
2. Navigate to **Security > SSL certificate and key management > Configuration settings > Manage endpoint security configurations** .
3. Select the appropriate outbound configuration to get to your server.

   For example **(cell):SwSt4-AS1-119Node01Cell:(node):SwSt4-AS1-119Node01 management scope**.
4. Under **Related Items**, click **Key stores and certificates** and select the **AlfrescoTrustStore** key store.
5. Under **Additional Properties**, click **Signer certificates and Retrieve From Port** .
6. Click the **Retrieve From Port** button.

7. In the **Host field**, enter `docs.google.com`.
8. In the **Port** field, enter `443`.
9. In the **Alias field**, enter `docs.google.com_cert`.
10. Click **Retrieve Signer Information**.
11. Verify that the certificate information is for a certificate that you can trust.
12. Click **Apply**.
13. Click **Save**.
14. Restart your application server.

Enabling YouTube with WebSphere

This section describes how to enable YouTube for an instance of Alfresco running on WebSphere.

The steps required to complete this task vary depending upon the embedded search engine: Solr or Lucene.

> If you initially enable Solr on Alfresco and then switch to Lucene, you will already have **AlfrescoTrustStore** in the certificates. To enable YouTube in this scenario, use the instructions for YouTube with Alfresco (Solr enabled).

Enabling YouTube with Alfresco (Lucene enabled) deployed on WebSphere

This section describes how to enable YouTube for Alfresco deployed within WebSphere and using the Lucene search engine.

1. Download the **Root 2 VeriSign Class 3 Public Primary CA** certificate from Download Primary PCA Root Certificates.
2. Open the IBM Websphere console in a browser and navigate to **Security > SSL certificate and key management > Key stores and certificates > NodeDefaultTrustStore > Signer certificates** .
3. Click **Add**.
4. Type the alias name.
5. Enter the path to your certificate.
6. Click **Apply**.
7. Click **Save**.
8. Restart the server.

Enabling YouTube with Alfresco (Solr enabled) deployed on WebSphere

This section describes how to enable YouTube with Alfresco deployed on WebSphere.

1. Log into the WebSphere administrative console.
2. Navigate to **Security > SSL certificate and key management > Configuration settings > Manage endpoint security configurations** .
3. Select the appropriate outbound configuration to get to your server.

   For example **(cell):SwSt4-AS1-119Node01Cell:(node):SwSt4-AS1-119Node01 management scope**.
4. Under **Related Items**, click **Key stores and certificates** and select the **AlfrescoTrustStore** key store.
5. Under **Additional Properties**, click **Signer certificates and Retrieve From Port**.
6. Click the **Retrieve From Port** button.

7. In the **Host field**, enter `www.google.com`.
8. In the **Port** field, enter `443`.
9. In the **Alias field**, enter `www.google.com_cert`.
10. Click **Retrieve Signer Information**.
11. Verify that the certificate information is for a certificate that you can trust.
12. Click **Apply**.
13. Click **Save**.
14. Restart your application server.

Enabling SlideShare with WebSphere

This section descibes how to enable Slideshare for an instance of Alfresco running on WebSphere. Using SlideShare, you can upload and share PowerPoint presentations, Word documents, and Adobe PDF Portfolios.

The steps required to complete this task vary depending upon the embedded search engine: Solr or Lucene.

If you initially enable Solr on Alfresco and then switch to Lucene, you will already have **AlfrescoTrustStore** in the certificates. To enable SlideShare in this scenario, use the instructions for SlideShare with Alfresco (Solr enabled).

Enabling SlideShare with Alfresco (Lucene enabled) deployed on WebSphere

This section describes how to enable SlideShare for Alfresco deployed on WebSphere and using the Lucene search engine.

1. Download the **Root 2 VeriSign Class 3 Public Primary CA** certificate from Download Primary PCA Root Certificates.
2. Open the IBM Websphere console in a browser and navigate to **Security > SSL certificate and key management > Key stores and certificates > NodeDefaultTrustStore > Signer certificates** .
3. Click **Add**.
4. Enter the alias name.
5. Enter the path to your certificate and select the DER option.
6. Click **Apply**.
7. Click **Save**.
8. Restart the server.

Enable SlideShare with Alfresco (Solr enabled) deployed on WebSphere

Using SlideShare you can upload and share PowerPoint presentations, Word documents and Adobe PDF Portfolios. This section describes how to enable SlideShare with Alfresco deployed within WebSphere.

1. Log into the WebSphere administrative console.
2. Navigate to **Security > SSL certificate and key management > Configuration settings > Manage endpoint security configurations** .
3. Select the appropriate outbound configuration to get to your server, for example, **(cell):SwSt4-AS1-119Node01Cell:(node):SwSt4-AS1-119Node01 management scope**.
4. Under **Related Items**, click **Key stores and certificates** and select the **AlfrescoTrustStore** key store.

5. Under **Additional Properties**, click **Signer certificates and Retrieve From Port** .
6. Click the **Retrieve From Port** button.
7. In the **Host field**, enter `www.slideshare.net`.
8. In the **Port** field, enter `443`.
9. In the **Alias field**, enter `www.slideshare.net_cert`.
10. Click **Retrieve Signer Information**.
11. Verify that the certificate information is for a certificate that you can trust.
12. Click **Apply**.
13. Click **Save**.
14. Restart your application server.

Installing software required for Alfresco

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.

   All OpenOffice components, including Writer are required. If you install OpenOffice on Ubuntu using the `apt-get` utility rather than the OpenOffice installer, use the following command to install all of the components:

   ```
   sudo apt-get install openoffice.org
   ```
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.

Installing ImageMagick

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

1. Verify that ImageMagick is already installed on your system. Ensure that you have Ghostscripts installed. To download Ghostscripts go to Ghostscripts download. Ensure that 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 ImageMagick downloads 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`.

   c. If you have a custom fonts directory, set the `mg.gslib= property` to `mg.gslib=C:/myfontdir` (where myfontdir, is your custom fonts directory).

   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/`.

   Test that you are able to convert a pdf using the command `convert filename.pdf[0] filename.png`.

Installing Ghostscript

ImageMagick uses Ghostscript to render Postscript and PDF files, as well as formats where a translator to Postscript is available. ImageMagick will also use Ghostscript fonts to support the standard set of Postscript fonts.

The Ghostscript executable file is entirely platform-specific.

- For Windows:
  - Download Ghostscript (32 bit) from the Ghostscript Downloads Page.

    - Browse to the location of your downloaded file and install the application.
    - Update the `img.gslib` property in the `alfresco-global.properties` file as shown below:

      ```
      img.gslib = <GhostScript_installation_dir>/lib
      ```

- For Linux:
    - Download Ghostscript from the Ghostscript Downloads Page.
    - Add the following to the `alfresco-global.properties` file:

      ```
      img.gslib = /usr/share/ghostscript/<version>/lib
      ```

- For Solaris:
    - Download Ghostscript from the Ghostscript Downloads Page.
    - Run the following commands:

      ```
      ./configure
      make
      make install
      ```

      This installs Ghostscript at `/usr/local/`.
    - Update the `img.gslib` property as shown below:

      ```
      img.gslib = /usr/local/share/ghostscript/<version>/lib
      ```

Installing Flash Player

1. Browse to the Adobe website: http://www.adobe.com.
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.

Installing SWF Tools 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 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

This section describes the steps used to install the SWF Tools. 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

Alfresco is localized using the language packs supplied in the default install. Alfresco supports the following language packs: German (de), English (en), Spanish (es), French (fr), Italian (it), Japanese (ja), Dutch (nl), and Russian (ru).

The language in which Alfresco displays switches depending on the locale selected for the browser. Set your browser to your most appropriate locale. This will make sure that the special characters for each language display correctly.

The source-localized files are encoded in ASCII, and the special and accented characters are displayed using escape sequences. The source files have been renamed using the corresponding locale for each language. For example, `site-welcome.properties` is called `sitewelcome_ fr.properties` for the French version.

If you wish to use 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 setup wizards, the default language packs are already installed. If you have installed Alfresco manually, you must install the supported language pack manually.

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/langs`
   - For Explorer, unpack to: `<TOMCAT_HOME>/webapps/alfresco/scripts/tiny_mce/langs`
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. Use the Module Management Tool (MMT) to install and manage AMP files. This section describes how to install an AMP in an Alfresco WAR using the MMT.

The MMT is included in the Alfresco installers, and it is also available as a separate JAR file from the Alfresco WAR file bundle (`alfresco-enterprise-4.1.5.zip`).

For Tomcat, alternatively, run the `apply_amps` command in the Alfresco `\bin` directory, which applies all the AMP files that are located in the `amps` and `amps_share` directories.

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 and configuring Alfresco WCM

Alfresco Web Content Management provides:

- An Open Source, lower cost, and lower risk solution that offers greater control and visibility over current and future costs
- A full ECM suite providing complete control over any content from within a single integrated platform – documents, images, video, audio, etc.
- A complete collaboration environment for creating, approving and publishing content to the web
- A platform built on industry standard technology that is as simple to use as a shared network drive

The Alfresco repository provides an implementation for WCM called Web Quick Start (WQS)

Web Quick Start

Web Quick Start is packaged in four parts:

- An Alfresco Module Package (AMP) that extends the repository to support a generic website model
- An AMP that extends Alfresco Share for editing content for the website, managing the structure of the website, and publishing content using workflow.
- A JAR file that contains a Java API for accessing the website data held in the repository.
- A web application that, when deployed to a servlet container such as Tomcat, delivers a fictional financial news website. The web application is a Spring MVC application constructed using Spring Surf, and communicating with the Alfresco repository using the Java API. As well as dynamically building the website from data held in the repository, Web Quick Start also provides examples of user generated content whereby content is sent from the web application back to the repository.

About Web Quick Start

Using standard development tools developers can quickly deploy the comprehensive content management capabilities of Alfresco to build new and innovative web applications. Developed using the Spring framework with Spring Surf, the Web Quick Start allows developers to easily extend Alfresco WCM to add new features to support the demands of the CMO.

Installing Alfresco and Web Quick Start

When you run the setup wizard, you can choose to install a number of Alfresco components. Web Quick Start is provided as a component but it not selected by default.

Manually installing Web Quick Start

This procedure describes how to copy the AMP files into their appropriate AMP directories and uses the `apply_amps.bat` or `.sh` file to apply them. Alternatively, use the Module Management Tool (MMT) to apply the AMP file.

1. Download the Web Quick Start zip bundle file:

   `alfresco-enterprise-wcmqs-4.1.5.zip`
2. Locate your Alfresco installation directory.
3. Copy the AMP files into the relevant amps directories for Alfresco and Share:
   a. Copy the `alfresco-enterprise-wcmqs-4.1.5.amp` file to the amps directory.
   b. Copy the `alfresco-enterprise-wcmqs-share-4.1.5.amp` file to the amps-share directory.
4. Apply the AMP files using the `apply_amps` command for the Tomcat application server, or, alternatively, use the Module Management Tool (MMT).
5. Copy the website WAR (`wcmqs.war`) into the `webapps` directory of your existing Alfresco installation.

   For example, on Windows with a Tomcat application server, this is `C:\Alfresco\tomcat\webapps`.
6. Copy the Alfresco Web Editor file (`awe.war`) into the `webapps` directory to replace the existing `awe.war` file.
7. Delete the existing alfresco and share directories.
8. Restart the Alfresco server.

Creating the Web Quick Start site

1. Open Share.
2. Click **Create Site**.

   This creates a new collaboration site.
3. Type a name for the site, for example, **Web Quick Start**.
4. Type a URL name for the site, for example **wcmqs**.
5. Click **OK**. The new site displays in your My Sites dashlet.
6. Open the new site.
7. Click **Customize Dashboard**.
8. Click **Add Dashlets**.
9. Drag the Web Quick Start dashlet to your dashboard layout.
10. Click **OK**.

    The Web Quick Start dashlet displays in the site dashboard.

Importing Web Quick Start demo data

1. Click **Import Website Data**.

   Choose the sample content to import: **Government** or **Finance**.

   Both samples are identical in functionality but contain different images and section headings. The samples provide an example of how developers can package and import their own sample site data.

   The system imports the data for the demo website.
2. Refresh the browser running Share.

The Web Quick Start dashlet now displays a link to the **Web Quick Start Help**.

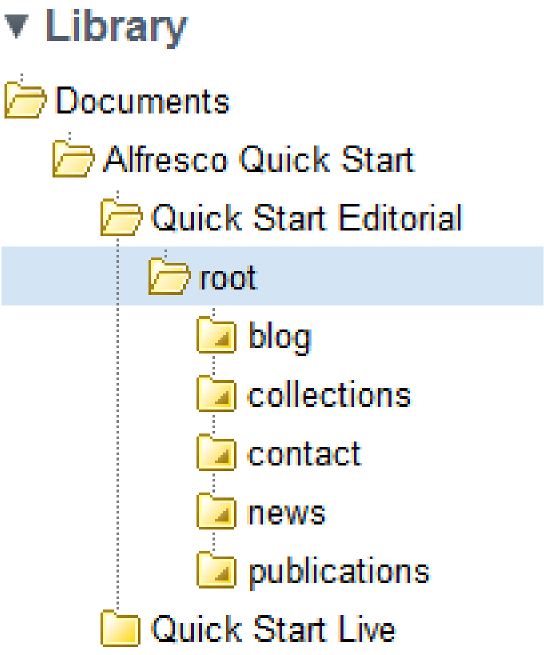
By default, Web Quick Start is configured to be accessed at `localhost` on port 8080. If these settings are relevant for your installation and the `wcmqs.war` is running in the same container as Alfresco, you will now be able to access the Web Quick Start editorial website on http://localhost:8080/wcmqs.

To change the server host name, port, or web application context from the default values, refer to Configuring Web Quick Start.

Configuring Web Quick Start

1. Open the Web Quick Start site.
2. Navigate to the Document Library.

   The default site structure will have the following structure:

   The site structure contains two folders: `Quick Start Editorial` and `Quick Start Live`. These folders represent a separation between the work in progress content, and the finished, reviewed, editorially complete content that is then published to the “Live” environment.

If your web container is running on port 8080 and the web application is running in the same container as Alfresco, the setup is complete and you should be able to access the web site on http://localhost:8080/wcmqs.

Configuring the web application host name, port, and context

The Web Quick Start installation assumes that the web application has been deployed to localhost on port 8080, using the context of wcmqs. This means that the editorial website can be accessed at http://localhost:8080/wcmqs. The "live" website can be accessed as default on http://127.0.0.1:8080/wcmqs.

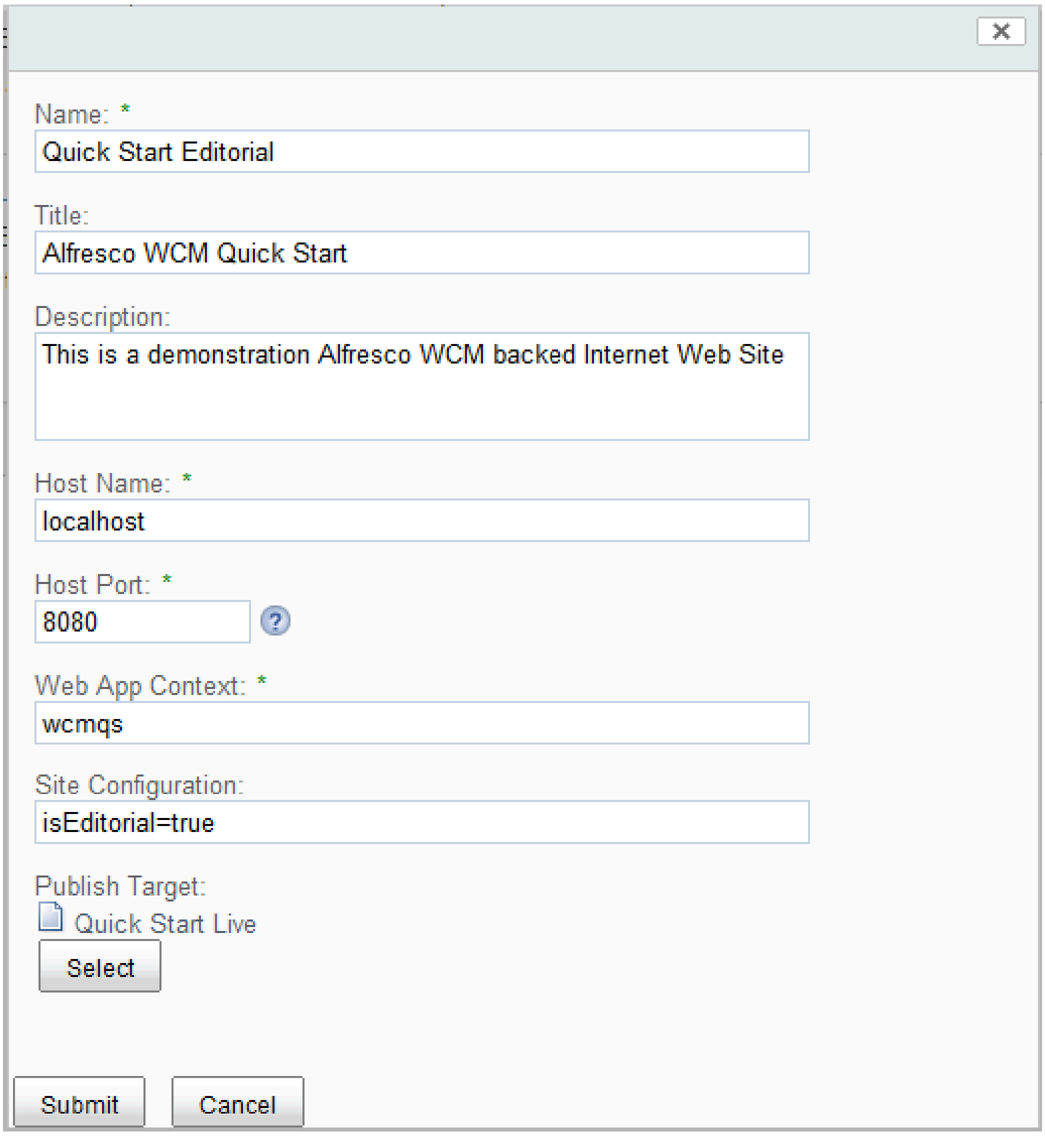
If you are not running the web application on port 8080 or if the web application is deployed to a different container or host, you can configure the site to the required location.

1. In the Web Quick Start site, navigate to the **Document Library**.
2. Click **Edit Metadata** on either the **Quick Start Editorial** folder, or the **Quick Start Live** folder.
3. Configure the **Host Name**, **Port**, and **Web App Context** fields to point to the location your web application (`wcmqs.war`).
4. Click **Submit**.

Disabling AWE on the Live environment

This procedure configures the web application to view the "Live" site structure.

1. Edit the metadata properties on the **Quick Start Live** folder.
2. In the **Site Configuration** field, enter the `isEditorial=true` flag.



Name: * Quick Start Editorial

Title: Alfresco WCM Quick Start

Description: This is a demonstration Alfresco WCM backed Internet Web Site

Host Name: * localhost

Host Port: * 8080

Web App Context: * wcmqs

Site Configuration: isEditorial=true

Publish Target: Quick Start Live

3. Click **Submit**.

The default configuration sets the host address to 127.0.0.1, so if you are running Web Quick Start locally, you can view the editorial environment on http://localhost:8080/wcmqs and the live on http://127.0.0.1:8080/wcmqs.

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.

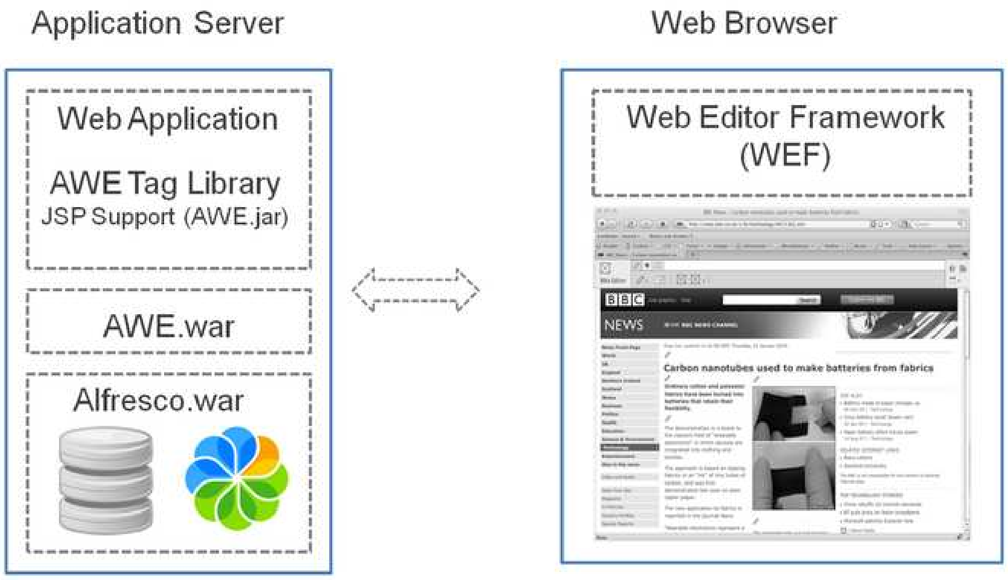
The Alfresco Web Editor uses the Forms Service default template.

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 it identifies the editable content. By default, it assumes that you have JavaScript enabled but it can also run without JavaScript.

Alfresco Web Editor deployment

The simplest way to deploy the Alfresco Web Editor (AWE) 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.

By default the AWE assumes your Alfresco repository is at `http://localhost:8080/alfresco/s/`. If your repository is not located here, you can use custom configuration to tell the AWE where to find your repository. To change the default repository location, add the following XML in the AWE configuration file with your values for **MYSERVER** and **MYPORT**:

```
<alfresco-config>
   <plug-ins>
      <element-readers>
```

```
         <element-reader element-name="remote"
 class="org.springframework.extensions.config.RemoteConfigElementReader" />
      </element-readers>
   </plug-ins>

   <config evaluator="string-compare" condition="Remote">
      <remote>
         <endpoint>
            <id>alfresco</id>
            <name>Alfresco - user access</name>
            <description>Access to Alfresco Repository WebScripts that require
 user authentication</description>
            <connector-id>alfresco</connector-id>
            <endpoint-url>http://MYSERVER:MYPORT/alfresco/s
            </endpoint-url>
            <identity>user</identity>
         </endpoint>
      </remote>
   </config>
</alfresco-config>
```

The AWE configuration file is placed on the classpath named `shared/classes/alfresco/web-extension/awe-config-custom.xml`.

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-enterprise-webeditor-4.1.5.zip`.

1. Shut down your Alfresco server.
2. Download the `alfresco-enterprise-webeditor-4.1.5.zip` file.
3. Deploy the `awe.war` file into the same application server instance as the Alfresco repository.
4. Copy the `alfresco-webeditor-taglib.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.jar`
   d. `alfresco-webeditor-plugin.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.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.

Configuring Alfresco Web Editor

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

This section describes the tag library configuration.

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 identifier 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 is used in the quick edit drop down menu of editable items, as the title of the form edit popup/dialog and the alt text and tool tip text of the edit icon.

   formId
   This is an optional attribute that specifies which form will be used when the marked area is edited.

   nestedMarker
   This is an optional attribute, which defines whether the editable area is nested within another HTML tag that represents the content being edited. If it is 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 (AWE) 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 forms configuration techniques.

When starting up, the AWE looks for a configuration file on the classpath named shared/classes/alfresco/web-extension/awe-config-custom.xml. Place any custom form definitions in this file.

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>
```

Installing and configuring Alfresco DocLib Portlets

This describes how to install and configure Alfresco DocLib Portlets within Liferay. DocLib Portlets provide the rich document management capabilities of the Share Document Library page component within a portal. This feature consists of three portlets for accessing Share sites and the Alfresco repository. In the Share sites portlet, you have access to the sites you are already a member of and all public sites.

Once deployed to Liferay, you will be unable to log in to the native Share application. Therefore, to create and manage Share sites (including managing site membership and accessing other functionality outside the Document Library component), you will require a second Share instance deployed to a separate Tomcat server.

Before proceeding with the installation and configuration, ensure the following:

- The Alfresco repository (alfresco.war) running must be Enterprise 3.3.3 or later.
- Alfresco has been installed using the installers or is running in a Tomcat 6 container.
- Liferay Portal 5.2.3 or Liferay Portal 6.0 GA3 is installed and working.
- The Java version must be a supported version for use with Alfresco: OpenJDK 1.6.0.2, IBM JDK 1.6 (Latest), ir JDK 6 u27 x64

- The administrative user knows the location where Liferay is deployed.
- The required modifications have been completed if you are using Tomcat 6. The modifications must be made for both Alfresco and Liferay if you are using different servers.
- The deployed version of `share.war` in Liferay must be the same as the deployed version of `alfresco.war`.

DocLib Portlets capabilities

The Share DocLib Portlets include most of the functionality available in a standard Share Document Library.

The following Document Library features are not available in DocLib Portlets:

- The **View in Alfresco Explorer** action is not available for folders.
- When viewing the Details page for a folder or content item, the **Share** section does not display.
- User names do not link to the related Profile page.

As the Share header does not display in the DocLib Portlet, the following framework functionality is not available:

Share toolbar:
My Dashboard, My Profile, Sites menu, People Finder, Search, Help, Logout

Other page components:
Wiki, Blog, Calendar, Links, Discussions, Data Lists

Site dashboard:
Invite, Customize Dashboard, Edit Site Details, Customize Site, Leave Site

Refer to the Document Library page component section of the Share User Help for full details of the functionality available.

Configuring Liferay

If you are running Liferay and Alfresco on the same machine, you need to change the port numbers used by the Liferay Tomcat server to prevent conflicts.

1. Open the `<LIFERAY_HOME>/tomcat-6.0.18/conf/server.xml` file.
2. Locate the following lines and update the port numbers as shown:

```
<Server port="8105" shutdown="SHUTDOWN">

    <Connector port="8180" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" URIEncoding="UTF-8" />

    <Connector port="8109" protocol="AJP/1.3" redirectPort="8443"
URIEncoding="UTF-8" />
```

3. Save the file.

Configuring Alfresco

This task describes how to configure Alfresco for Doclib Portlets.

1. Browse to `<ALFRESCO_HOME>/tomcat/shared/classes/alfresco/`.
2. Open the file `alfresco-global.properties`.
3. Add the following:

```
authentication.chain=alfrescoNtlm1:alfrescoNtlm,external1:external
external.authentication.proxyUserName=
```

4. Save the file.

Configuring the Liferay Share web application

This task describes how to deploy the web application.

1. Deploy the Share web application to Liferay by copying the `share.war` file from your Alfresco Tomcat instance into Liferay's `deploy` folder.
2. Locate the `<LIFERAY_HOME>/tomcat-6.0.18/conf/catalina.properties` file and open it in an editor.
3. Locate the `shared.loader` property and set it to the following value:

   ```
   shared.loader=${catalina.home}/shared/classes,${catalina.home}/shared/
   lib/*.jar
   ```
4. Create the directory `<LIFERAY_HOME>/tomcat-6.0.18/shared/classes/alfresco/web-extension/`.
5. Add the following configuration to a new file called `share-config-custom.xml`.

   If your Alfresco Tomcat instance is running on another host or a non-standard port, you must modify the endpoint-url parameters to point to the correct location of your repository. For example:

```

   <!-- example port config used to access remote Alfresco server
 (default is 8080) -->
   <config evaluator="string-compare" condition="Remote">
      <remote>
         <endpoint>
            <id>alfresco-noauth</id>
            <name>Alfresco - unauthenticated access</name>
            <description>Access to Alfresco Repository WebScripts
 that do not require authentication</description>
            <connector-id>alfresco</connector-id>
            <endpoint-url>http://localhost:8080/alfresco/s</
endpoint-url>
            <identity>none</identity>
         </endpoint>

         <endpoint>
            <id>alfresco</id>
            <name>Alfresco - user access</name>
            <description>Access to Alfresco Repository WebScripts
 that require user authentication</description>
            <connector-id>alfresco</connector-id>
            <endpoint-url>http://localhost:8080/alfresco/s</
endpoint-url>
            <identity>user</identity>
         </endpoint>

         <endpoint>
            <id>alfresco-feed</id>
            <name>Alfresco Feed</name>
            <description>Alfresco Feed - supports basic HTTP
 authentication via the EndPointProxyServlet</description>
            <connector-id>http</connector-id>
            <endpoint-url>http://localhost:8080/alfresco/s</
endpoint-url>
            <basic-auth>true</basic-auth>
            <identity>user</identity>
         </endpoint>
      </remote>
   </config>


   <!--
```

```
        Overriding endpoints to reference an Alfresco server with
 external SSO enabled
        NOTE: If utilising a load balancer between web-tier and
 repository cluster, the "sticky
              sessions" feature of your load balancer must be used.
        NOTE: If alfresco server location is not localhost:8080 then
 also combine changes from the
              "example port config" section below.
        *Optional* keystore contains SSL client certificate +
 trusted CAs.
        Used to authenticate share to an external SSO system such as
 CAS
        Remove the keystore section if not required i.e. for NTLM.

        NOTE: For Kerberos SSO rename the "KerberosDisabled"
 condition above to "Kerberos"

        NOTE: For external SSO, switch the endpoint connector to
 "AlfrescoHeader" and set
              the userHeader to the name of the HTTP header that the
 external SSO
              uses to provide the authenticated user name.
   -->

   <config evaluator="string-compare" condition="Remote">
      <remote>
         <keystore>
             <path>alfresco/web-extension/alfresco-system.p12</path>
             <type>pkcs12</type>
             <password>alfresco-system</password>
         </keystore>

         <connector>
            <id>alfrescoCookie</id>
            <name>Alfresco Connector</name>
            <description>Connects to an Alfresco instance using
 cookie-based authentication</description>

 <class>org.alfresco.web.site.servlet.SlingshotAlfrescoConnector</
class>
         </connector>

         <connector>
            <id>alfrescoHeader</id>
            <name>Alfresco Connector</name>
            <description>Connects to an Alfresco instance using
 header and cookie-based authentication</description>

 <class>org.alfresco.web.site.servlet.SlingshotAlfrescoConnector</
class>
            <userHeader>SsoUserHeader</userHeader>
         </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>

</alfresco-config>
```

Creating Liferay users

This task describes how to create the Liferay users.

Ensure that the Alfresco server is started first, and then start up Liferay. Check that no errors are recorded in the application logs.

1. Log in to Liferay as an administrator user.
2. Create a new user account for each of the Alfresco users that you wish to give access to the portal.

   The users must have already been set up in Alfresco with the correct permissions. The screen name of the user in Liferay must match their Alfresco user name.

Adding portlets to Liferay

This task describes how to add the portlets to Liferay.

Ensure that the Alfresco server is started first, and then start up Liferay. Check that no errors are recorded in the application logs.

1. Log in to Liferay.
2. Create a new page and set the layout to one full-sized portlet.
3. In the **Add Application** menu, expand **Alfresco** and select **Share: My Document Libraries**.

   Liferay creates the My Document Libraries portlet.
4. Create another page with the same layout and add the **Share: Repository Browser** portlet.
5. Create another full-sized page and add the **Share: Site Document Library** portlet. A message displays indicating that the portlet must be configured before use. Select the portlet **Preferences** option, and then select the site to which the portlet will be bound.

   ⚠ Each of the three Alfresco Share portlets must be deployed to its own Liferay page. There is no support for deploying more than one portlet to the same page.

Installing and configuring Alfresco XAM Connector

The Alfresco XAM Connector module provides integration between Alfresco and Content Addressable Storage (CAS) systems. XAM is a series of APIs developed to interface with different storage vendor's CAS systems.

CAS systems store and locate files using addresses based on the file’s content, rather than a physical location address. CAS systems are typically used for long-term storage of content that does not require frequent access or where it is stored for regulatory purposes.

When a CAS system stores content, it generates a unique key or hash, which is based on the content. The hash is generated from the content properties, such as the name, date, or content itself. An example hash might be `d8668fbab44d88f8601e625fd88dee84`. This hash is then used as the address of the stored content, and which is then used to retrieve the content. When a request is made to the CAS using this address, it returns the associated content.

The benefits of using CAS systems are:

- Content can be located easily even in large volumes of data
- Content integrity: if stored content has been altered then there is a mismatch between the hash passed as the address and hash computed on the fly

- Avoids redundancy by recognizing that the hash is already present and so does not store it again

The Alfresco XAM Connector module integrates Alfresco with a XAM-enabled Content Access Store (CAS). Currently, this module supports the EMC Centera store.

The module uses a series of properties to control the way that you access the store, and so on. There is also a feature of this module that allows you to set retention policies, such as, preventing content from being deleted for a period of time (for example, retaining invoices for seven years).

The XAM Connector module can be applied to Alfresco Enterprise 3.3.5 or later.

Software prerequisites for XAM Connector module

The Alfresco XAM Connector module supports the EMC Centera.

The XAM Connector must be installed and configured with a completely new Alfresco instance that has never been started. Once an Alfresco server starts, it automatically creates some initial content. The XAM Connector, therefore, will not work with an existing installation of Alfresco.

To use the Alfresco XAM Connector module, you need to ensure that you have the prerequisite software installed on your machine. The software is available from the EMC Community Network. You will need to register and login to the Community Network before you can access the required software.

Download the following software from these sites:

- Centera VIM and XAM: To obtain the Centera VIM and XAM libraries, send a request to the following distribution list, stating that you are an Alfresco customer using the Alfresco XAM Connector: CenteraSDKGurus@emc.com.
- Server details and .pea files: https://community.emc.com/docs/DOC-1038
- XAM tools: https://community.emc.com/docs/DOC-2542

Setting up the Centera test environment

These steps describe how to set up the test environment for Centera to integrate with the Alfresco XAM Connector module.

1. Download and extract Centera VIM and XAM into one of the following appropriate directories:

   - (Linux) `/opt/Centera`
   - (Windows) `C:\prog\centera`

   Create a subdirectory structure of the Centera directory to include the following directories:

   ```
   docs
   include
   lib
   lib32
   lib64
   ```

   The following files are also included:

   ```
   Centera_SDK_XAM_VIM_ReferenceGuide.pdf
   Centera_SDK_XAM_Windows_ReleaseNotes.pdf
   ReadMe.txt
   us2_armTest1.pea
   XAM_Arch.pdf
   XAM_C_API.pdf
   XAM_Java_API.pdf
   ```

2. Move the libraries to the relevant `/lib` directory for your system.

- Choose `/lib32` for 32-bit systems
- Choose `/lib64` for 64-bit systems

3. Download the Centera `us2_armTest1.pea` file.
4. Move the `us2_armTest1.pea` file to the `/opt/Centera` or `C:\prog\centera` directory.
5. Download and unzip the XAM tools to any location.

The structure of the directory will be similar to the following example (for Windows):

```
13/12/2010  16:03    <DIR>          .
13/12/2010  16:03    <DIR>          ..
09/12/2009  12:44         1,095,932 Centera_SDK_XAM_VIM_ReferenceGuide.pdf
09/12/2009  12:44           350,372 Centera_SDK_XAM_Windows_ReleaseNotes.pdf
13/12/2010  15:54    <DIR>          docs
13/12/2010  15:56    <DIR>          include
13/12/2010  15:56    <DIR>          lib
13/12/2010  15:56    <DIR>          lib32
13/12/2010  15:56    <DIR>          lib64
09/12/2009  12:44             2,344 ReadMe.txt
11/10/2010  12:20               294 us2_armTest1.pea
09/12/2009  12:44         1,402,087 XAM_Arch.pdf
09/12/2009  12:44         1,419,797 XAM_C_API.pdf
09/12/2009  12:44           881,682 XAM_Java_API.pdf
               7 File(s)      5,152,508 bytes
               7 Dir(s)  91,181,363,200 bytes free
```

The structure of the C:\progs\centera\lib32 directory is similar to the following example:

```
13/12/2010  15:56    <DIR>          .
13/12/2010  15:56    <DIR>          ..
09/12/2009  12:44           839,680 centera_vim.dll
09/12/2009  12:44           831,488 FPCore.dll
09/12/2009  12:44           450,560 fpos32.dll
09/12/2009  12:44         2,011,136 fpparser.dll
09/12/2009  12:44           184,320 FPStreams.dll
09/12/2009  12:44           438,272 FPUtils.dll
09/12/2009  12:44           192,512 FPXML.dll
09/12/2009  12:44           262,144 pai_module.dll
09/12/2009  12:44           401,408 xam.dll
09/12/2009  12:44            64,114 xam.lib
09/12/2009  12:44            53,248 xam_toolkit.dll
09/12/2009  12:44             2,844 xam_toolkit.lib
              12 File(s)      5,731,726 bytes
               2 Dir(s)  91,162,980,352 bytes free
```

The structure of the C:\progs\centera\lib64 directory is similar to the following example:

```
13/12/2010  15:56    <DIR>          .
13/12/2010  15:56    <DIR>          ..
09/12/2009  12:44           940,032 centera_vim.dll
09/12/2009  12:44         1,149,952 FPCore.dll
09/12/2009  12:44           634,368 fpos64.dll
09/12/2009  12:44         2,959,872 fpparser.dll
09/12/2009  12:44           239,616 FPStreams.dll
09/12/2009  12:44           565,760 FPUtils.dll
09/12/2009  12:44           244,224 FPXML.dll
09/12/2009  12:44           439,296 pai_module.dll
09/12/2009  12:44           390,656 xam.dll
09/12/2009  12:44            61,690 xam.lib
09/12/2009  12:44            11,776 xam_toolkit.dll
09/12/2009  12:44             2,826 xam_toolkit.lib
              12 File(s)      7,640,068 bytes
               2 Dir(s)  91,150,495,744 bytes free
```

Configuring the XAM connection

These steps describe how to configure the XAM connection.

1. Open the `<classpathRoot>/alfresco-global.properties` file.
2. Add the `xam.xri` property.

   For example, `xam.xri=snia xam://centera_vim!128.221.200.60?/opt/centera/us2_armTest1.pea`

   The `xam.xri` property specifies the details of the XAM server, which in this case, is `centera_vim!128.221.200.60?`. The property value also includes the location of the Centera `us2_armTest1.pea` file. For example, `/opt/centera/us2_armTest1.pea` or `C:\prog\centera\us2_armTest1.pea`.
3. There are various additional properties that can be set to control the behavior of the XAM Connector module.

   For example, the retention period for storing content is `xam.archive.retentionPeriodDays=1`.

   The sample `alfresco-global.properties` file supplied in the XAM connector AMP provides example settings and values.
4. Save the `alfresco-global.properties` file.
5. Ensure Java can find the libraries by setting the `PATH` and `LD_LIBRARY_PATH` environment variables.

   For example:

   ```
   export PATH=$PATH:/opt/centera/lib64
   export LD_LIBRARY_PATH=/opt/centera/lib64
   ```

Alfresco XAM Connector module properties

The following properties can be set for the XAM connector module.

Set the Alfresco XAM Connector module properties are set in the `alfresco-global.properties file`.

snia-xam://centera_vim!128.221.200.60?/opt/centera/us2_armTest1.pea xam.xri=
Specifies the full XAM connection string.

xam.archive.storeName=xamArchive
Specifies the name of the XAM store that will be used by the `xam:archive` behavior.

xam.archive.retentionPeriodDays=0
Specifies the number of days to retain a XAM document. Use `0` to ignore; `>0` days to retain.

xam.archive.addLock=true
Specifies whether to add the `cm:lockable` aspect automatically. Set to true to apply a lock when the aspect is added; set to false to not apply a lock

xam.archive.cronExpression=0 0 21 * * ?
Specifies a cron expression for the XAM archive cleaner job.

xam.archive.bindingPropertiesPattern=vnd\\.com\\.alfresco\\..*
Specifies the pattern for all binding properties. Any property (full property name at time of writing) that does not match will be written as non-binding. For example, `vnd\\.com\\.alfresco\\..*` will match all properties prefixed with `vnd.com.alfresco`. Refer to http://download.oracle.com/javase/tutorial/essential/regex/, also http://download.oracle.com/javase/6/docs/api/.

xam.archive.app.vendor= xam.archive.app.name= xam.archive.app.version= xam.archive.app.db=${db.url}
The XAM well-known properties, which will be automatically populated.

xam.archive.globalPropertiesPrefix=vnd.com.alfresco.

xam.archive.globalPropertiesToWrite=xam.archive.app.vendor, xam.archive.app.name, xam.archive.app.version, xam.archive.app.db

The list of global properties to add to the XSet (comma-separated). For example, `${xam.archive.globalPropertiesPrefix}xam.archive.app.vendor`. This can be a list of any value that can be set in the `alfresco-global.properties` file but you should import any required properties using variable replacement to get consistent naming.

xam.archive.contentFieldName=${xam.archive.globalPropertiesPrefix}content

Specifies the name of the property against which to store content.

xam.archive.nodePropertiesPrefix=xam.archive.node.

xam.archive.nodePropertiesToWrite=sys:ref, sys:path, cm:name, cm:created, cm:creator, cm:owner

The list of node properties to add to the XSet (comma-separated, namespace-prefixes). For example, `${xam.archive.globalPropertiesPrefix}${xam.archive.nodePropertiesPrefix}cm:name`. Properties that are not present on the node are ignored. Implicit properties are generated and can be listed, for example, `sys:ref`, `sys:path`.

xam.archive.forceBackgroundStoreMove

Specifies whether to move content to the XAM store immediately or as a background job. The aspect `xam.archivemodel:archivePending` is added to the document, pending the move to the XAM store. Set to false to move the content binaries to XAM as soon as the retention date is set. Set to true to move the content when the scheduled job runs. The default value for this property is false.

Testing the XAM connection

These steps describe how to test the XAM connection.

The shXAM tool is provided within the XAM tools. Use the shXAM tool to connect to the XAM server using the `xam.xri` property that you specified in the `alfresco-global.properties` file.

1. Run the shXAM tool.
2. Run the following command:

   ```
   connect snia-xam://centera_vim!128.221.200.60?/opt/centera/
   us2_armTest1.pea
   ```

   An example of the output is as follows:

   ```
   shXAM>connect snia-xam://centera_vim!128.221.200.60?/opt/centera/
   us2_armTest1.pea

   Connected to an XSystem with XRI: snia-xam://centera_vim!128.221.200.60?/
   opt/centera/us2_armTest1.pea
   ```

Installing the XAM Connector module

These steps describe how to install the XAM Connector to an instance of Alfresco.

The XAM Connector is packaged as an Alfresco Module Package (AMP) file.

1. Browse to the Alfresco Support Portal.

   http://support.alfresco.com
2. Download the `xam-connector-1.6.0-5.amp` file.
3. Use the Module Management Tool (MMT) to install the AMP.

   If your Alfresco installation is running within the Tomcat application server, you can use the `<installLocation>\bin\apply_amps` command to apply all AMP files that are located in the `amps` directory.

4. Restart the Alfresco server.

   You will see the following message in the logs to confirm that the XAM Connector is installed:

   ```
   Starting module org_alfresco_module_xamconnector version 1.0.
   ```

Testing the XAM Connector module

These steps describe how to test the XAM Connector module with Alfresco.

1. Enable `DEBUG` logging for the Alfresco XAM components.

   For example:

   ```
   log4j.logger.org.alfresco.enterprise.repo.content.xam=DEBUG
   log4j.logger.org.alfresco.enterprise.repo.xam=DEBUG
   ```
2. Add the `xam:archived` aspect in the `share-config-custom.xml` file.

   For example:

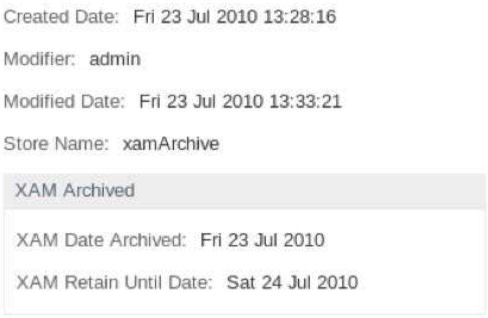
   ```
   <alfresco-config>

      <config evaluator="node-type" condition="cm:content">
         <forms>
            <form>

               <!-- 2 column template -->
               <edit-form />

               <field-visibility>

               <!-- aspect: cm:storeSelector -->
               <show id="cm:storeName" />

               <!-- aspect: xam:archive -->
               <show id="xam:dateArchived" for-mode="view" />
               <show id="xam:retainUntil" for-mode="view" />

               <show id="cm:content" for-mode="view" />

               </field-visibility>
               <appearance>
                  <!-- Store Selector -->
                  <field id="cm:storeName" label="Store Name"
    description="Content Store Name" />
                  <set id="xam-archive" appearance="bordered-panel"
    label="XAM Archived" />
                  <field id="xam:dateArchived" label="XAM Date Archived"
    set="xam-archive" />
                  <field id="xam:retainUntil" label="XAM Retain Until Date"
    set="xam-archive" />

               </appearance>
            </form>
         </forms>
      </config>

   <config evaluator="string-compare" condition="DocumentLibrary">
      <aspects>
            <visible>
               <aspect name="cm:storeSelector" label="Store Selector"/>
               <aspect name="xam:archive" label="XAM Archive" />
            </visible>
            <addable>
               <aspect name="xam:archive" label="XAM Archive" />
            </addable>
      </aspects>
   </config>
   ```

```
</alfresco-config>
```

3. Create a sample document.

   For example, `abc.txt`.

4. Apply the `xam:archived` aspect to the document.

   Created Date: Fri 23 Jul 2010 13:28:16

   Modifier: admin

   Modified Date: Fri 23 Jul 2010 13:33:21

   Store Name: xamArchive

   XAM Archived

   XAM Date Archived: Fri 23 Jul 2010

   XAM Retain Until Date: Sat 24 Jul 2010

5. View the metadata for the document.

   You must also ensure that the new store is the **xamStore** and that the **retainedUntil** date is set.

6. Use the Node Browser to locate the content URL for the node.

   The URL must be a XAM XRI-based URL. The debug output should also show the XUID used.

7. Copy the XAM XRI, and then open the XSet using the shXAM tool.

   For example:

   ```
   shXAM>openxset
    AAAEcwA9f4xCRlE4MU1OM1Q4NzRGZTIySkQ5NVFINjM2RUNHNDE1SkpPRDlCUDFHQkdMUExMQVJMTlNMR

   XSet opened
   ```

   a. Use the following command to check that the base retention has been set:

      `viewFields .xset.retention.base`

   b. Use the following command to check that node properties have been copied over:

      `viewFields xam.archive`

Setting up the XAMContentStore as the primary store

These steps describe how to set up the XAMContentStore to be the primary store for all content. This setup relates to new content and cannot be done retrospectively, unless all content is moved from the file system to XAM.

1. Create a bean called `fileContentStore` that uses the XAMContentStore.
2. Copy the `org_alfresco_module_xamconnector_xamContentStore` bean to a custom context and rename it `fileContentStore`.

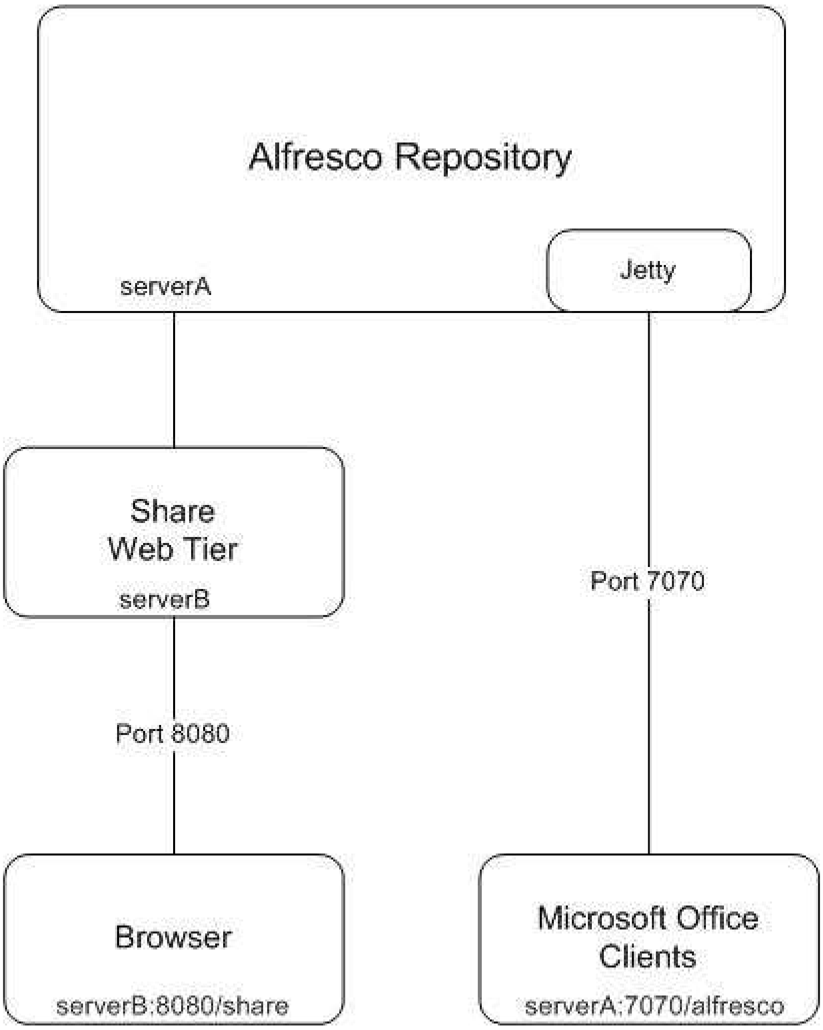
Installing and configuring Microsoft Office SharePoint Protocol Support

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. Browse to the Alfresco Enterprise download area.

2. Download and expand the `alfresco-enterprise-spp-4.1.5.zip` file.
3. Shut down the Alfresco server.
4. Move the `alfresco-enterprise-spp-4.1.5.amp` file to the `amps` directory.

   This file holds the functionality for the SharePoint connector.
5. Apply the SharePoint AMP to the `alfresco.war` file using the Module Management Tool (MMT).

   For Tomcat, alternatively, run the `apply_amps` command in the Alfresco `\bin` directory, which applies all the AMP files that are located in the `amps` and `amps_share` directories.
6. Start the Alfresco server.
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
   ```

Prerequisites for using SharePoint Protocol

The SharePoint Protocol module lets you manage Alfresco content from within Microsoft office. To use SharePoint with Alfresco, you need to install an important update from Microsoft.

Refer to the update "Software Update for Web Folders (KB907306)" or the visit the following link for full details http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=15123.

It is important to install this update, otherwise you will not be able to access the Alfresco content from within Microsoft Word.

Configuring SharePoint Protocol Support

Ensure that you have applied the SharePoint Protocol Support AMP.

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`. The properties have a format of `vti.share.*`.

These properties do not need to be changed as they point to specific pages in Share. However, if, for example, you wish to run the repository and Share on different machine, you may need to modify the first three properties (`vti.server.*`). Set your custom properties and overrides in the `alfresco-global.properties` file.

1. Open the `alfresco-global.properties` file.
2. The following table describes the properties that you can add.

Property	Description
`vti.server.port`	This property configures the port on which the SharePoint server listens. This is the port for the vti embedded server. The default port number is 7070. Use this port in File Open/Save windows and for Document Workspace creation.

`vti.server.external.host=${localname}`
`vti.server.external.port=${vti.server.port}`
	These values are used by Share to generate the **Edit Online** link, which opens the document using the SharePoint module. These parameters are used by `vti-server.get` web script. Share uses this script to determine vti host and port.

`vti.share.siteInBrowser=/page/site/.../dashboard`	If you click the **Open site in browser** link in the **Shared Workspace** panel, the site dashboard will open in a browser. This property generates the browser URL. Dots in this template are replaced with the site `shortname`.

Property	Description
`vti.share.siteSettings=/page/site/.../customise-site`	This property generates the Share URL for **Change Site Settings**.
`vti.share.siteGroupMembership=/page/site/.../site-members`	This property generates the Share URL for the **Site Group Membership** page.
`vti.share.userInformation=/page/user/.../profile`	This property generates the Share URL for the **Edit User Information** page.
`vti.share.documentLibrary=/page/site/.../documentlibrary`	This property is not used.
`vti.share.documentDetails=/page/site/.../document-details`	This property generates the Share URL for the **Modify settings for document version**s page. This link is in the Version History.
`vti.share.calendar=/page/site/.../calendar`	This property is used only for Outlook. The SharePoint Protocol Support module generates the Share URL for Outlook Meeting Workspace and places this link to mail template.

The value for the context path of the Alfresco repository being used is specified using the sysAdmin subsystem property `alfresco.context`. The default is `alfresco`.

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.

  Refer to the knowledge base article 870853 on the Microsoft website to enable the Online Editing functionality.
- To use Online Editing on Windows 7, you must set `BasicAuthLevel` in the registry.

  Basic authentication is disabled on Office 2010 by default. To enable it, create or edit the following registry key: `HKCU\Software\Microsoft\Office\14.0\Common\Internet: BasicAuthLevel=2`.

  Alternatively, use Pass-through or Kerberos authentication.

  It is important to set the `vti.server.external.host` and `vti.server.external.port` properties in the `alfresco-global.properties` file to the externally resolvable host and port name that SharePoint clients will communicate with. These properties default to the host machine's local name and port 7070, respectively. These values are used by Share to generate the **Edit Online** link, which opens the document using the SharePoint module.

Setting up SharePoint Protocol Support to work with Office 2010

Alfresco supports the use of Office 2010 clients and Office 2007/2010 on Windows 7 with Alfresco Enterprise 3.4.0 or later. Some installations of Windows 7 restrict access to Basic HTTP authentication on non-SSL connections.

1. Set the following registry keys.

Each regkey takes the following values:

Options	Description
0	Basic authentication disabled
1	Basic authentication enabled for SSL shares only
2 or greater	Basic authentication enabled for SSL shares and for non-SSL shares

a. Change or create the following registry key and set its value to 2.

```
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WebClient
\Parameters\BasicAuthLevel" (REG_DWORD)
```

b. Change or create the following registry key and set its value to 2:

```
“HKEY_CURRENT_USER\Software\Microsoft\Office\14.0\Common\Internet
\BasicAuthLevel” (REG_DWORD)
```

c. Restart the web client Windows service for the changes to take effect.

2. Ensure that the Microsoft patch for the Web Folders component (KB907306) is installed on your Windows Vista client. This patch contains a fix for a bug in Web Folders for Windows Server 2003 or Windows Vista. To install this patch:

   a. Download the patch KB907306 from the following location: http://www.microsoft.com/downloads/details.aspx?familyid=17C36612-632E-4C04-9382-987622ED1D64&displaylang=en

   b. Double-click the `Webfldrs-KB907306-ENU.exe` file.

      The default installation location is `\Program Files\Common Files\Microsoft Shared\Web Folders`.

   c. Follow the instructions in the setup wizard to complete the installation.

3. If the Office client requests credentials for each operation, it may be related to domain policy and the client settings. For example:

   - **Windows 7 Prompting for Authentication When Accessing SharePoint Documents**
   - **You are prompted to enter your credentials when you access an FQDN site from a computer that is running Windows Vista or Windows 7 and has no proxy configured** http://support.microsoft.com/kb/943280

If you experience problems with the prompts for your credentials, click the small down arrow on the right side of the **Open** button (instead of clicking the **Open** button itself). When you have clicked this arrow, the client will prompt you for your credentials and you can start working with documents from the Alfresco sites.

Setting up sticky sessions with SharePoint Protocol Support

The SharePoint Protocol Support module uses an embedded jetty server (running on port 7070) which receives the client's SharePoint Protocol requests and then communicates with the Alfresco services. When using a clustered environment (for example, two nodes with shared content) using a load balancer between the nodes and the clients, you should set sticky sessions for the SharePoint Protocol Support module.

1. Open the `alfresco-global.properties` file for each cluster node.
2. Set the `vti.server.sessionIdManager.workerName` property for the VTI server. For example:

For Alfresco node 1:

```
vti.server.sessionIdManager.workerName=Alfresco1
```

For Alfresco node 2:

```
vti.server.sessionIdManager.workerName=Alfresco2
```

3. Configure the load balancer stickiness.

   For example, in apache 2.2 using `mod_proxy` and `mod_proxy_balancer`.

```
# map to cluster with session affinity (sticky sessions)
ProxyPass /balancer !
ProxyPass / balancer://my_cluster/ stickysession=VTISESSIONID
 nofailover=On

<Proxy balancer://my_cluster>
    BalancerMember http://yourjetty1:7070 route=Alfresco1
    BalancerMember http://yourjetty2:7070 route=Alfresco2
</Proxy>
```

Setting up SharePoint Protocol Support to work with HTTPS

1. Open the `vti-context.xml` file.

   The location of this file is `<configRoot>\classes\alfresco\module\org.alfresco.module.vti\context`.

2. Configure `SslSocketConnector` for Jetty. Comment out the existing `vtiServerConnector` bean, and uncomment the `<bean id="vtiServerConnector" class="org.mortbay.jetty.security.SslSocketConnector">` bean:

```
    <bean id="vtiServerConnector"
 class="org.mortbay.jetty.security.SslSocketConnector">
        <property name="port">
             <value>${vti.server.port}</value>
        </property>
        <property name="headerBufferSize">
             <value>32768</value>
        </property>
        <property name="maxIdleTime">
             <value>30000</value>
        </property>
        <property name="keystore">
             <value>${vti.server.ssl.keystore}</value>
        </property>
        <property name="keyPassword">
             <value>${vti.server.ssl.password}</value>
        </property>
        <property name="password">
             <value>${vti.server.ssl.password}</value>
        </property>
        <property name="keystoreType">
             <value>JCEKS</value>
        </property>
    </bean>
```

   For more information, refer to http://docs.codehaus.org/display/JETTY/Ssl+Connector+Guide and http://jetty.codehaus.org/jetty/jetty-6/apidocs/org/mortbay/jetty/security/SslSocketConnector.html.

   This example configures HTTPS using the default port 7070, which avoids rewrites in some configuration files and templates.

3. Use the Java `keytool` utility to generate a key pair for the connector:

```
%JAVA_HOME%\bin\keytool.exe -genkeypair -alias alfresco -keystore
 %ALFRESCO_HOME%\alf_data\keystore\vti.ssl.keystore
```

```
-storepass changeit  -keypass changeit
-keyalg RSA -validity 360 -keysize 2048 -storetype  JCEKS
```

4. Browse to the `<extension>` directory.

   For example, for Tomcat 6:

   - (Windows) `C:\Alfresco\tomcat\shared\classes\alfresco\extension`
   - (Linux) `tomcat/shared/classes/alfresco/extension`
5. Open the `custom-vti.properties` file.
6. Add the following properties to the file.

   ```
   vti.server.port=7070
   vti.server.protocol=https
   vti.server.ssl.keystore=%ALFRESCO_HOME%\alf_data\keystore
   \vti.ssl.keystore
   vti.server.ssl.password=changeit
   ```

   Do not change the value for the `vti.server.ssl.password` property. The default value for this property is `changeit`.
7. Save the `custom-vti.properties` file.

The SharePoint Support module will be able to handle requests using HTTPS on 7070 port only.

Installing and configuring the Alfresco Transformation Server

This section describes how to install and configure the Alfresco Transformation Server.

Transformation server overview

The Alfresco Transformation Server is a stable, fast, and scalable solution for high-quality transformations of Microsoft Office documents. It is an enterprise-scale and enterprise quality alternative for OpenOffice.

The server features an open architecture, and it offers the following features:

High quality
The Alfresco Transformation Server uses genuine Microsoft Office software to transform MS Word, Excel, and PowerPoint documents to PDF and SWF. This guarantees the handling of all Office files and pixel-perfect transformations, and it corrects previous layout issues in the Share preview feature.

Scalable
The Alfresco Transformation Server communicates with Alfresco using an HTTP REST API, which means that you can scale up by adding multiple instances of the server and connecting them through a standard HTTP Network Load Balancer.

Stable
If Microsoft Office can open and transform your document, then so can the Alfresco Transformation Server. Robust error handling will take care of corrupt and encrypted documents. A Web Console shows you a detailed report if there is a problem during transformation, allowing you to correct documents.

Fast
The Alfresco Transformation Server is two to three times faster when transforming multi-megabyte Office documents when compared with OpenOffice on the same hardware.

Extensible format support

The Alfresco Transformation Server supports the transformation of MS Office formats. Upcoming versions will support image and video transformations. Contact Alfresco Support if you need support for other formats.

> The transformation server does not support text and html files. However the standard functionality provided by the Alfresco server with OpenOffice installed will still allow preview of these files.

Transformation Server prerequisites

The Alfresco Transformation Server consists of two software modules:

- Standalone Transformation Server
- Alfresco Transformation Client

The Standalone Transformation Server runs on Windows and takes care of the file transformations.

The Alfresco Transformation Client runs as a part of the Alfresco ECM server and takes care of the communication between Alfresco and the Transformation Server.

Standalone Transformation Server prerequisites

The Alfresco Transformation Server requires prerequisite software components to be installed and available on the same machine.

The following software is required:

- Microsoft Windows 2008 Server R2 SP1 x64 with the latest hot fixes (English)
- Microsoft Office 2010 SP1 x86 (English)
- Java Development Kit 1.6 update 30 x86 (or later updates of the JDK 1.6)

The following points are important to note before you install the Transformation Server:

- Install only the English versions of MS Windows Server 2008 and Office 2010 because other languages result in unpredictable behavior.

  > Although the server must be configured in English, this has no impact on the transformation language used for documents.

- Make sure that the Windows print spooler service is running
- Java 7 is not supported
- Java 6 x64 is not supported

Alfresco Transformation Client prerequisites

The Alfresco Transformation Client is available for Alfresco Enterprise 4.0 and above.

Transformation Server License

The Alfresco Transformation Server is sold as a separate product, which can only be enabled with a separate license key.

The license key replaces your existing Enterprise license key.

Request your Transformation Server license key from Alfresco Support.

Installing the Alfresco Transformation Server

This section describes how to install all the components required for the Alfresco Transformation Server.

The following artifacts are shipped for the Alfresco Transformation Server:

- `alfresco-4.1-transformationserver-amps-2.1.3.zip`
- `alfresco-4.1-transformationserver-server-1.3.3.zip`

The Transformation Server AMP zip file (`alfresco-4.1-transformationserver-amps-2.1.3.zip`) contains the following artifacts:

- `alfresco-transformationserver-repo-2.1.3.amp`
- `alfresco-transformationserver-share-2.1.3.amp`
- `TransformationServer-amps-2.1.3-releaseNotes.html`

The two AMP files must be applied to the repository and Share, respectively. The Share AMP file also adds a page to the Share Admin Console, which gives information on the status of the Transformation Server.

The Transformation Server zip file (`alfresco-4.1-transformationserver-server-1.3.3.zip`) contains the following artifacts:

- `alfresco-4.1-transformationserver-server-1.3.3.msi`
- `TransformationServer-server-1.3.3-releaseNotes.html`

Installing the Transformation Server consists of two parts:

1. Installing the MSI installation package on the standalone Transformation Server.
2. Installing the relevant AMP package and updating the license on the Alfresco server.

When upgrading the Transformation Server, the previous installation must be uninstalled first. If your old version of the Transformation Server is earlier than 1.3.1, you will need to use the Control Panel's Uninstall a program option to remove the old version, and then manually remove the Transformation Server directory (by default, `C:\Program Files (x86)\Transformation Server\`). If your old version of the Transformation Server is 1.3.1 or later, the new Transformation Server msi will prompt you to let it uninstall the previous version. Once that is done, you can run the msi again to install the new version. There is no need to manually remove anything.

Installing the standalone Transformation Server

This section describes how to install the standalone Transformation Server.

Before you start the installation, verify that you have:

- installed and activated Windows 2008 Server
- installed and activated Microsoft Office 2010
- logged on to the Windows Server as a user with administrator rights

1. Double click the MSI installer package `alfresco-4.1-transformationserver-server-1.3.3.msi`.

   The Welcome screen displays.
2. Click **Next**.

   The copyright information screen displays.
3. Click **Next**.
4. Select an installation folder or accept the default folder, and then click **Next**.
5. Click **Next** to start the installation.

You see a progress bar during the installation and the **Cluster Setting** dialog box displays. Use this dialog box to configure clustered transformation servers.

6. Choose the number of servers you have from the **Number of servers** list.
7. If you have multiple transformation servers, complete the following steps to configure them.
   a. In the **Log database host** field, specify the hostname of the remote transformation server where the logs from this server should be sent.
   b. In the **Cluster node name** field, specify a unique name for this transformation server.
   c. Click **OK** to continue.
8. If you do not have multiple transformation servers, click **OK** to continue.
9. Click **Next** to finish the installation.

   The **Alfresco Transformation Server has been successfully installed** message displays.
10. Click **Close**.
11. Verify that the installation has completed successfully.
    a. Check the Windows Services in the management console.
    b. Locate the new service called **Transformation Service**, and check that it is set to **Started**.

Installing the Transformation Server on Alfresco

This section describes how to install the Transformation Server AMP and to update the required license.

Before you start, make sure that you verify the following prerequisites:

- Check that your Alfresco Enterprise server is correctly configured and tested
- Make sure that you have the correct Transformation Server ZIP file for the version of Alfresco that you are running
- Make sure that you have an updated license file (a `*.lic` file)

1. Stop the Alfresco server.
2. Open a terminal (Linux) or command line window (Windows).
3. Navigate to the `<ALFRESCO_HOME>/amps` directory.
4. Copy the AMP files into the relevant AMPs directories for Alfresco and Share:
   a. Copy the `alfresco-transformationserver-repo-2.1.2.amp` file to the `<ALFRESCO_HOME>/amps` directory.
   b. Copy the `alfresco-transformationserver-share-2.1.2.amp` file to the `<ALFRESCO_HOME>/amps_share` directory.
5. Install the AMP package using the `apply_amps` command.
   - Linux: `bin/apply_amps.sh`
   - Windows: `bin\apply_amps.bat`
6. Copy your updated license file into the Alfresco installation folder.

   Delete all files with extension `*.installed` in this directory.
7. Start the Alfresco server.
8. Monitor your Alfresco log.

You will see successful log entries about the license installation and the installation of the Alfresco Module Package (depending on the configuration of your log level).

Configuring the Alfresco Transformation Server

This section describes how to configure the components installed for the Alfresco Transformation Server.

Configuring the Alfresco Transformation Server consists of two parts:

1. Configuring the Standalone Transformation Server using the Web Console.
2. Configuring the Alfresco Transformation Client using a properties file or JMX.

Configuring the Standalone Transformation Server

This section describes how to configure the Standalone Transformation Server. You need only to change the password of the transformation service.

1. Open your browser and navigate to the following URL:

   `http://<tranformation-host>:<port>/transformation-server/settings`

   (or `https://` when using SSL)
2. Enter your login name and a password.

   By default, the login name is set to `alfresco`, and the password is set to `alfresco`. The login name `alfresco` cannot be changed.
3. Enter a new password, and then click **Change** to save the password.

If you close and reopen your browser, reenter your login and new password.

Configuring the Transformation Client

This section describes how to configure the Transformation Client by defining several parameters like using HTTP or HTTPS, quality settings, and so on.

There are two ways that you can configure the Alfresco Transformation Client:

- Using the alfresco-global.properties file
- Using a JMX client

Configuration using the global properties file

You configure the Transformation Client by adding the relevant properties to the Alfresco global properties file.

1. Open the `alfresco-global.properties` file.
2. Add the required properties for configuration settings on the Transformation Client.
3. Save the `alfresco-global.properties` file, and then restart your Alfresco server.

The following table shows an overview of the available properties:

Property	Default value	Description
`transformserver.aliveCheckTimeout`	2	Sets the timeout for the connection tester in seconds. If the transformation server does not answer in this time interval, it is considered to be off line.

Property	Default value	Description
`transformserver.test.cronExpression`	0/10 * * * ?	Sets the cron expression that defines how often the connection tester will check. The default is every 10 minutes.
`transformserver.disableSSLCertificateValidation`	false	Set this property to true to allow self-signed certificates (that is, it is not issued by an official Cert Authority).
`transformserver.username`	alfresco	The user name used to connect to the Transformation Server. ✎ **Do not change** this default.
`transformserver.password`	alfresco	The password used to connect to the Transformation Server. ✎ **Always change the password from the default.**
`transformserver.qualityPreference`	QUALITY	There are two values for this property: • QUALITY: optimizes the SWF preview for quality. • SIZE: optimizes the SWF preview for size. This is interesting if you have a lot of big office docs, for example, PPT > 100 MB.
`transformserver.transformationTimeout`	300	Sets the time in seconds to wait for the transformation to complete before assuming that it has hung and therefore stop the transformation. If you are transforming very large or complex files, this time may need to be increased.
`transformserver.url`		The URL of your Transformation Server (or the network load balancer if you are using more then one Transformation Server). Use `https://` if you want to use encrypted communication between the Alfresco server and the Transformation Server.

In a normal setup, you will always overwrite the `transformserver.password` and `transformserver.url` properties. If you want to use SSL encryption with the default certificate of the transformation server, make sure that you set `transformserver.disableSSLCertificateValidation=true`.

Configuration using JMX

The Transformation Client configuration parameters are exposed as JMX MBeans, which means that you can view and set the parameters using a JMX client.

See Runtime administration with a JMX client for instructions on how to connect a JMX client to your Alfresco server.

Using the Transformation Server

Whenever you upload your Office files in Share, you will now be using the Alfresco Transformation Server, and you can see results in the Share preview.

Administrators can view information about the server and transformation errors using the Web Console.

Using the Transformation Server Web Console

Use the Transformation Server Web Console to view information about the server and transformation errors. The server lets you view the status of the server, a historical view of all the transformations completed, and the number of successful and failed transformations.

Only Administrators can access and use the Transformation Server Web Console.

1. To open the Transformation Server Web Console, open a browser, and then navigate to the following URL:

   ```
   http://<transformation-host>:</port>:/transformation-server/
   ```

   Use `https://` if you use SSL.

   The **Server Status** view is the default view when you open the Transformation Server Web Console. The **Server Status** view shows an overview of the health and the memory use of the Transformation Server. Ensure that you have the flash plug-in to see the **Active Threads** and **Memory Usage** graphics.
2. Click **History** view.

   Alternatively, you can go directly to the **History** view by opening a browser, and then navigating to the following URL:

   ```
   http://<transformation-host>:<port>:/transformation-server/
   transformations
   ```

   The **History** view shows the details of the document transformations. It provides a number of search functions that allow administrators to find transformation problems for specific documents.
3. You can query the transformation history using the following parameters:
   - Date-time From and To
   - File name
   - Status
   - User name
4. To investigate errors, set the **Outcome** field to **Error**. Hover over the warning sign to view an indication of the problem with the file.
5. Click the **Statistics** view.

   Alternatively, you can go directly to the **Statistics** view by opening a browser, and then navigating to the following URL:

   ```
   http://<transformation-host>:<port>:/transformation-server/stats
   ```

   The **Statistics** view indicates the number of transformations, and the success or failed ratio.
6. Click the reset link to reset the counter.

Integrating with monitoring tools

You can integrate the Alfresco Transformation Sever with monitoring tools, for example Nagios or Hyperic, by using HTTP REST calls.

The tool should call the Transformation Server URL with a set of parameters, and then monitor the response.

Two calls are available:

1. Connection tester call.

   This call is also used by the Alfresco Transformation Client to test availability. It checks the transformation service is up and responding.

   a. URL: http://<transformation-host>:<port>:/transformation-server/ /service/transform/v1/version
   b. HTTP Method: GET
   c. Make sure that you include basic authentication credentials to your call.
2. Transformation execution call.

   This call posts an Office file to the Transformation Service to check whether the transformation engine is still functioning. This can be used for more in-depth monitoring.

   a. URL: http://<transformation-host>:<port>:/transformation-server/ /service/transform/v1/available
   b. HTTP Method: POST
   c. Make sure that you include basic authentication credentials to your call.

Upgrading

This section describes the recommended procedure for performing an upgrade.

Upgrading Alfresco

Before starting an upgrade:

- Ensure that you have backed up your production environment, for example, back up your database and content store (`alf_data` directory)

  If Solr is being used, only the following directories must be backed up from the `dir.root` directory:

  - `contentstore` directory
  - `solr/workspace/` directory
  - `solr/archive/` directory
  - `contentstore.deleted` directory (optional)

  If Lucene is being used, only the following directories must be backed up from the `dir.root` directory:

  - `contentstore` directory
  - `lucene-indexes` directory
  - `contentstore.deleted` directory (optional)
- If you are upgrading from Alfresco version 3.4.x to 4.x.y, ensure that Lucene subsystem is enabled for the duration of the upgrade. You can enable the Solr subsystem after the upgrade is complete. However, if you are upgrading from Alfresco version 4.0.x to 4.1.x and use Solr, you do not need to enable Lucene; Solr is automatically enabled.
- 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.
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 Alfresco repository.
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.

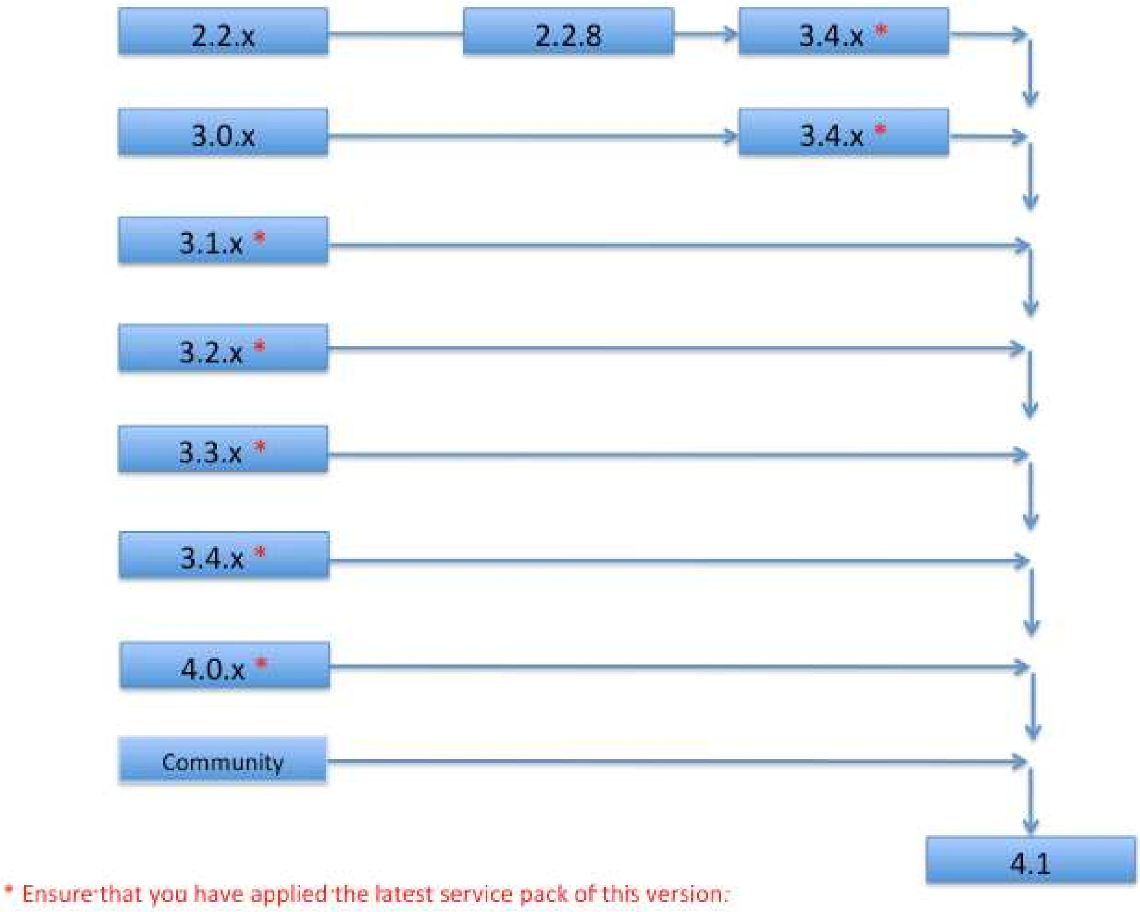
5. Download and install the new version of the Alfresco WAR in a different directory to the existing installation. See Installing Alfresco.
6. Validate the new installation to check that it is working correctly. See Validating the upgrade.
   a. Configure the new installation with a new repository (not the existing one).
   b. Start Alfresco and validate that the system works correctly.
   c. Shut down Alfresco and (optionally) clear the new repository.
7. Manually check your existing custom overrides or configurations in your original configuration file copies and only update/add those appropriate configurations or files to the newer version in the new extension files. See Configuring the installation.
8. Deploy your customizations into the new Alfresco instance.
9. 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.
10. Fully test the working and configuration of your customizations.
11. Shut down Alfresco.
12. Clear out the Solr `alfrescoModels` and the indexes (or in case of Lucene, the Lucene indexes).
13. Clear out your existing repository.
14. Restore the backup into the new repository (choose `index.recovery.mode=AUTO` for Lucene indexes).
15. Start Alfresco.
16. Once you are happy with the upgraded system, remove the old Alfresco installation and repository.

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. Alfresco 4.0.x, 3.x, and 2.2.x are considered to be major versions.

The following diagram shows the available upgrade paths:



The following list describes the upgrade paths:

- Direct upgrades to 4.1.5 are supported from only 3.1.x and later, with the latest Service Pack applied.
- Upgrades from Alfresco 3.0.x require the latest service pack of version 3.4.x before upgrading to version 4.1.5.
- Upgrades from Alfresco 2.2.x require the Service Pack 2.2.8 to be applied first, and then the latest service pack of version 3.4.x before being able to upgrade to version 4.1.5.

If you are upgrading from an earlier release that is not shown in this section, contact Alfresco Support for assistance.

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. Save the file.
8. 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.
9. Start the server.

The configuration overrides will ensure that the server immediately directs data to the appropriate locations.

Upgrading configurations

This page describes the important information for upgrading from Alfresco Enterprise releases prior to version 3.2.

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

When you upgrade 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=`

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.

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.

If you want your existing jBPM workflows to work after an upgrade, the `system.workflow.engine.jbpm.enabled` property should be set to `true` in the `alfresco-global.properties` file. However, the jBPM workflow definitions will be hidden by default, so new jBPM workflows cannot be started.

```
system.workflow.engine.jbpm.enabled=true
```

Validating an upgrade

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.

Upgrading a cluster

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.

   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.

Upgrading multi-tenancy

If upgrading to the latest Alfresco version, your existing MT sample extension files are no longer relevant and should be deleted. It is also recommended that you backup your existing MT files. Follow the steps below to upgrade multi-tenancy in Alfresco:

1. Perform the steps described in the Upgrading Alfresco section.
2. Take a backup of the following three existing MT extension files and delete them from the existing MT extension directory:

   a. `alfresco/extension/mt/mt-context.xml` to `alfresco/extension/mt/mt-context.xml`

b. `alfresco/extension/mt/mt-admin-context.xml` to `alfresco/extension/mt/mt-admin-context.xml`

c. `alfresco/extension/mt/mt-contentstore-context.xml` to `alfresco/extension/mt/mt-contentstore-context.xml`

Administering

This section provides guidance on configuring, maintaining, and administering an Alfresco production environment.

Starting and stopping

This section describes how to run the Alfresco server, Share, Explorer, virtualization server, and standalone deployment engine.

Starting the Alfresco server

- (Windows)
  - Browse to `C:\Alfresco`, and double-click `servicerun.bat`, or open the Control Panel **Services** window and start the following services:
    - `alfrescoPostgreSQL`
    - `alfrescoTomcat`
  - If you installed Alfresco as a service, from the **Start** menu, select **All Programs > Alfresco Enterprise > Alfresco Enterprise Service > Start Alfresco Enterprise service.**
- (Linux) Browse to `/opt/alfresco/` and run `./alfresco.sh start`.

  If you installed Alfresco using the setup wizard, the alfresco.sh script included in the installation disables the Security-Enhanced Linux (SELinux) feature across the system.

  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`.

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

```
INFO: Server startup in nnnn ms
```

Stopping the Alfresco server

- (Windows)
  - Open the Control Panel **Services** window and stop the following services:
    - `alfrescoPostgreSQL`
    - `alfrescoTomcat`
  - Click the **Start** menu, and select **All Programs > Alfresco Enterprise > Alfresco Enterprise Service > Stop Alfresco Enterprise service**.

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

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. For example, `http://<your-host>:8080/share`.

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

Alfresco Share opens.

2. Log in using `admin` as the default user name, and then enter the 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. For example, `http://<your-host>:8080/alfresco`.

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

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

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)
  - Click the **Start** menu, and select **All Programs > Alfresco Enterprise > Alfresco Enterprise Virtual Server > Start Virtual Server**.
- (Linux) Browse to `/bin` 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

This section describes how to stop the Website preview feature by stopping the Alfresco virtualization server.

- (Windows)
  - Click the **Start** menu, and select **All Programs > Alfresco Enterprise > Alfresco Enterprise Virtual Server > Stop Virtual Server**.
- (Linux) Browse to `/opt/alfresco/`, and run `sh virtual_alf.sh stop`.

Starting the standalone 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 standalone 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

Configuration overview

Alfresco is preconfigured with a set of system configuration parameters. Many of the system configuration parameters are completely exposed as properties, which you can extend or override.

The system configuration parameters are found in the following files:

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

It is not recommended that you edit these files directly but that you should extend or override them using the following methods:

- Editing the global properties (`alfresco-global.properties`) file
- Admin Console in Share
- 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 Admin Console in the Share user interface is an alternative way of making changes to some of the Alfresco configurations and maintaining Share.

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. 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 client 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-services-context.xml` file.
2. Comment out the `RMIRegistryFactoryBean` section.
3. Save the file.

   When you restart the server, you will see the error: `No bean named 'registry' is defined' when re-starting`.

Configuring Alfresco with JConsole

This section describes how to use the JMX client, JConsole for Alfresco runtime administration. JConsole is a JMX client available in the Oracle 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 the WAR file, you must modify properties in the sample `alfresco-global.properties` file manually.

A sample global properties file is supplied with the Alfresco installation. 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 properties.

Modifying the global properties file

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.sample` file.

   This file contains sample configuration settings for Alfresco. To enable or modify a setting, ensure that you remove the comment (#) character.
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. Set the database connection properties.

Property	Description
`db.username=alfresco`	Specifies the name of the main Alfresco database user. This name is used to authenticate with the database.
`db.password=alfresco`	Specifies the password for the Alfresco database user. This password is used to authenticate with the database.

   Additional database properties may be set for further configuration. Refer to the Configuring databases for more information.
5. Specify the locations of the following external software:

Property	Description
`ooo.exe=`	Specifies the location of the OpenOffice installation.
`ooo.enabled=`	Specifies whether to use the Direct OpenOffice subsystem.

Property	Description
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.
jodconverter.enabled=	Specifies whether to use the JODConverter. Set the property to jodconverter.enabled=true.
img.root=	Specifies the location of the ImageMagick installation.
swf.exe=	Specifies the location of the SWF tools installation.

6. Uncomment the db.driver= and db.url= properties for the database that you require.

   These properties are grouped into sections for MySQL, Oracle, SQL Server, and PostgreSQL connection, each containing sample settings.
7. Select a JDBC driver used with each connection type.
8. Add your global custom configurations.
9. Save your file without the .sample extension.

You need to restart the Alfresco server for the configuration changes to take effect.

Setting composite properties in the global properties file

This section uses 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

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, Content Store locations, and CIFS domain name.

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`.

Modifying Spring bean definition files

The Spring bean definitions are within configuration files 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.

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

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

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 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 wild card 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 the core system properties, including:

   - `dir.root`

     This folder is where the binary content and indexes are stored. The `alf_data` folder is where they are stored by default, but you should change this to your own location. The path is relative by default, but it 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.

     Alfresco recommends that you do not store Lucene indexes on an NFS volume. The indexes must be on a local disk. For best performance, use a separate hardware chain (for example, controller, disk, and so on) to avoid I/O contention with other operations, like storing content and other applications.
   - `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.

Customizing individual configuration items

The Alfresco configuration is implemented using three types of files:

- Properties files
- Configuration files
- Bean files

Customizing properties files

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 configuration files

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.
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.
4. To add another item, you only need a single item. The other items will remain enabled.
5. Customize the contents of the remaining `<config>` tags.
6. Save your customized file.

Configuration files

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>
```

Customizing bean 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 resulting bean must still have four values. However, if you want to add a value, then the resulting 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.

Managing Alfresco using the Admin Console

The Admin Console is a browser-based console that lets you manage your administration operations.

In the Admin Console, you can manage users, groups, email, file systems, and transformer availability. You can also view information about the repository and server, and statistics (like the number of users/groups, storage used, and so on). You can also manage archived documents.

The Admin Console is visible only if you are an Administrator user or a user who is a member of the `ALFRESCO_ADMINISTRATORS` group.

Opening the Admin Console

You can only see the Admin Console if you are an administrator user or a user who is a member of the `ALFRESCO_ADMINISTRATORS` group.

1. On the toolbar, expand the **More** menu, and then click **More** in the **Admin Tools...** list.
2. Click the tools on the left navigation bar.

   You see the various tools available and the options that you can set.

   Alternatively, you can select a number of the tools directly from the **More** menu. The **More** menu shows the most commonly used tools in the Admin Console, for example:

   - **Application**
   - **Groups**
   - **Replication Jobs**
   - **Repository**
   - **Trashcan**
   - **Users**

Admin Console tools in the More menu

The **More** menu on the toolbar contains links to Admin Console pages you may wish to access often. These links to the Admin Console page are visible only to Administrators.

The **More** menu contains the following Admin Console links (shown below **Admin Tools...**):

Application
Shows the **Application** tool where you set the theme and logo for Alfresco.

Groups
Shows the **Groups** tool where you create and manage the user groups.

Replication Jobs
Shows the summary of replication jobs tool where you can create new jobs.

Repository
Shows the **Repository** tool where you can manage the repository, enable Google Docs integration, and download the JMX Zip Dump.

Trashcan
Shows the **Trashcan** tool where you access the deleted items, and you can purge or restore items.

Users
Shows the **Users** tool where you create and manage the user accounts.

More

Opens the page showing all the available Admin Console tools.

Specifying application preferences

The Application tool in the Admin Console lets you set application preferences.

Changing the look and feel theme

The look and feel of the user interface is set by a theme. The Application tool lets you select a color scheme for the user interface.

1. Open the Admin Console, and then click **Application**.
2. On the **Options** page, select a theme from the list.

   Choose one of the available themes:

   - **Green Theme**
   - **Default Blue Theme**
   - **Yellow Theme**
   - **Google Docs Theme**
   - **High Contrast Theme**
3. Click **Apply**.

The page refreshes to display with the selected theme. The changed theme affects all users from the next time they login and persists across sessions.

A new installation uses the default theme, which comprises the CSS and image assets used across all pages.

Site managers can customize the theme for an individual site. If a site theme has been changed, this will override any theme setting made in the Admin Console.

Changing the logo

The Alfresco logo on the top left is at the top left-side of Share. You can change the logo to another image file.

1. Open the Admin Console, and then click **Application**.
2. On the **Options** page, click **Upload**.

   You'll see the **Upload File** window.
3. Click the icon to browse for a file to upload.
4. Choose a file and click **Open**.

   You can choose to upload any image you like but there are some recommendations for suitable sizes for the image. The maximum recommended image height for your image file is 48 pixels.

   The file you chose shows in the **Upload File** window. If it's not the right file, you can click **Remove** to start again.
5. Click **Upload File(s)**.
6. When you see that the file is successfully uploaded, click **OK**.
7. Click Apply.

   The newly uploaded file now becomes the logo for Alfresco.
8. If you wish to change the logo back to the default Alfresco logo, click **Reset** to display the original logo, and then click **Apply**.

Managing categories

Manage your categories on the **Category Manager** page.

1. Open the Admin Console, and then click **Category Manager**.

   The **Category Manager** page shows a tree structure of the categories created in the system. The top level is called **Category Root** and by default, the following sub-categories are listed:

   - **Languages**
   - **Regions**
   - **Software Document Classification**
   - **Tags**

2. Click the category icons (▤) to expand the list of categories.

   When you hover over the category name, you see the available action icons for: **Edit category** (✎), **Add category** (✚), and **Delete category** (✖).

3. To edit a category, click the **Edit Category** icon, edit the category name inline, and then click **Save**.

4. To add a category, click the **Add Category** icon, enter a name in the **Category name** field, and then click **OK**.

   When using Solr, there maybe a delay before the new category appears in a search query until after Solr has been reindexed. Categories are eventually consistent.

5. To delete a category, click the **Delete Category** icon, and then click **Delete** to confirm that you wish to delete the category.

   The category is deleted from the system. Any content is removed from that category label.

Using the Node Browser

This is a read-only feature with basic search capability.

1. Open the Admin Console, and then click **Node Browser**.

   By default, the search criteria `PATH:"/"` is shown in the field. This shows the results for the `workspace://SpacesStore` repository store.

2. Click the **Select Store** list to select a store in the repository.

   Each store is an area of the repository. The nodes contained within each store are organized hierarchically. The node displayed is the root node of the selected store.

3. Search the selected store:

   a. Select the search type:

      - **noderef**
      - **xpath**
      - **jcr-xpath**
      - **lucene**
      - **fts-alfresco**
      - **cmis-strict**
      - **cmis-alfresco**

      The default is **fts-alfresco**.

   b. Enter the search criteria in the field.

      Use the search syntax relevant for your chosen search type.

   c. Click **Search**.

4. Click the reference link to browse the details.

   The details of the properties, aspects, children, parents, associations, source associations, and permissions are displayed for the node.

5. Click Back to Search to browse another node.

Managing tags

Tags can be added to content within the Document Library. Use the **Tag Manager** page to view, edit, and delete all the tags that have been created by users.

1. Open the Admin Console, and then click **Tag Manager**.

   The **Tag Manager** page shows a list of the tags that have been created, the name of the user who created or modified the tag, and the date on which the change was made.

   If there are no tags in the system, you see the message: **No tags found**.

   When you hover over the right hand **Actions** column, you see the available action icons for: **Edit tag** (✎) and **Delete tag** (✗).

   a. To edit a tag, click the **Edit tag** icon, edit the tag name in the **Rename Tag** field, and then click **OK**.

   b. To delete a tag, click the **Delete tag** icon, and then click **Delete** to confirm that you wish to delete the tag.

      The tag is deleted from the system and removed from any content where it was previously tagged.

2. Click the tag name to see a list of the repository content that uses this tag.
3. Click the user name to see the profile of the user who last modified the tag.

Emptying deleted files from the Trashcan

When a user deletes any content, the file is not completely removed from the system. At first, it is placed in the Trashcan.

On the Trashcan page, you’ll see a list of the content that has been deleted by users. The content items are listed in date order. If there are no deleted content items, you’ll see a message saying: **No items exist**.

1. Open the Admin Console, and then click **Trashcan**.
2. On the **Trashcan** page, select the content item that you wish to remove from the Trashcan.

   a. To remove an item from the Trashcan, click **Delete** next to the name.

   b. To remove all items from the Trashcan, click **Empty**.

3. To return the item back to the original place in the Document Library, click **Recover**.

Managing social content publishing

The current out-of-the-box support is for the following social platforms:

- Facebook
- Flickr

- Linkedin
- SlideShare
- Twitter
- YouTube

Any content asset from the document library can be published to any appropriate publishing channel.

The framework allows you to develop and plug-in additional channels.

Create a new channel

Create new channels on the Channel Manager page.

1. Open the Admin Console, and then click **Channel Manager**.

   On a new installation, there are no existing channels created.
2. Click **New**, and then select the required channel type.

   Choose from the following channels:

   - Facebook
   - Flickr
   - LinkedIn
   - SlideShare
   - Twitter
   - YouTube
3. Follow the setup instructions for the channel you choose.

   When you access Share for the Admin Console, use the correct URL for your Alfresco instance, rather than using `http://localhost:8080/share`. This ensures that the service provider for the relevant channel knows the location of Share when channel authorization is complete. If these are incorrect, then the authorization may fail.

Creating a Facebook channel

Create new Facebook channels on the Channel Manager page.

1. Open the Admin Console, and then click **Channel Manager**.
2. Click **New**.
3. Select the **Facebook**.

   You see the Facebook **Request for permission** page stating that Alfresco is requesting permission for access to your Facebook account.
4. Click **Allow**.

   You see a popup window indicating that a new Facebook channel has been created. The Facebook channel displays on the **Channel Manager** page in the Admin Console.

The name of the new channel is **New Facebook channel**. Hover the mouse pointer over the name and you see an edit icon. Click the icon to change the channel name.

Creating a Flickr channel

Create new Flickr channels on the Channel Manager page.

1. Open the Admin Console, and then click **Channel Manager**.
2. Click **New**.

3. Select the **Flickr**.

   You see the Flickr **Sign in to Yahoo** authorization page.
4. Enter your Yahoo ID and password, and then click **Sign in**.

   You are then requested to authorize the Alfresco can access the Flickr account.
5. Click **OK, I'LL AUTHORIZE IT**.

   You see a popup window indicating that a new Flickr channel has been created. The Flickr channel displays on the **Channel Manager** page in the Admin Console.

The name of the new channel is **New Flickr channel**. Hover the mouse pointer over the name and you see an edit icon. Click the icon to change the channel name.

Creating a LinkedIn channel

Create new LinkedIn channels on the Channel Manager page.

1. Open the Admin Console, and then click **Channel Manager**.
2. Click **New**.
3. Select the **LinkedIn**.

   You see the LinkedIn authorization page.
4. Enter your LinkedIn user name and password, and then click **Sign in**.

   LinkedIn will then list the applications that you have authorized.

   The LinkedIn channel displays on the **Channel Manager** page in the Admin Console.

The name of the new channel is **New LinkedIn channel**. Hover the mouse pointer over the name and you see an edit icon. Click the icon to change the channel name.

Creating a SlideShare channel

Create new SlideShare channels on the Channel Manager page.

1. Open the Admin Console, and then click **Channel Manager**.
2. Click **New**.
3. Select the **SlideShare**.

   You see the SlideShare authorization page.
4. Enter your SlideShare user name and password, and then click **OK**.

   SlideShare will then list the applications that you have authorized.

   The SlideShare channel displays on the **Channel Manager** page in the Admin Console.

The name of the new channel is **New SlideShare channel**. Hover the mouse pointer over the name and you see an edit icon. Click the icon to change the channel name.

Creating a Twitter channel

Create new Twitter channels on the Channel Manager page.

1. Open the Admin Console, and then click **Channel Manager**.
2. Click **New**.
3. Select the **Twitter**.

   You see the Twitter authorization page requesting the user name and password of the Twitter account.
4. Enter your Twitter user name and password, and then click **Authorize app**.

   You see a popup window indicating that a new Twitter channel has been created. The Twitter channel displays on the **Channel Manager** page in the Admin Console.

The name of the new channel is **New Twitter channel**. Hover the mouse pointer over the name and you see an edit icon. Click the icon to change the channel name.

Creating a YouTube channel

Create new YouTube channels on the Channel Manager page.

1. Open the Admin Console, and then click **Channel Manager**.
2. Click **New**.
3. Select the **YouTube**.

   You see the Alfresco Channel Authentication page requesting your user name and password for the new YouTube channel.
4. Enter your YouTube account user name and password and then click **Login**.

   You see a popup window indicating that a new YouTube channel has been created. The YouTube channel displays on the **Channel Manager** page in the Admin Console.

The name of the new channel is **YouTube channel**. Hover the mouse pointer over the name and you see an edit icon. Click the icon to change the channel name.

Changing a channel permission

Permissions can be set either by

- Inherited permissions
- Locally set permissions

When a channel is created, all users are set to the Consumer role by default. This means that no users will have permission to publish content except the administrative user.

1. Open the Admin Console, and then click **Channels Management**.
2. Locate the channel that you wish to change the permissions.
3. Click **Permissions**.

   You see the **Manage Permissions** page for the channel. Permissions are inherited by default. To disable inherited permissions, click **Inherit Permissions**, and then click **Yes**.

Authorizing a channel

1. Open the Admin Console, and then click **Channels Management**.
2. Locate the channel that you wish to reauthorize.
3. Click **Reauthorize**.

   You see the relevant authorization pages for the channel.
4. Follow the instructions for authorizing the channel.

Deleting a channel

1. Open the Admin Console, and then click **Channels Management**.
2. Locate the channel that you wish to delete.
3. Click **Delete**.

   You are asked to confirm that you wish to delete this channel.
4. Click **OK**.

   The channel is deleted and the channel icon is removed from the Channels Management page.

Managing activities feed emails

The Activities Feed shows the settings for activity emails in your Alfresco environment. Activity emails are sent to the Administrator user to provide information for managing Alfresco.

Email notification are sent for all events that you would see in the activity feed dashlet. This includes events that are triggered in all of the sites that you are a member of and of all people you are following. Scheduled activity digests are sent with an email, based on the data that is available in your personal activity feed.

1. Open the Admin Console, and then click **Activities Feed**.
2. On the **Activities Feed** page, click **Edit**.

   You see the **Edit: Activities Feed** page.
3. To activate emails for the Activities Feed, select the **Feed Notifier Enabled** check box.
4. Change the properties to control the frequency of emails and number of items shown.

Property	Example setting	What is it?
Maximum Age (mins)	44640	This sets the maximum age in minutes of the activity events shown in the Activities Feed notification emails. Activities that are older than the maximum age are not shown. The default is set to 44640 (31-day month).
Repeat Interval (mins)	1440	This sets how often that you will receive the Activities Feed notification emails. The default is set for you to receive emails every day.
Maximum Size	100	This sets the maximum number of activity events that are reported on in the Activities Feed notification emails.

5. When you've finished changing the properties, click **Save**.

   If you do not want to save the changes, click **Cancel**.
6. Finally, you need to make sure that you have set the **Host** property on the **Email (Outbound)** page in the Admin Console.

   Setting this property to the email host means that users can receive emails, and send and receive site invites. Individual users can enable or disable email notifications in their **My Profile** settings.

Activities feed email notifications list

The email description is generated using the following format, for example, for a newly created blog post:

```
org.alfresco.blog.post-created={1} created blog post {0}
```

Where

- 0 = Item title / page link
- 1 = User profile link
- 2 = custom0

- 3 = custom1
- 4 = Site link
- 5 = second user profile link

The following table shows the list of activities that trigger an email notification.

Activity	Email description
Create a blog post	`org.alfresco.blog.post-created={1} created blog post {0}`
Update a blog post	`org.alfresco.blog.post-updated={1} updated blog post {0}`
Delete a blog post	`org.alfresco.blog.post-deleted={1} deleted blog post {0}`
Write a comment	`org.alfresco.comments.comment-created={1} commented on {0}`
Update a comment	`org.alfresco.comments.comment-updated={1} updated comment on {0}`
Delete a comment	`org.alfresco.comments.comment-deleted={1} deleted a comment from {0}`
Start a discussion	`org.alfresco.discussions.post-created={1} started discussion {0}`
Update a discussion	`org.alfresco.discussions.post-updated={1} updated discussion {0}`
Delete a discussion	`org.alfresco.discussions.post-deleted={1} deleted discussion {0}`
Reply to a discussion	`org.alfresco.discussions.reply-created={1} replied to the discussion {0}`
Update a reply	`org.alfresco.discussions.reply-updated={1} updated a reply to {0}`
Create a calendar event	`org.alfresco.calendar.event-created={1} created calendar event {0}`
Update a calendar event	`org.alfresco.calendar.event-updated={1} updated calendar event {0}`
Delete a calendar event	`org.alfresco.calendar.event-deleted={1} deleted calendar event {0}`
Add a document	`org.alfresco.documentlibrary.file-added={1} added document {0}`
Add multiple documents (showing the number of documents)	`org.alfresco.documentlibrary.files-added={1} added {0} documents`
Create a document	`org.alfresco.documentlibrary.file-created={1} created document {0}`
Delete a document	`org.alfresco.documentlibrary.file-deleted={1} deleted {0}`
Delete multiple documents	`org.alfresco.documentlibrary.files-deleted={1} deleted {0} documents`
Update a document	`org.alfresco.documentlibrary.file-updated={1} updated document {0}`
Update multiple documents	`org.alfresco.documentlibrary.files-updated={1} updated {0} documents`

Activity	Email description
Checkout a document to Google Docs	`org.alfresco.documentlibrary.google-docs-checkout={1} checked out document {0} to Google Docs`
Checkin a document from Google Docs	`org.alfresco.documentlibrary.google-docs-checkin={1} checked in document {0} from Google Docs`
Edit documents	`org.alfresco.documentlibrary.inline-edit={1} edited document {0}`
Like a document	`org.alfresco.documentlibrary.file-liked={1} liked document {0}`
Like a folder	`org.alfresco.documentlibrary.folder-liked={1} liked folder {0}`
Create a wiki page	`org.alfresco.wiki.page-created={1} created wiki page {0}`
Update a wiki page	`org.alfresco.wiki.page-edited={1} updated wiki page {0}`
Rename a wiki page	`org.alfresco.wiki.page-renamed={1} renamed wiki page from {2} to {0}`
Delete a wiki page	`org.alfresco.wiki.page-deleted={1} deleted wiki page {0}`
Add a group to a site with a particular role	`org.alfresco.site.group-added={1} group added to site {4} with role {2}`
Remove a group from a site	`org.alfresco.site.group-removed={1} group removed from site {4}`
Change a group role	`org.alfresco.site.group-role-changed={1} group role changed to {2}`
Join a site with a particular role	`org.alfresco.site.user-joined={1} joined site {4} with role {2}`
User left a site	`org.alfresco.site.user-left={1} left site {4}`
User role changed to a particular role	`org.alfresco.site.user-role-changed={1} role changed to {2}`
Like a site	`org.alfresco.site.liked={1} liked site {0}`
Create a link	`org.alfresco.links.link-created={1} created link {0}`
Update a link	`org.alfresco.links.link-updated={1} updated link {0}`
Delete a link	`org.alfresco.links.link-deleted={1} deleted link {0}`
Create a data list	`org.alfresco.datalists.list-created={1} created data list {0}`
Update a data list	`org.alfresco.datalists.list-updated={1} updated data list {0}`
Delete a data list	`org.alfresco.datalists.list-deleted={1} deleted data list {0}`
User following another user	`org.alfresco.subscriptions.followed={1} is now following {5}`

Activity	Email description
User subscribed	`org.alfresco.subscriptions.subscribed={1} has subscribed to {2}`
Change profile	`org.alfresco.profile.status-changed={1}: {2}`

Managing file servers

The **Fileservers** page handles the properties for the CIFS and FTP servers.

1. Open the Admin Console, and then click **Fileservers**.
2. On the **Fileservers** page, click **Edit**.

   You see the **Edit: Fileservers** page. The page is divided up into sections for Filesystem properties, CIFS properties, and FTP properties.
3. Set the Filesystem property:

Filesystem property	Example setting	What is it?
Filesystem Name	Alfresco	This is the name given to the file system when using CIFS, WebDAV, or FTP. For example, Alfresco.

4. Set the CIFS properties for connecting to Alfresco using CIFS:

CIFS property	Example setting	What is it?
CIFS Enabled	enabled	This checkbox enables or disables the CIFS server.
Host Announce	enabled	This checkbox enables the announcement of the CIFS server to the local domain/ workgroup so that it shows up in Network Places/Network Neighborhood.
Domain		This is the domain or workgroup to which the server belongs. This defaults to the domain/workgroup of the server, if not specified.
Server Name	${localname}A	This is the host name for the Alfresco CIFS server. This can be a maximum of 16 characters and must be unique on the network. You can use the special token {localname} in place of the local server's host name and a unique name can be generated by prepending/ appending to it.

CIFS property	Example setting	What is it?
Session Timeout	900	This is 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.

5. Set the FTP properties for connecting to Alfresco using FTP:

FTP property	Example setting	What is it?
FTP Enabled	Enabled	This checkbox enables or disables the FTP server.
Port	2121	This is the port that the FTP server listens for incoming connections on.
Dataport From	0	This sets the lower limit for the data ports range of ports.
Dataport To	0	This sets the upper limit for the data ports range of ports.

6. Click **Save**.

   If you do not want to save the changes, click **Cancel**.

Integrating with Google Docs

Alfresco provides a way to integrate with the documents that you want to store and edit in Google Docs. This feature is not available to you out of the box and you must enable the settings and Google Docs user name before you can start using it.

Enabling Google Docs integration

The **Google Docs** page handles the properties for the integration of content between Google Docs and Alfresco. Use this page to enable the Google Docs integration feature.

Before you enable Google Docs integration, set up a Google Docs account that will be used for administrative purposes for the integration between Alfresco and Google Docs. This administration Google Docs account will be viewed by all Alfresco users and is intended only to be used for the documents that you want to share between Alfresco and Google Docs.

1. Open the Admin Console, and then click **Google Docs**.
2. On the **Google Docs** page, click **Edit**.

   You see the **Edit: Google Docs** page.
3. Set the Google Docs properties.

Google Docs property	Example setting	What is it?
Enabled	disabled	Click this check box to enable the Google Docs feature. You also need to type in a Google Docs user name and password for the feature to be fully enabled.

Google Docs property	Example setting	What is it?
Username		This is the administrative Google Docs user name that is used for the integration between Alfresco and Google Docs. For users to be able to access the shared documents in the Google Docs administration account, they must add their personal Google Docs user name in their own profile. ✎ As the administrator, you can set a personal Google Docs user name in your profile. To avoid confusion, use a different Google Docs user name in the profile from the administrative Google Docs user name on the Admin Console.
Password		This is the password of the administrative Google Docs user name.

4. Click **Save**.

   If you do not want to save the changes, click **Cancel**.

Google Docs upload and create formats

Google Docs restricts the formats of files or documents that can be uploaded or created.

The following table shows the file format restrictions for content that integrates with Google Docs.

File extension	MIME type
`DOC`	application/msword
`XLS`	application/vnd.ms-excel
`PPT`	application/ppt

Google Docs integration supports only Office 2003 formats.

When you upload a supported MIME type, you'll be able to apply the *Google Docs Editable* aspect to the content, and then it can be checked out to Google Docs. For all unsupported MIME types, the *Google Docs Editable* aspect will not be available.

Disabling Google Docs integration

The **Google Docs** page handles the properties for the integration of content between Google Docs and Alfresco.

1. Open the Admin Console, and then click **Google Docs**.
2. On the **Google Docs** page, click **Edit**.

   You see the **Edit: Google Docs** page.
3. Deselect the **Enabled** check box.

   Do not remove the Google Docs user name and password.

4. Click **Save**.

   If you do not want to save the changes, click **Cancel**.

This disables the Google Docs feature.

Managing your Alfresco license

Your access and use of Alfresco is managed by a license. The license sets the maximum number of users and a maximum number of content objects that you can use. The license is not tied to specific servers, for example, by IP or MAC address.

You need to purchase and upload licenses appropriate to the number of users, content objects, and your support requirements.

Viewing your license details

The License Descriptor page shows the details of your license. Use this page to check your license restrictions and use.

1. Open the Admin Console, and then click **License Descriptor**.
2. View the description your license.

License Descriptor property	What is it?
License Subject	This shows the version that you have installed.
Days	This shows the number of days that your license subscription covers.
Holder	This shows the name of the license holder. For example, this could be your company name.
Valid Until	This shows the date and time when your license will expire.
License Mode	This shows the type of license that is issued.
Max Content Objects	This shows the maximum number of content objects that you can have in the system.
Max Users	This shows the maximum number of users that you can have in the system.
Remaining Days	This shows the number of days remaining before your license expires.
Issued	This shows the date when your license was issued by Alfresco.
Issuer	This shows the original location for where the license was generated.
Heart Beat Disabled	This shows whether your license allows the heartbeat functionality to be disabled, which means that automatic repository statistics are sent to Alfresco.

3. View the information about your license usage.

License Usage Information property	What is it?
Users	This shows the current number of users named in the system. The value of this property may show 0 users if you have a license that has no user restrictions.

License Usage Information property	What is it?
Content Objects	This shows the current number of content objects in the system, which includes documents and the related thumbnails, wiki items, blog posts, and so on.

Uploading a new license

When you first run Alfresco, it defaults to using a 30-day trial license. This section describes how to upload your own Alfresco Enterprise license file. The license file sets the capabilities of your Alfresco system.

You will receive an email confirming your access to the Support Portal, where you can retrieve your specific Enterprise license key. For more information on downloading your license key, see the Knowledge Base article in the Support Portal: **How can I download my Alfresco Enterprise License File/key?**.

The license file has a filename of `<license-name>.lic`.

There are two different methods that you can use to upload you license file.

Uploading your license using the Alfresco Admin Console

Before you start, ensure that Alfresco is running and that you can access the Admin Console.

1. Copy the license file to the directory in which Alfresco is installed.

   The license file has an extension of `.lic`.

   For example, on Windows, copy the file to the `C:\Alfresco` directory; on Linux, copy the file to `/opt/alfresco-x.x.x`.
2. Open the Admin Console, and then click **License Descriptor**.
3. On the **License Descriptor** page, click **Edit**.

   You'll see the **Edit: License Descriptor** page.
4. Click **Load License**.

   This uploads the `<license-name>.lic` license file that is in the install directory.

You have installed your Alfresco license and you can see the new license restrictions in the **License Descriptor** in the Admin Console. Alfresco renames the file to `<license-name>.lic.installed`.

Uploading your license manually

This section is an alternative method of installing the license for Alfresco Enterprise.

This method of installing your license uses a `license` directory within the installed product. Before you start, ensure that the Alfresco server is not running.

1. Copy the license file to your machine.

   The license file has an extension of `.lic`.
2. Ensure that the Alfresco server is not running.
3. 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
   ```
4. Create a new directory called `license`.
5. Move the `.lic` file into the new `license` directory.

6. Start the Alfresco server.

You have installed the Alfresco license file.

When you run Alfresco, the server detects the existence of the `.lic` file and installs your license. Alfresco renames the file to `<license-name>.lic.installed`.

If you are installing Alfresco manually, you may need to add the `dir.license.external` property and directory location in the `alfresco-global.properties` file. For example, for Tomcat on Windows, add:

```
dir.license.external=C:\Alfresco\tomcat\shared\classes\alfresco\extension
\license
```

When your license is about to expire, you must purchase a new license and upload it to your system. When you purchase further licenses, repeat the steps using the new license file.

A license key is unique to a specific version of Alfresco. Note that when you upgrade to a new version of Alfresco, you will need to install a new license key.

Managing replication jobs

The Replication Jobs tool in the Admin Console enables you to create and manage replication jobs in Share.

A replication job specifies the content to be replicated; the day and time the job is to be performed; and the target location for the replicated content.

The job is controlled by the Replication Service, and it calls the Transfer Service, which allows folders and content to be automatically copied between Alfresco repositories. A replication job can be run according to a schedule or on-demand.

By default, any replicated content is read-only in the target repository. This ensures the integrity of the content is not compromised by uncontrolled updates.

Viewing a replication job

Select a replication job to view the job details and display the available actions.

1. On the toolbar, expand the **More** menu and click **Replication Jobs** in the Tools list.

   The Replication Jobs page displays a summary of recently run jobs and a list of existing replication jobs. In this list, use the menu provided to sort the jobs by Status, Name, and Last Run Date.
2. In the Jobs section, click a job to view its details.

   The job appears highlighted in the list and its details appear on the right side of the page.

Creating a new replication job

You can create any number of replication jobs to suit your needs.

1. Expand the **More** menu.
2. Select the **Replication Jobs** menu item.

   The **Admin Console** displays.
3. In the **Jobs** section, click **Create Job**.

   The **Create New Replication Job** page displays. Fields marked with an asterisk (*) are required.
4. In the **Jobs** section, type the name for the job in the **Name** field.
5. In the **Jobs** section, type the description for the job in the **Description** field.

6. In the **Payload** section, click **Select**.
7. Navigate the repository and click **Add** to the right of each branch of the repository that you want to include in the payload. This is the content that will be replicated (copied) when the job is run.
8. Click **OK**.
9. In the Transfer Target section, click **Select**.
10. Navigate the Transfer Target Groups and click **Select** to the right of the desired target.
11. Click **OK**.

    Out of the box, one target group, **Default Group**, is available. Use the repository browser to create additional target groups (Data Dictionary > Transfers > Transfer Target Group). A rule defined on the **Default Group** transfer target folder specializes the type of any folder created within it. Refer to Setting up replication jobs.
12. Select the **Schedule job** check box, then enter the date and time the job is to run. Specify the repeat period for this job.
13. Select the **Enabled** check box.

    You must enable a replication job for it to be run.
14. Click **Create Job**.

    The job created appears highlighted in the Jobs list. The job details appear on the right side of the page.

Managing existing jobs

The Replication Jobs page of the Admin Console displays a list of all existing replication jobs. For each job in this list, you can perform any of the following actions to manage and maintain the jobs:

- Edit a job
- Manually run a job
- Cancel a job
- Delete a job

Editing a replication job

You can easily update existing replication jobs. In addition to changing the job details, you can use this feature to disable a job so that it will not be run.

1. On the toolbar, expand the **More** menu and click **Replication Jobs** in the Tools list.
2. In the Jobs section, click the job you want to edit.

    The job appears highlighted in the list and its details appear on the right side of the page.
3. Click **Edit**.

    The Edit Replication Job page appears.
4. Edit the replication job as necessary. All job details—name, description, payload, transfer target, and schedule—are available for editing.

    Add and remove source items as necessary. Click **Remove** to the right of a single item to remove it. Click **Remove All** beneath the list to remove all items.

    Deselect the **Enabled** check box to prevent the job from being run.
5. Click **Save**.

    The main page displays the updated job details.

Manually running a replication job

The **Run Job** feature enables you to manually run a replication job. You can do this at any time. If a schedule is set for the job, it remains in place and will be run at the appropriate time.

1. On the toolbar, expand the **More** menu and click **Replication Jobs** in the Tools list.
2. In the Jobs section, click the job you want to run.

   The job appears highlighted in the list and its details appear on the right side of the page.

   For a job to be run, it must be enabled.
3. Click **Run Job**.

   The Status section on the right side of the page indicates that the job is running. The date and time the job started is displayed.

Cancelling a replication job

You can cancel a job that is currently running, regardless of whether it was started automatically (that is, it is a scheduled job) or manually.

1. On the toolbar, expand the **More** menu and click **Replication Jobs** in the Tools list.
2. In the Jobs section, click the currently running job that you want to cancel.

   An icon (⟳) to the left of the job name indicates a job is currently running.

   The Status section on the right side of the page indicates the start time of the selected job.

   If the job was already displayed, you may need to click **Refresh** to update the status.
3. Click **Cancel Job**.

   The job is stopped and a report is created.

Deleting a replication job

If you no longer need a replication job, you can delete it from the Jobs list. If there is a chance you might need the job again, you may prefer to edit the job and simply disable it.

1. On the toolbar, expand the **More** menu and click **Replication Jobs** in the Tools list.
2. In the Jobs section, click the job you want to delete.

   The job appears highlighted in the list and its details appear on the right side of the page.
3. Click **Delete**.

   A message prompts you to confirm the deletion of the selected job.
4. Click **Delete**.

   The selected job is deleted from the jobs list.

Viewing replication job reports

Two reports—local and remote—are available for each replication job run successfully.

The local report is the transfer report from the sending system, which manages the content being transferred to the receiving system. The local report details the speed at which the files were transferred and other related details.

The remote report is the transfer report from the receiving system. This report indicates whether files were created, updated, modified, or deleted as part of the transfer.

1. On the toolbar, expand the **More** menu and click **Replication Jobs** in the Tools list.
2. In the Jobs section, click the job you want want to view.

The job appears highlighted in the list and its details appear on the right side of the page.

3. Select the desired report:

    - Click **View Local Report**.
    - Click **View Remote Report**.

    The selected report displays on the details page of the Repository Document Library.

Viewing the Repository Descriptor

There are two Repository Descriptor pages.

The Repository Descriptor (Originally Installed) page shows the details about the Alfresco repository when it was originally installed.

The Repository Descriptor (Current) page shows the details about the Alfresco repository as they currently stand. The properties on this page are the same as on the Repository Descriptor (Originally Installed) page, but it shows the most current information.

These pages are for viewing only and cannot be edited.

1. Open the Admin Console, and then click either:

    - **Repository Descriptor (Originally Installed)**
    - **Repository Descriptor (Current)**

2. View the repository properties:

    General Information

Property	What is it?
Id	This shows the ID of the repository.
Schema	This shows the schema in use.

The Version Information provides details about your Alfresco installation.

Property	What is it?
Version	This is the full version number in the format of *major*.*minor*.*revision* (*build*).
Label	This is version label.
Major	This is the major version number.
Minor	This is the minor version number.
Revision	This is the revision version number.
Build	This the build number.
Number	This is the version number of the release.

Downloading a repository dump

When you make configuration changes in the Admin Console, the current values of your running system are stored in an area in the JMX interface. When talking to Alfresco Support, it may be convenient to have a dump of these settings.

The **Repository Tools** provides a web JMX dumper so you can easily access this information.

1. Open the Admin Console, and then click **Repository Tools**.
2. Click the **Download JMX Zip Dump** link.

3. Save the file on to your computer.

   The saved `.zip` file has the name **jmxdump** appended with the current date (in the format YYYY_MM_DD).
4. Extract the file and open the `.txt` file with your preferred program.

Viewing the system runtime information

The **Runtime** page shows the properties that specify the available memory in the system.

1. Open the Admin Console, and then click **Runtime**.
2. View the runtime properties:

General property	Default setting	What is it?
Free Memory	142755664	This is the amount of free memory in bytes.
Maximum Memory	800980992	This is the maximum amount of memory that the JVM will attempt to use in bytes.
Total Memory	374071296	This is the total amount of memory in use in bytes.

Managing subscriptions to follow users

The **Subscriptions** page allows you to enable or disable the Follow feature for users to follow each other in Share.

1. Open the Admin Console, and then click **Subscriptions**.
2. On the **Subscriptions** page, click **Edit**.

   You see the **Edit: Subscriptions** page.
3. Use the **Enabled** checkbox to choose whether to enable or disable the Follow feature for all users:

   - Select the checkbox to enable subscriptions
   - Deselect the checkbox to disable subscriptions

   The **Enabled** checkbox is selected by default. This allows users to follow other users and then filter activities according to who they are following. If you disable subscriptions, users will not be able to follow users and they will not see the activities. For example, on the **My Profile** page, the **I'm Following** and **Following Me** options are not visible.
4. Click **Save** to apply the changes you have made to the properties.

   If you do not want to save the changes, click **Cancel**.

Viewing the system administration properties (Sysadmin)

The **Sysadmin** page shows the properties for server administration. These are properties that are used throughout Alfresco.

1. Open the Admin Console, and then click **Sysadmin**.
2. View the general properties:

General property	Default setting	What is it?
Repository Context	alfresco	This is the context path of the Alfresco web application. The default is `alfresco`.

General property	Default setting	What is it?
Repository Host	${localname}	This is the externally resolvable host name of the Alfresco web application. The default value is `${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.
Repository Port	8080	This is the externally resolvable port number of the Alfresco web application URL. The default is `8080`.
Repository Protocol	http	This is the protocol component of the Alfresco web application. The default is `http`.
Share Context	share	This is context path component of the Share web application URL The default is `share`.
Share Host	${localname}	This is the externally resolvable host name of the Share web application URL. The default value is `${localname}`.
Share Port	8080	This is the externally resolvable port number of the Share web application URL. The default is `8080`.
Share Protocol	http	This is the protocol to use. The default is `http`.
Allowed Users	(None)	This is a comma-separated list of users who are allowed to log in. Leave this property empty if all users are allowed to log in.
Max Users	-1	This is the maximum number of users who are allowed to log in or -1 if there is no limit.
Allowed Writes	No	This is indicates whether the repository will allow write operations (provided that the license is valid). When you set this to No, the repository is in read-only mode.

Managing workflow

Alfresco workflows run on an embedded Activiti workflow engine.

- Activiti is a lightweight workflow and Business Process Management (BPM) platform for business managers, developers, and system administrators. At its core is a BPMN 2.0 process engine for Java. Activiti is the default workflow engine in Alfresco.

- In previous versions of Alfresco, a jBPM workflow engine was used, and is still shipped as part of Alfresco.

Alfresco recommends that you use the Activiti workflow engine for all new workflows. In a new Alfresco installation, jBPM is disabled by default.

For upgrades, you can also enable jBPM so that the existing migrated workflows can continue. That is, you can run the JBPM and Activiti engines side by side. However, the jBPM workflow definitions will be hidden by default, so you cannot start new jBPM workflows.

Viewing the workflow engine properties

The **Workflow** page shows the properties for the workflow engines.

1. Open the Admin Console, and then click **Workflow**.
2. View the Activiti Engine properties:

General property	Default setting	What is it?
Enabled	Yes	This shows that the Activiti workflow engine is enabled by default. You can disable this engine using the `system.workflow.engine.activiti.enabled=false` property in the `alfresco-global.properties` file.
Number of Tasks		This specifies a count of the Activiti-defined tasks defined in the system.
Number of Definitions		This specifies a count of the Activiti definitions defined in the system.
Number of Workflows		This specifies a count of the Activiti workflows defined in the system.

3. View the jBPM Engine properties:

General property	Default setting	What is it?
Enabled	No	This shows that the jBPM workflow engine is disabled by default. You can enable this engine using the `system.workflow.engine.jbpm.enabled=true` property in the `alfresco-global.properties` file.
Definitions Visible		This property specified whether the jBPM workflow definitions are visible or not within the system. By default, you do not see jBPM workflows. You can change this setting using the `system.workflow.engine.jbpm.definitions.visible=true` property.
Number of Tasks		This specifies a count of the jBPM-defined tasks defined in the system.
Number of Definitions		This specifies a count of the jBPM definitions defined in the system.
Number of Workflows		This specifies a count of the jBPM workflows defined in the system.

Starting the Activiti workflow console

Alfresco provides the Activiti workflow console for managing Activiti based workflows and process definitions.

To start the Activiti workflow console:

1. Open the Admin Console, and click **Workflow**.

2. In the **Activiti Tools** click the **Activiti Workflow Console** link.

   The Activiti workflow console opens in a new tab in your browser.

Managing search

The Search tool in the Admin Console lets you manage the search mechanisms and settings.

Setting Lucene properties

The Lucene page shows the properties for using Lucene with Alfresco.

1. Open the Admin Console, and then click **Lucene**.
2. On the **Lucene** page, click **Edit**.

   You see the **Edit: Lucene** page.
3. Select the index recovery mode:

Property	Description
VALIDATE	The default setting. Checks the first 1000 and last 1000 transactions in the database have been indexed. If not then a message will be logged and the server will stop. Also reports any stores which do not have an index present - if found the server will not start.
AUTO	Checks the first 1000 transactions in the database have been indexed, and if not does a full index rebuild. Then checks the last 1000 transactions in the database have been indexed, and if not finds the last indexed transaction and indexes all subsequent transactions. Also reports any stores which do not have an index present - if found the server will not start.
FULL	Forces a full reindex.

4. To activate content indexing, select the **Content Indexing Enabled** checkbox.
5. To enable or disable in-transaction indexing, use the **Disable in Transaction Indexing** checkbox. This checkbox is not selected by default.
6. Change the Lucene backup properties.

Property	Example setting	What is it?
Index Backup Directory	<installLocation>/alf_data/backup-lucene-indexes	This specifies the default directory for Lucene index backups.
Backup Cron Expression	0 0 3 * * ?	Specifies a unix-like expression, using the same syntax as the cron command, that defines when backups occur.

7. Change the Lucene advanced properties.

Property	Exampl setting	What is it?
Indexing Batch Size	1000	Specifies the batch size for an index rebuild. Set it lower to trigger an index rebuild.

Property	Example setting	What is it?
Max. Atomic Transformation Time	20	Specifies that transformations of content that are likely to take longer than this time (in ms) will be done in the background. To force atomic content indexing, increase this value.
Missing Content Cron Expression	* * * * * ? 2099	Specifies a Quartz cron expression that defines when a job will run to try and re-index documents whose content has not been indexed.
Max. number of indexed tokens per document	1000	Specifies the maximum number of tokens that will be indexed per document.
Max Merged Documents	1000000	Specifies the maximum number of merged documents. This option applies to the merge process - the resulting index will be optimized.
Max. Merge Factor	6	Specifies the maximum merge factor. This option applies to the merge process - the resulting index will be optimized.

8. When you've finished changing the properties, click **Save**.

   If you do not want to save the changes, click **Cancel**.

Using search manager

The Search Manager page allows you to change the search mechanism to be used in Alfresco.

1. Open the Admin Console, and then click **Search Manager**.
2. On the **Search Manager** page, click **Edit**.

   You see the **Edit: Search Manager** page.
3. Select the search service:

Property	Description
Solr	Specifies that Solr is the default search mechanism for Alfresco.
Lucene	Specifies that Lucene is the default search mechanism for Alfresco.

4. When you've finished selecting the search service, click **Save**.

   If you do not want to save the changes, click **Cancel**.

Setting Solr properties

The Solr page shows the properties for using Solr with Alfresco.

1. Open the Admin Console, and then click **Solr**.
2. On the **Solr** page, click **Edit**.

   You see the **Edit: Solr** page.
3. To activate index tracking, select the **Tracking Enabled** checkbox.

   This option enables Solr to connect to the Alfresco server, and to track and index updates. When reading the tracking information, compare the "from Transaction" with the "to Transaction" details. If they are the same, then the tracker does not have anything to do. To check the last transaction ID in the repository, use the following query to the database:

```
select max(id) from alf_transaction;
```

   The result will match the “from Transaction” value in the log if Solr is up to date.
4. Change the General properties.

Property	Example setting	What is it?
Solr Hostname	localhost	This specifies the host name on which the Solr server is running.
Solr Port (non-SSL)	8080	This specifies the port number.
Solr SSL Port	8443	This specifies the SSL port number.

5. Change the Main store properties.

Property	Example setting	What is it?
Index Lag (Seconds)	0 s	This specifies the tracker property to set the time (in seconds) that the Solr full text index is currently behind the repository updates.
Backup Cron Expression	0 0 2 * * ?	Specifies a unix-like expression, using the same syntax as the cron command, that defines when backups occur.
Backup Location		This specifies the full-path location for the backup to be stored.

6. Change the Archive Store properties.

Property	Example setting	What is it?
Index Lag (Seconds)	0 s	This specifies the tracker property to set the time (in seconds) that the Solr full text index is currently behind the repository updates.
Backup Cron Expression	0 0 2 * * ?	Specifies a unix-like expression, using the same syntax as the cron command, that defines when backups occur.
Backup Location		This specifies the full-path location for the backup to be stored.

7. When you've finished changing the properties, click **Save**.

   If you do not want to save the changes, click **Cancel**.

Managing users

The Users tool in the Admin Console lets you create and manage the user accounts.

Creating a new user

The Admin Console lets you easily create users accounts.

1. Open the Admin Console, and then click **Users**.

   You"ll see the **User Search** page.

2. Click **New User**.

   The **New User** page appears. Fields marked with an asterisk (*) are required.

3. Complete all the required user fields.

Field	What is it?
First Name	Type the first name of the new user.
Email	Type an email address that the user will use for receiving Alfresco notification emails.
User Name	Type a user name for the new user.
Password	Type a password for the user account. ✎ Enter a minimum of five characters otherwise you'll not be able to see the **Create User** button.
Verify Password	Repeat the password. Make sure that you type the same password you typed in the **Password** field.

4. 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.

   Content quotas are disabled by default. You can change the default setting by adding the following property to the `alfresco-global.properties` file: `system.usages.enabled=true`.

6. Click **Create User**.

   ✎ The create buttons are not available until you complete all the required fields. If you didn't type in matching passwords, you'll see a message to say that the password fields do not match.

   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.

Uploading multiple users

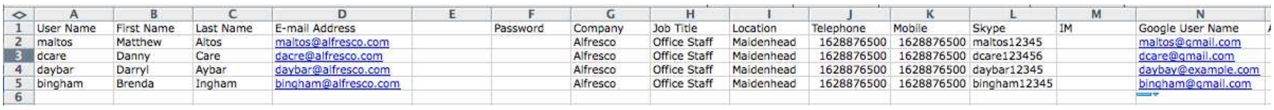
The Admin Console lets you easily upload externally created users from within a comma-separated (CSV) file.

When initially setting up the accounts for your users, it can be time consuming to create multiple users individually. Alfresco lets you create these users by uploading a file that contains the list of all your users. The file needs to contain the names and other details, separated but commas.

You can create this file, either from a text file or from a Microsoft Office spreadsheet. You need to create the file using named headings and the following order:

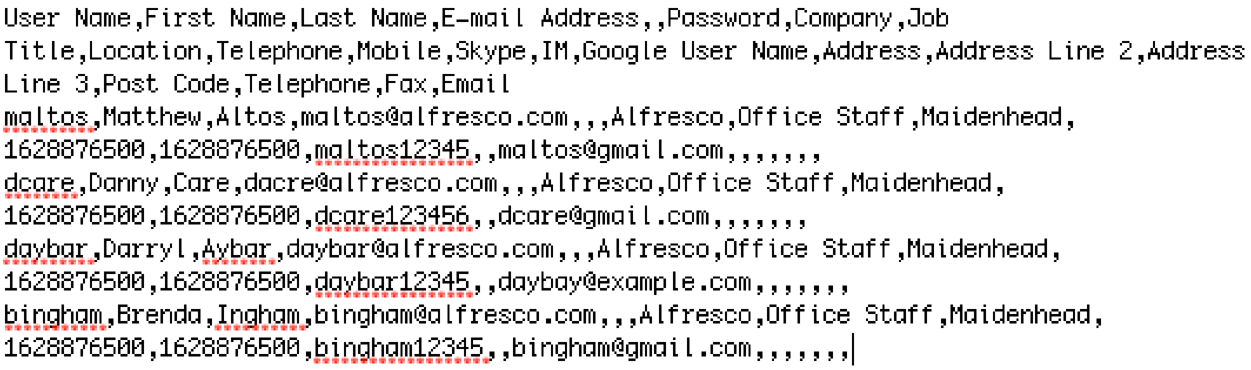
```
User Name,First Name,Last Name,E-mail Address,,Password,Company,Job
 Title,Location,Telephone,Mobile,
Skype,IM,Google User Name,Address,Address Line 2,Address Line 3,Post
 Code,Telephone,Fax,Email
```

You don't need values for all the headings for each users. For example, the following sample shows the content of a CSV file using Microsoft Excel:



◇	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	User Name	First Name	Last Name	E-mail Address		Password	Company	Job Title	Location	Telephone	Mobile	Skype	IM	Google User Name
2	maltos	Matthew	Altos	maltos@alfresco.com			Alfresco	Office Staff	Maidenhead	1628876500	1628876500	maltos12345		maltos@gmail.com
3	dcare	Danny	Care	dacre@alfresco.com			Alfresco	Office Staff	Maidenhead	1628876500	1628876500	dcare123456		dcare@gmail.com
4	daybar	Darryl	Aybar	daybar@alfresco.com			Alfresco	Office Staff	Maidenhead	1628876500	1628876500	daybar12345		daybay@example.com
5	bingham	Brenda	Ingham	bingham@alfresco.com			Alfresco	Office Staff	Maidenhead	1628876500	1628876500	bingham12345		bingham@gmail.com
6														

Save the file as a `.csv` file, which you can then upload into Alfresco.



```
User Name,First Name,Last Name,E-mail Address,,Password,Company,Job
Title,Location,Telephone,Mobile,Skype,IM,Google User Name,Address,Address Line 2,Address
Line 3,Post Code,Telephone,Fax,Email
maltos,Matthew,Altos,maltos@alfresco.com,,,Alfresco,Office Staff,Maidenhead,
1628876500,1628876500,maltos12345,,maltos@gmail.com,,,,,,,
dcare,Danny,Care,dacre@alfresco.com,,,Alfresco,Office Staff,Maidenhead,
1628876500,1628876500,dcare123456,,dcare@gmail.com,,,,,,,
daybar,Darryl,Aybar,daybar@alfresco.com,,,Alfresco,Office Staff,Maidenhead,
1628876500,1628876500,daybar12345,,daybay@example.com,,,,,,,
bingham,Brenda,Ingham,bingham@alfresco.com,,,Alfresco,Office Staff,Maidenhead,
1628876500,1628876500,bingham12345,,bingham@gmail.com,,,,,,,
```

1. Open the Admin Console, and then click **Users**.

   You'll see the **User Search** page.
2. Click **Upload User CVS File**.
3. Locate and upload the CSV file:
   a. Click the Select file(s) to upload icon.

   b. Browse for the CSV file containing the users.

      The CSV file has an extension of `.csv`.

   c. Select the file, and then click **Open**.

   d. Click **Upload File(s)**.

   The users from the CSV file are uploaded into Alfresco and you see the **Upload Results** page showing the list of user names and status. An email will be sent to the user informing them of their new Alfresco user account.

Searching for and viewing a user account

The User Search feature lets you locate any user and view that user's account information.

1. Open the Admin Console, and then click **Users**.

   You see the **User Search** page.
2. In the search box, enter the full or partial name of the user.

   The search is not case sensitive.
3. Click **Search**.

   In the results table, you can click the column headings to sort the results.

   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 a user to show the related user profile and account details.

You see the **User Profile** page. From here you can edit or delete the user account.

Editing a user account

Edit a user account to change a user's personal information, group affiliation, quota, and password.

1. Open the Admin Console, and then click **Users**.

   You'll see the **User Search** page.
2. Search for a user, and then select the 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** 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 a group. 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 keep the account but stop the user having access to the application, consider simply disabling the user account.

1. Open the Admin Console, and then click **Users**.

   You see the **User Search** page.
2. Search for a user, and then select the user.
3. On the **User Profile** page, click **Delete User**.

   A message prompts you to confirm that you want to delete the user account.
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. Open the Admin Console, and then click **Users**.

   You see the **User Search** page.
2. Search for a user, and then select the user.
3. On the **User Profile** page, click **Edit User**.

   You see the **Edit User** page.
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 shows 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. Open the Admin Console, and then click **Users**.

   You see the **User Search** page.
2. Search for a user, and then select the user.
3. On the **User Profile** page, click **Edit User**.

   You see the **Edit User** page.
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 can manage the user's membership in existing user groups. You can edit a user account at any time to add and remove the user from groups.

1. Open the Admin Console, and then click **Users**.

   You see the **User Search** page.
2. Search for a user, and then select the user.
3. On the **User Profile** page, click **Edit User**.

   You see the **Edit User** page.
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 group. Click **Add** to the right of each group you want the user to be a part of. The groups the user belongs to show 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 lets you create and manage user groups.

Browsing the user groups

The Groups page contains a multi-paned panel that lets you navigate the hierarchy of user groups.

1. Open the Admin Console, and then click **Groups**.
2. On the **Groups** page, click **Browse**.

   The left most pane displays all top-level user groups.
3. To view all groups, including the system groups, select the **Show System Groups** checkbox, and then click **Browse**.

System groups are created in the background, for example, when you create a site. You can show these groups so that you can edit the **Display Name**, add users, or delete the group.

4. Click a group to display its contents in the panel directly to the right.

   The content can be subgroups and/or individual users. The counter in the footer of the groups table indicates the number of pages within this column.
5. Browse through the group structure.

   You see a navigation path displayed at the top of the panel indicating your selections stemming from the initial pane.
6. Click any link in this path to step back to that selection.
7. Browse a different group by clicking the first link in the navigation path to return to the top-level groups, and then select a new group.

Searching for a group

The Search feature enables you to locate any user group, regardless of where it exists in the group hierarchy. Once located, you can edit or delete the group.

1. Open the Admin Console, and then click **Groups**.
2. In the search box, type the full or partial identifier, not display name, of the desired group.

   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. Open the Admin Console, and then click **Groups**.
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 fields.

Field	What is it?
Identifier	This is a name that the system uses to identify the group. Once you have created the group, you cannot change this identifier.
Display Name	This is the group name that shows in Alfresco where you manage groups and also is the name shown to members of this group.

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. Open the Admin Console, and then click **Groups**.
2. On the **Groups** page, click **Browse**.

   The leftmost pane shows all the top-level user groups.
3. Navigate the group structure or use the search feature to locate the user group you want to edit.

   The search is not case sensitive.
4. Position the cursor over the desired group to display its available actions, and then click the **Edit Group** icon.
5. Edit the group's **Display Name**.
6. Click **Save Changes**.

Deleting an existing group

Delete a user group to remove it from the system.

1. Open the Admin Console, and then click **Groups**.
2. On the **Groups** page, click **Browse**.

   The leftmost pane shows all the 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. Open the Admin Console, and then click **Groups**.
2. On the **Groups** page, click **Browse**.

   The leftmost pane shows all the 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 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 user.

   The individual user or group is added as a child to the group selected in the panel.

Managing IMAP emails

This section describes how to set the options for emails sent using IMAP.

1. Open the Admin Console, and then click **Email (IMAP)**.
2. On the **Email (IMAP)** page, click **Edit**.

   You see the **Edit: Email (IMAP)** page.
3. Set the IMAP properties:

   The IMAP properties let you configure the IMAP Home space, which is used to store user mailboxes.

Global property	Example setting	What is it?
Enabled	Enables or disables the IMAP subsystem.	
Mail from Default	This is the default email address that is displayed if you are using an email client to view a document that does not have a "From" address.	
Mail to Default	This is the default email address that is displayed if you are using an email client to view a document that does not have a "To" address.	
Extraction Enabled	If an incoming email has an attachment, should it be extracted and stored in the repository as a separate document.	
Host	This sets the host name of your IMAP server. Replace this value with the IP address (or corresponding DNS address) of your external IP interface. A value of 0.0.0.0 in Unix will make it listen on the specified port on all IP interfaces.	
Port	This is the port number, for example 143.	

4. To activate IMAP emails, select the **Enabled** checkbox.
5. Click **Save** to apply the changes you have made to the properties.

   If you do not want to save the changes, click **Cancel**.

Managing inbound emails

This section describes how to set the options for inbound emails. You need to set these properties to activate sending and receiving site invites, and also for receiving activity notification emails.

1. Open the Admin Console, and then click **Email (Inbound)**.
2. On the **Email (Inbound)** page, click **Edit**.

You see the **Edit: Email (Inbound)** page.

3. Set the general inbound email properties:

General property	Example setting	What is it?
Enabled	Yes	This is used to enable or disable the inbound email properties.
Unknown User	anonymous	This is the user name to authenticate as when the sender address is not recognized.
Allowed Senders		This 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.
Blocked Senders		This 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.
Domain	alfresco.com	This is the default domain for the email server.
Port	25	This is the default port number for the email server.

4. Click **Save** to apply the changes you have made to the properties.

   If you do not want to save the changes, click **Cancel**.

Managing outbound emails

This section describes how to set the options for outbound emails. You need to set these properties to activate sending and receiving site invites, and also for receiving activity notification emails.

1. Open the Admin Console, and then click **Email (Outbound)**.
2. On the **Email (Outbound)** page, click **Edit**.

   You see the **Edit: Email (Outbound)** page.
3. Set the general outbound email properties:

General property	Example setting	What is it?
Host	smtp.example.com	This is the host name for the outbound email server.
Port	25	This is the port number of the outbound email server.

General property	Example setting	What is it?
Protocol	smtp	This is the protocol that the email server uses.
Encoding	UTF-8	This default encoding for outbound emails.
Sender Address		This is the email address of the user account that sends the outbound emails.

4. Set the properties for email authentication:

Authentication email property	Example setting	What is it?
Authentication required	No	This enables or disables authentication for emails.
Username	anonymous	This is user name that will be used for authentication when accessing outbound emails.
Password		This is password that will be used for authentication when accessing outbound emails.

5. Set the properties for the test email:

   You can

Test email property	Example setting	What is it?
Send test message at startup	enabled	This activates the email test message.
To		This is the email address where the test message is sent.
Subject	Outbound SMTP	This is the subject title of the test email.
Message	The Outbound SMPT email subsystem is working.	This is the body text of the test email.

6. Click **Save** to apply the changes you have made to the properties.

   If you do not want to save the changes, click **Cancel**.

Managing OpenOffice

This section describes how to view the properties for the OpenOffice.

There are three OpenOffice pages in the Admin Console: OOoJodConverter, OpenOffice System, and OpenOffice.

1. Open the Admin Console, and then click an OpenOffice page.
2. View the OpenOffice JodConverter properties.

Property	Example setting	What is it?
Enabled	No	This enables or disables the OpenOffice JodConverter for transformations.
Max. Tasks per process	200	This is the maximum number of tasks.

Property	Example setting	What is it?
OpenOffice Home	/Applications/alfresco-4.1.5/ openoffice.app/Contents	This shows the directory path locations of OpenOffice.
Port Number	8100	This is the port number that OpenOffice uses.
Task Execution Timeout	120000	This is the duration in milliseconds after which a task will timeout.
Task Queue Timeout	30000	This is the duration in milliseconds after which a the task queue will timeout.

3. View the OpenOffice System properties:

Property	Example setting	What is it?
Enabled	Yes	This enables or disables the OpenOffice transformations.
Exe	*<alfresco-3.5.0-installLocation>*/openoffice/ program/soffice.bin	This shows the directory path locations of the OpenOffice that Alfresco uses for transformation.
Port	8100	This is the port number that OpenOffice uses.

Viewing the ImageMagick transformer details

The Transformer ImageMagick page shows the location of the installed version of ImageMagick that Alfresco uses for transformations. Use this page to check the location.

1. Open the Admin Console, and then click **Transformer ImageMagick**.
2. View the description the ImageMagick install directory path.

License Usage Information property	What is it?
Version information	This shows the current version number.
Available	This shows whether Alfresco has access to ImageMagick.

Viewing the pdf2swf transformer details

The Transformer pdf2swf page shows the location of the installed version of SWT Tools that Alfresco uses for transformations. Use this page to check the location.

1. Open the Admin Console, and then click **Transformer pdf2swf**.
2. View the description the SWF Tools install directory path.

License Usage Information property	What is it?
VersionString	This shows the current version number.
Available	This shows whether Alfresco has access to SWF Tools.

Configuring databases

Configuring a PostgreSQL database

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

1. Install the PostgreSQL database connector. The database connector allows PostgreSQL database to talk to the Alfresco server.
   a. Download `postgresql-9.0.802.jdbc4.jar` from the PostgreSQL download site: http://www.postgresql.org/download/.
   b. Copy the JAR file into the `<TOMCAT_HOME>/lib` directory for Tomcat 6.
2. Create a database named `alfresco`.
3. Create a user named `alfresco`.

   This user must have write permissions on all tables and sequences.
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.sample` 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. Uncomment the following properties:

   ```
   # PostgreSQL connection (requires postgresql-8.2-504.jdbc3.jar or
    equivalent)
   #
   db.driver=org.postgresql.Driver
   db.url=jdbc:postgresql://${db.host}:${db.port}/${db.name}
   ```
10. Set the other database connection properties.

    ```
    db.name=alfresco
    db.username=alfresco
    db.password=alfresco
    db.host=localhost
    db.port=5432
    db.pool.max=40
    ```

    Ensure that these database connection properties are not commented out.
11. Save the file without the `.sample` extension.
12. To allow password-authenticated connections through TCP/IP, ensure the PostgreSQL configuration file, `postgresql.conf`, contains the following:

    ```
    host all all 127.0.0.1/32 password
    ```
13. Restart the Alfresco server.

    If you receive JDBC errors, ensure the location of the PostgreSQL JDBC drivers are on the system path, or add them to the relevant `lib` directory of the application server.

Configuring an Oracle database

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

The Oracle database is case sensitive, so any configuration setting that you add into the `alfresco-global.properties` file must match the case used in Oracle.

The Oracle database must be created with the AL32UTF8 character set.

1. Install the Oracle database connector. The database connector allows Oracle database to talk to the Alfresco server.
   a. Download `ojdbc6.jar` from the Oracle download site.
   b. Copy the JAR file into the `<TOMCAT_HOME>/lib` directory for Tomcat 6.

2. Create a database named `alfresco`.
3. Create a user named `alfresco`.

   The `alfresco` user must have Connect and Resource privileges in Oracle.

   This user must have write permissions on all tables and sequences.
4. Set the new user's password to `alfresco`.
5. Ensure the `alfresco` user has the required privileges to create and modify tables.

   You can remove these privileges once the server has started, but they may also be required for upgrades.
6. Open the `<classpathRoot>/alfresco-global.properties.sample` 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. Uncomment the following properties:

   ```
   # Oracle connection
   #
   db.driver=oracle.jdbc.OracleDriver
   db.url=jdbc:oracle:thin:@${db.host}:${db.port}:${db.name}
   ```
10. Set the other database connection properties.

    ```
    db.name=alfresco
    db.username=alfresco
    db.password=alfresco
    db.host=localhost
    db.port=1521
    db.pool.max=40
    ```

    Ensure that these database connection properties are not commented out.
11. Save the file without the `.sample` extension.
12. Copy the Oracle `JDBC driver JAR` into `/lib`.
13. Restart 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: `ojdbc6.jar`. However, if you see the following error:

    ```
    java.sql.SQLException: OAUTH marshaling failure
    ```

    add the `Doracle.jdbc.thinLogonCapability=o3` parameter to `JAVA_OPTS`.

Configuring a SQL Server database

This section describes how to configure a Microsoft SQL Server database for use with Alfresco.

1. Install the Microsoft SQL Server database connector. The database connector allows SQL Server database to talk to the Alfresco server.

   a. Download `jtds-1.2.5.jar` from the Microsoft SQL Server download site.

   b. Copy the jTDS JDBC driver into the `<TOMCAT_HOME>/lib` directory for Tomcat 6.

      This release requires the jTDS driver Version 1.2.5 for compatibility with the SQL Server database.

2. Create a database named alfresco.

   Create the database using default collation settings.
3. Create a user named alfresco.

   This user must have write permissions on all tables and sequences.
4. Set the new user's password to alfresco.
5. 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.
6. Enable snapshot isolation mode with the following command:

   ```
   ALTER DATABASE alfresco SET ALLOW_SNAPSHOT_ISOLATION ON;
   ```
7. Ensure that the TCP connectivity is enabled on the fixed port number 1433.
8. Open the <classpathRoot>/alfresco-global.properties.sample file.
9. Locate the following line:

   ```
   dir.root=./alf_data
   ```
10. 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
11. Uncomment the following properties:

    ```
    # SQLServer connection
    #
    db.driver=net.sourceforge.jtds.jdbc.Driver
    db.url=jdbc:jtds:sqlserver://${db.host}:${db.port}/${db.name}
    db.txn.isolation=4096
    ```
12. Set the other database connection properties.

    ```
    db.name=alfresco
    db.username=alfresco
    db.password=alfresco
    db.host=localhost
    db.port=1433
    db.pool.max=40
    ```

    Ensure that these database connection properties are not commented out.
13. Save the file without the .sample extension.
14. Restart the Alfresco server.

    If you receive JDBC errors, ensure the location of the SQL Server JDBC drivers are on the system path, or add them to the relevant lib directory of the application server.

Configuring the MySQL database

1. Install the 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.

   a. Download mysql-connector-java-5.x.x-bin.jar from the MySQL download site: http://dev.mysql.com/.

   b. Copy the JAR file into the <TOMCAT_HOME>/lib directory for Tomcat 6.
2. Create a database named alfresco.

   If you are using MySQL and require the use of non-US-ASCII characters, you need to set the encoding for internationalization. This allows you to store content with accents in the repository. The database must be created with the UTF-8 character set and the utf8_bin collation. Although MySQL is a unicode database, and Unicode strings in

Java, the JDBC driver may corrupt your non-English data. Ensure that you keep the `?useUnicode=yes&characterEncoding=UTF-8` parameters at the end of the JDBC URL.

You also must ensure that the MySQL database is set to use UTF-8 and InnoDB. Refer to Configuration settings for using MySQL with Alfresco.

3. Create a user named `alfresco`.
4. Set the new user's password to `alfresco`.
5. Open the `<classpathRoot>/alfresco-global.properties.sample` file.
6. Locate the following line:

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

7. 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`
8. Uncomment the following properties:

```
db.driver=org.gjt.mm.mysql.Driver
db.url=jdbc:mysql://${db.host}:${db.port}/${db.name}?
useUnicode=yes&characterEncoding=UTF-8
```

9. Set the other database connection properties.

```
db.name=alfresco
db.username=alfresco
db.password=alfresco
db.host=localhost
db.port=3306
db.pool.max=40
```

Ensure that these database connection properties are not commented out.

10. (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
```

11. Save the file without the `.sample` extension.
12. Restart the Alfresco server.

If you receive JDBC errors, ensure the location of the MySQL JDBC drivers are on the system path, or add them to the relevant `lib` directory of the application server.

Optimizing MySQL to work with Alfresco

When installing MySQL, there are some settings that are required for it to work with Alfresco. This section describes the configuration settings that you should use in your MySQL instance.

The following table represents the specific settings in the MySQL configuration wizard that enable MySQL to work effectively with Alfresco.

Configuration wizard dialog	Setting for Alfresco
Server Type	Choose **Dedicated MySQL Server Machine.** The option selected determines the memory allocation.

Configuration wizard dialog	Setting for Alfresco
Database usage	Choose **Transactional Database Only**. This creates a database that uses InnoDB as its storage engine.
InnoDB Tablespace	Accept the default drive and path.
Concurrent Connections	Select **Decision Support (DSS) OLAP**. This sets the approximate number of concurrent connections to the server.
Networking and Strict Mode Options	Accept the default networking options (**Enable TCP/IP Networking**, **Port Number 3306**), and the default server SQL mode (**Enable Strict Mode**).
Character Set	Select **Best Support for Multilingualism**. This sets the default character set to be UTF-8 (set in `character-set-server`).
Security Options	Select **Modify Security Settings**. Type the root password `admin`, then retype the password.

Use the following variable setting to enable MySQL to prevent some update operations from locking database access. Add this setting to your MySQL configuration file (`/etc/my.cnf`) in the `[mysqld]` section:

```
innodb_locks_unsafe_for_binlog = 1
```

Ensure that you restart the MySQL server after adding this setting.

The effect of enabling this variable is similar to setting the transaction isolation level to `READ_COMMITTED`.

By default, table aliases are case sensitive on Unix but not on Windows or Mac OS X. Use the following variable setting to enable MySQL server to handle case sensitivity of database and table names:

```
 lower_case_table_names=1
```

Using this variable setting allows MySQL to convert all table names to lowercase on storage and lookup. This behavior also applies to database names and table aliases. This setting also prevents data transfer problems between platforms and between file systems with varying case sensitivity.

Refer to the http://dev.mysql.com/ website for more information on this variable.

Configuring a DB2 database

This section describes how to configure a DB2 database for use with Alfresco.

1. Install the DB2 database connector. The database connector allows DB2 database to talk to the Alfresco server.
   a. Obtain a copy of `db2jcc4.jar`. This should be available in the `/java` or `/jdbc` directory of your DB2 installation.
   b. Copy the JAR file into the `<TOMCAT_HOME>/lib` directory for Tomcat 6.
2. Create a database named `alfresco`.

   Create the database with a larger page size of 32K. Ensure that the database is created with the UTF-8 character set.

   If you do not create the database with these settings, you will see error SQL0286N (`sqlCode -286, sqlstate 42727`) because the schema is created for tables that do not fit the page size.

3. Ensure that the `cur_commit` database configuration parameter is set to `ON`.

   For new databases, this parameter is set to ON, by default. If you have upgraded from a previous DB2 release, you must set this parameter manually.

4. Create a user named `alfresco` and set the associated schema.

   This user must have write permissions on all tables and sequences.

   DB2 only integrates with the operating system security. You can not add a database user with a password in the DB2 database as you can with some other databases, for example the Oracle database.

5. Open the `<classpathRoot>/alfresco-global.properties.sample` file.
6. Locate the following line:

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

7. 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`
8. Uncomment the following properties:

   ```
   # DB2 connection
   #
   db.driver=com.ibm.db2.jcc.DB2Driver
   db.url=jdbc:db2://${db.host}:${db.port}/
   ${db.name}:retrieveMessagesFromServerOnGetMessage=true;
   ```

9. Set the other database connection properties.

   ```
   db.name=alfresco
   db.host=localhost
   db.port=50000
   db.pool.max=40
   ```

   Ensure that these database connection properties are not commented out.

10. Save the file without the `.sample` extension.
11. Restart the Alfresco server.

    If you receive JDBC errors, ensure the location of the DB2 JDBC drivers are on the system path, or add them to the relevant `lib` directory of the application server.

Advanced database configuration properties

This topic describes the advanced properties you can use to configure databases with Alfresco.

Alfresco Enterprise supports Oracle, Microsoft SQL Server, DB2, as well as MySQL and PostgreSQL.

As an administrator, you need to edit some of these properties to customize your database configuration. However, many properties do not need to be edited. So, the advanced database configuration properties can be categorized into two groups on the basis of their relevance:

- properties that you **SHOULD** edit
- properties that you **COULD** edit

The following table lists and describes the properties that you **SHOULD** edit:

Property name	Description	Default value
`db.txn.isolation`	The JDBC code number for the transaction isolation level, corresponding to those in the `java.sql.Connection` class. The value of -1 indicates that the database's default transaction isolation level should be used. For the Microsoft SQL Server JDBC driver, the special value of 4096 should be used to enable snapshot isolation.	`-1`
`db.pool.initial`	The number of connections opened when the pool is initialized.	`10`
`db.pool.validate.query`	The SQL query that will be used to ensure that your connections are still alive. This is useful if your database closes long-running connections after periods of inactivity.	For Oracle database, use `select 1 from dual` For MySQL database, use `select 1` For SQL Server database, use `select 1`

The following table lists and describes the properties that you **COULD** edit:

Property name	Description	Default value
`db.pool.statements.enable`	A Boolean property. When set to `true` it indicates that all precompiled statements used on a connection will be kept open and cached for reuse.	`true`
`db.pool.statements.max`	The maximum number of precompiled statements to cache for each connection. The Alfresco default is 40. Note that Oracle does not allow more that 50 by default.	`40`
`db.pool.idle`	The maximum number of connections that are not in use kept open.	`-1`
`db.pool.min`	The minimum number of connections in the pool.	`0`
`db.pool.wait.max`	Time (in milliseconds) to wait for a connection to be returned before generating an exception when connections are unavailable. A value of 0 or -1 indicates that the exception should not be generated.	`-1`
`db.pool.validate.borrow`	A Boolean property. When set to `true` it indicates that connections will be validated before being borrowed from the pool.	`true`

Property name	Description	Default value
db.pool.validate.return	A Boolean property. When set to true it indicates that connections will be validated before being returned to the pool.	false
db.pool.evict.interval	Indicates the interval (in milliseconds) between eviction runs. If the value of this property is zero or less, idle objects will not be evicted in the background.	-1
db.pool.evict.idle.min	The minimum number of milliseconds that a connection may remain idle before it is eligible for eviction.	1800000
db.pool.evict.validate	A Boolean property. When set to true it indicates that the idle connections will be validated during eviction runs.	false
db.pool.abandoned.detect	A Boolean property. When set to true it indicates that a connection is considered abandoned and eligible for removal if it has been idle longer than the db.pool.abandoned.time.	false
db.pool.abandoned.time	The time in seconds before an abandoned connection can be removed.	300

Configuring 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:

ActivitiesFeed
Handles the activities notifications.

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.

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.

Search
Handles the search mechanism for Alfresco, which can be set either to solr or lucene.

googledocs
Handles the properties for Google Docs integration.

replication
Handles the settings for the replication jobs tool.

Subscriptions
Handles the settings for the activities feeds.

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 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 OpenOffice

Within Alfresco, you can transform a document from one format to another. This feature requires you to install OpenOffice.

Alfresco supports the following OpenOffice subsystems:

- **OOoJodconverter**: This subsystem is intended for **Enterprise** users only.
- **OOoDirect**: This subsystem is intended for **Community** users only.

⚠ Do not enable both these subsystems at the same time.

OOoJodconverter

The JodConverter integration, which is a library that improves the stability and performance of OpenOffice.org (OOo) within Alfresco. 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)

If you install Alfresco using the setup wizard, this subsystem is enabled. However, if you install Alfresco manually, this subsystem is disabled, by default, and you must enable it using the following property:

```
jodconverter.enabled=true
```

OOoDirect

The direct OpenOffice integration, in which the Alfresco server manages OpenOffice directly. If you install Alfresco using the setup wizard, this subsystem is disabled. However, if you install Alfresco manually, this subsystem is enabled, by default, and you must disable it using the following property:

```
ooo.enabled=false
```

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+.

You can change OpenOffice subsystem using the following ways:

- Modifying the `alfresco-global.properties` file
- Runtime administration using your JMX client
- Share Admin Console

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.

Share Admin Console

You can also change which OpenOffice subsystem is enabled on the Share Admin Console page.

Configuring OpenOffice in the global properties file

The subsystem for OpenOffice transformations is called OOoDirect. Using this direct OpenOffice integration, the Alfresco server manages OpenOffice directly. By default, this subsystem is enabled.

1. Open the `alfresco-global.properties` file.
2. Set the `ooo.exe` property to the path of the OpenOffice installation.
3. Ensure that the following line is set to true:

   ```
   ooo.enabled=true
   ```

4. Save the file.
5. Restart the Alfresco server.

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. The following are examples of install directory paths:

- Linux: `jodconverter.officeHome=/usr/lib/openoffice`
- Mac: `jodconverter.officeHome=/Applications/OpenOffice.org.app/Contents`
- Windows: `jodconverter.officeHome=c:/Alfresco/OpenOffice.org`

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 three 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.

Differential With Removals

All users and groups are queried to determine which ones no longer exist and can be disabled or deleted locally. In order to synchronize the attributes of the remaining users and groups, a differential sync is performed so only those users and groups that have changed since the last sync are updated or added locally.

Synchronization deletion

Users and groups removed from the LDAP directory or query are only identified when synchronization is triggered by the schedule job in either full mode or differential with removals mode.

Users and groups in Alfresco created as a result of a synchronization operation are tagged with an originating zone ID. This records the ID of the authentication subsystem instance that the user or group was queried from. On synchronization with a zone, only those users and groups tagged with that zone are candidates for deletion from Alfresco. This avoids accidental deletion of built-in groups, such as ALFRESCO_ADMINISTRATORS.

When a removed user or group is detected, Alfresco will behave in one of two ways, depending on the value of the `synchronization.allowDeletions` property. When `true` (the default value), Alfresco simply deletes the user or group from the local repository. When false, the user or group is simply untagged from its zone, thus converting it to an Alfresco local user or group. A removed user also loses its memberships from any of the LDAP groups they were in, whereas, a removed group is cleared of all their members. As the user or group is retained in the Alfresco repository, this setting has the advantage that the site memberships for that user or group are remembered, should they later be reactivated.

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 ignored if `Z1` is later in the authentication chain than `Z2`
- `A` is moved to `Z1` if `Z2` does not exist in the authentication chain or `Z1` is earlier in the authentication chain and the `synchronization.allowDeletions` property is `false`.
- `A` is deleted from `Z2` and recreated in `Z1`if `Z1` is earlier in the authentication chain and the `synchronization.allowDeletions` property is `true`.

Synchronization configuration properties

The synchronization subsystem manages synchronization of Alfresco by configuring the subsystem's 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.allowDeletions

Specifies if deletion of local users and groups is allowed. See Synchronization deletion and Collision resolution for the circumstances under which this may happen. The default is true. If false, then no sync job will be allowed to delete users or groups during the handling of removals or collision resolution.

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

⚠ CIFS and NFS traffic must be targeted to a single-cluster node. In a clustered installation, concurrent writes to the same documents using file server protocols (CIFS and NFS) and other clients are not currently recommended.

✎ Functions such as NTLM SSO and CIFS authentication can only be targeted at a single subsystem instance in the authentication chain. This is a restriction imposed by the authentication protocols themselves. For this reason, Alfresco targets these ‘direct’ authentication functions at the first member of the authentication chain that has them enabled.

✎ Alfresco recommends that you implement an allowed authentication mechanism relative to the file server you are using. For more information on the different types of authentication subsystems in Alfresco and their use, see Authentication subsystem types.

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

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, use the following commands:

   ```
   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 on which Alfresco is running.

3. On Linux, use the following commands to get started, but be aware these commands will delete all existing firewall and NAT rules and could be a security risk:

   ```
   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 DNAT --to-
   destination :1445
   iptables -t nat -A PREROUTING -p tcp --dport 139 -j DNAT --to-
   destination :1139
   iptables -t nat -A PREROUTING -p tcp --dport 137:139 -j DNAT --to-
   destination :1137-1139
   iptables -t nat -A PREROUTING -p udp --dport 137:139 -j DNAT --to-
   destination :1137-1139
   ```

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

When using the subsystem extension classpath mechanism, create a Spring bean override file called `<extension>\subsystems\fileServers\default\default\file-servers-context.xml` (note that the you need to include the `default\default` part of the path, which is intentional). Add the site-specific customization of these default values to the override file.

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
`File`	File open/close
`FileIO`	File read/write

Flag	Description
`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 CIFS on Windows Server 2008 R2

The following instructions describe how to configure the Alfresco CIFS server on Windows Server 2008 R2.

To use these instructions, you must not have altered any `etc/hosts` files either on the server or client. Also you must not have modified the Windows Registry either on the server or client.

1. Install Windows Server 2008 R2 out-of-the box.

   You do not need to change the `/etc/hosts` or File and Printer Sharing configuration.

2. Configure a WINS server.

   a. If the server is a domain controller or already part of a domain, this may already be controlled by a Domain Policy.

      To install one on Windows Server 2008 R2, see the following article: http://technet.microsoft.com/en-us/library/ff710463%28WS.10%29.aspx.

   b. To manually configure an existing WINS server use the following steps:

      1. Go to **Control Panel\Network and Internet\Network and Sharing Center > Change Adapter Settings > Local Area Connection > Properties**.
      2. Select **Internet Protocol Version 4 (TCP/IPv4)** and then select **Properties**. .
      3. On the **General** tab, select **Advanced** and then select the **WINS** tab.
      4. Click **Add** and then add the IP address of the WINS server in your network and select **Enable NetBIOS over TCP/IP**.

        5. Click **OK > OK > Close**.

3. Ensure that you install Alfresco using the x64 setup wizard.

   See Installing Alfresco Enterprise on Windows.

4. Configure the Windows Server 2008 R2 firewall to create a rule to block 445.

   a. Open **Control Panel\Network and Internet\Network and Sharing Center > Windows Firewall > Advanced Settings**.

   b. Select **Inbound Rules**.

   c. On the right-side of the window, click **New Rule**.

   d. Follow the instructions on the wizard:

      1. Rule Type > Port, Next.
      2. Rule apply to "TCP", Specific Local Ports > 445, Next,
      3. Action > Block the connection, Next,
      4. Profile > Select ALL network types (Domain, Public, Private)
      5. Name > "Alfresco CIFS (Block 445)", Description the same.

   e. Select **Finish**.

5. Configure the Windows Server 2008 R2 firewall to create a rule to allow 137,138,139.

   a. Open the **Control Panel\Network and Internet\Network and Sharing Center > Windows Firewall > Advanced Settings**.

   b. Select **Inbound Rules**.

   c. On the right-side of the window, click **New Rule**.

   d. Follow the instructions on the wizard:

      1. Rule Type > Port, Next.
      2. Rule apply to "TCP", Specific Local Ports > 137,138,139, Next,
      3. Action > Allow the connection, Next,
      4. Profile > Select ALL network types (Domain, Public, Private)
      5. Name > "Alfresco CIFS (Allow 137,138,139)", Description the same.

   e. Select **Finish**.

6. Configure the client (Windows XP and Windows 7)

   a. Go to **Control Panel\Network and Internet\Network and Sharing Center > Change Adapter Settings > Local Area Connection > Properties**.

   b. Select **Internet Protocol Version 4 (TCP/IPv4)** and click **Properties**.

   c. On the **General** tab, select **Advanced** and then select the **WINS** tab.

   d. Click **Add** and then add the IP address of the WINS server in your network and select **Enable NetBIOS over TCP/IP**.

   e. Click **OK > OK > Close**.

   f. Use the `net use R: \\{HOSTNAME}A\Alfresco * /USER:admin` command to check your connection.

      If the WINS server works correctly, you are then connected to Alfresco CIFS successfully.

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.bindto
Specifies the network adapter to bind with. If the network adapter is not specified, the server will bind to all the available adapters/addresses.

ftp.dataPortFrom
Limits the data ports to a specific range of ports. This property sets the lower limit.

ftp.dataPortTo
Limits the data ports to a specific range of ports. This property sets the upper limit.

ftp.keyStore
Specifies the path to the `keystore` file for FTPS support.

ftp.trustStore
Specifies the path to the `truststore` file for FTPS support.

ftp.passphrase
Specifies the passphrase for the `keystore` and `truststore` files.

ftp.requireSecureSession
Specifies whether only secure FTPS sessions will be allowed to log in to the FTP server. To force all connections to use FTPS, set `ftp.requireSecureSession=true`.

ftp.sslEngineDebug
Enables additional debug output from the Java SSLEngine class.

If you have IPv6 enabled on your system, Alfresco automatically uses IPv6.

The FTPS support runs over the same socket as normal connections; the connection is switched into SSL mode at the request of the client, usually before the user name and password is sent. The client can switch the socket back to plain text mode using the `CCC` command.

The `ftp.keyStore`, `ftp.trustStore`, and `ftp.passphrase` values must all be specified to enable FTPS support. Only explicit FTP over SSL/TLS mode is supported. Encrypted data sessions are not supported.

To setup the `keystore` and `truststore` files, follow the instructions from the Java6 JSSE Reference Guide. This will provide the values required for the `ftp.keyStore`, `ftp.trustStore` and `ftp.passphrase` values.

Enable debug output by setting the `SSL` debug flag using `ftp.sessionDebug=SSL`, and also by enabling the `log4j.logger.org.alfresco.fileserver=debug` log4j output.

FTP advanced Spring overrides

The FTP server beans are declared in the `file-servers-context.xml` file in `<configRoot>\classes\alfresco\subsystems\fileServers\default`.

When using the subsystem extension classpath mechanism, create a Spring bean override file called `<extension>\subsystems\fileServers\default\default\file-servers-context.xml` (note that the you need to include the `default\default` part of the path, which is intentional). Add the site-specific customization of these default values to the override file.

The following properties can be overridden on the `ftpServerConfig` bean.

bindTo
Specifies the address the FTP server binds to. If it is 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

When using the subsystem extension classpath mechanism, create a Spring bean override file called `<extension>\subsystems\fileServers\default\default\file-servers-context.xml` (note that the you need to include the `default\default` part of the path, which is intentional). Add the site-specific customization of these default values to the override file.

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

Alfresco supports the ability to generate and send email to an external SMTP server.

The email service is exposed as a spring bean called mailService, which is contained within the OutboundSMTP subsystem.

Configure the Alfresco repository to send emails to an external SMTP server by overriding the default settings. Set the email property overrides in the `alfresco-global.properties` file.

The following properties can be configured for the OutboundSMTP subsystem type.

mail.host=yourmailhost.com
Specifies the host name of the SMTP host, that is, the host name or IP address of the server to which email should be sent.

mail.port
Specifies the port number on which the SMTP service runs (the default is 25). By convention, the TCP port number 25 is reserved for SMTP, but this can be changed by your email administrator.

mail.username
Specifies the user name of the account that connects to the smtp server.

mail.password
Specifies the password for the user name used in `mail.username`.

mail.encoding
Specifies UTF-8 encoding for email. Set this value to UTF-8 or similar if encoding of email messages is required.

mail.from.default
Specifies the email address from which email notifications are sent. This setting is for emails that are not triggered by a user, for example, feed notification emails. If the current user does not have an email address, this property is used for the `from` field by the `MailActionExecutor`.

mail.from.enabled
If this property is set to false, then the value set in `mail.from.default` is always used. If this property is set to true, then the `from` field may be changed. This property may be required if your email server security settings insist on matching the `from` field with the authentication details.

mail.protocol
Specifies which protocol to use for sending email. The value can be either smtp or smtps.

mail.header
Optionally specifies the content transfer encoding for the message. If specified the **Content-Transfer-Encoding** is set to the value you specify.

The following properties are for SMTP.

mail.smtp.auth
Specifies if authentication is required or not. Specifies the use of SMTPS authentication. If true, attempt to authenticate the user using the `AUTH` command. Defaults to false.

mail.smtp.timeout
Specifies the timeout in milliseconds for SMTP.

mail.smtp.starttls.enable
Specifies if the transport layer security needs to be enabled or not. Specifies the use of `STARTTLS` command. If true, enables the use of the `STARTTLS` command to switch the connection to a TLS-protected connection before issuing any login commands. Defaults to false.

mail.smtp.debug
Specifies if debugging SMTP is required or not.

The following properties are for SMTPS.

mail.smtps.starttls.enable
Specifies if the transport layer security for smtps needs to be enabled or not.

mail.smtps.auth
Specifies if authentication for smtps is required or not.

The following properties can be set to define a test message when the subsystem starts.

mail.testmessage.send
Defines whether or not to send a test message.

mail.testmessage.to
Specifies the recipient of the test message.

mail.testmessage.subject
Specifies the message subject of the test message.

mail.testmessage.text
Specifies the message body of the test message.

The following property is for setting the email site invitation behavior.

notification.email.siteinvite
You must set the outbound email configuration for Share invitations to work correctly. This property allows you to control whether or not emails are sent out for site invites. If you have not configured the outbound email properties, set this property to false.

The following example shows the properties that you need to set to configure Gmail with Alfresco. Add these properties to the `alfresco-global.properties` file. For Tomcat 6, this file is located in the `<TOMCAT_HOME>/shared/ classes` directory.

```
# Sample Gmail settings
mail.host=smtp.gmail.com
mail.port=465
mail.username=user@gmail.com
mail.password=password
mail.protocol=smtps
mail.smtps.starttls.enable=true
mail.smtps.auth=true
```

The following example shows the properties that you need to set to configure Zimbra with Alfresco. Add these properties to the `alfresco-global.properties` file. For Tomcat 6, this file is located in the `<TOMCAT_HOME>/shared/ classes` directory.

```
# Sample Zimbra settings
Not authenticated.

mail.host=zimbra.<your company>
mail.port=25
mail.username=anonymous
mail.password=
# Set this value to UTF-8 or similar for encoding of email
 messages as required
mail.encoding=UTF-8
# Set this value to 7bit or similar for Asian encoding of email
 headers as required
mail.header=
mail.from.default=<default from address>
mail.smtp.auth=false
mail.smtp.timeout=30000
```

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

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 looked up 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.

Each user has their own set of mailboxes stored within Alfresco, for example, they have their own INBOX. Users can manage emails within Alfresco ECM, and the workflow, transformation, and permissions features are available.

In addition, Share sites can be nominated as IMAP Favorites. This means that the site contents show as a set of IMAP folders. Non-favorite sites are not shown.

A metadata extractor for IMAP emails (RFC822 messages) can extract values from the contents of the email message and store the values as Alfresco properties.

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. Add the following sample configuration properties:

   ```
   imap.server.enabled=true
   imap.server.port=143
   imap.server.host=x.x.x.x
   ```

   The x.x.x.x value is the IP address (or corresponding DNS name) of your external IP interface. Do not use localhost as the imap.server.host property value.

   The default port value is 143, which will be used even if you do not add the imap.server.port property. You will need to add in this property if you require a different port number.
3. Restart the Alfresco server.

Once the Alfresco server has restarted, the new configuration will take effect. Since the IMAP server has only one instance, make your configuration changes to the alfresco-global.properties file. You can also create a file called <extension>\subsystems\imap\default\default and add your changes for the IMAP subsystem configuration.

IMAP subsystem properties

This topic describes the properties that control the IMAP subsystem.

imap.server.enabled
Enables or disables the IMAP subsystem.

imap.server.port=143
IMAP has a reserved port number of 143. You can change it using this property.

imap.server.host=<your host name>
Replace this value with the IP address (or corresponding DNS address) of your external IP interface.

You should also configure the following properties of the sysAdmin subsystem.

alfresco.host
The host name of the Alfresco URL, which is externally resolved. Use ${localname} for the locally-configured host name.

alfresco.port
The port number of the Alfresco URL, which is externally resolved. For example, `8080`

alfresco.context
The context path component of the Alfresco URL. Typically this is `alfresco`.

To configure the IMAP Home space, which is used to store user mailboxes in ARCHIVE mode, in particular the user's INBOX, use the following properties.

imap.config.home.store=${spaces.store}
Specifies the default location for the IMAP mount point. For example, `${spaces.store}`.

imap.config.home.rootPath=/${spaces.company_home.childname}
Specifies the default location for the IMAP mount point. For example, `/${spaces.company_home.childname}`.

imap.config.home.folderPath=cm:Imap Home
Specifies the QName of the default location for the IMAP mount point. For example, `cm:Imap Home`.

An IMAP message may contain a message and a set of attachments, and the IMAP server can split the attachments into separate content nodes.

imap.server.attachments.extraction.enabled
Defines whether or not attachments are extracted.

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/drop, copy/paste, and so on, from the email client. The Share sites are not shown in the ARCHIVE mode.

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.

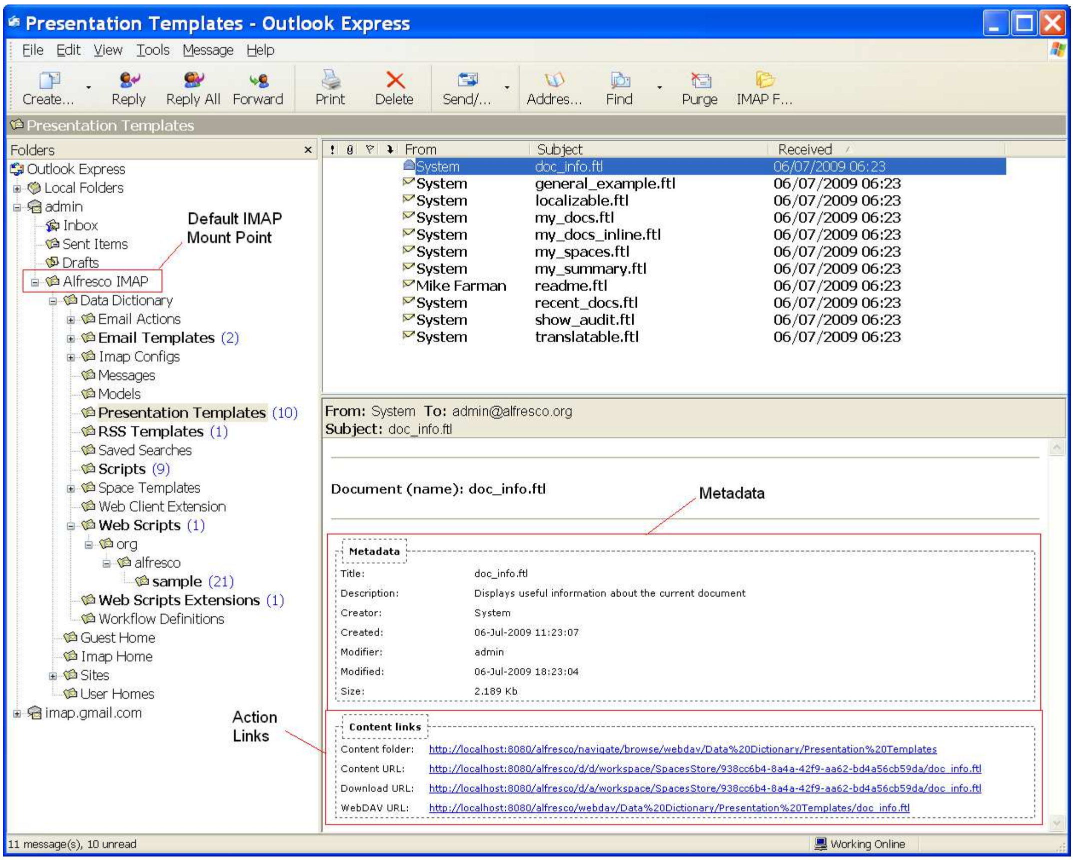
Mixed
A combination of both archive and virtual modes, that is, both document access and email management are available.

By default, a single mount point called **AlfrescoIMAP** is defined for **Company Home** and you can change it or add more mount points.

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 format request by the email client. The templates can be customized to change the metadata and actions available in the email body.



Default IMAP Mount Point

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

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

Action Links

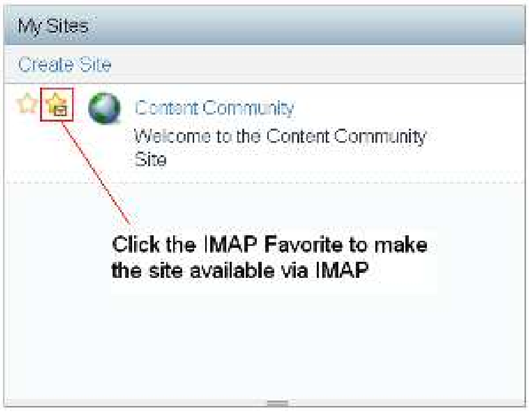
Content links

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

Marking sites as IMAP favorites

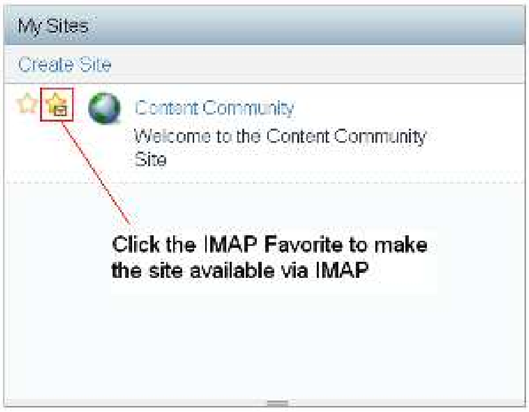
To have access to Alfresco Share sites using IMAP, the site(s) need to be added to your list of sites using Share IMAP Favorites.

1. Select **IMAP Favorites** in the Share **My Sites** dashlet on your **My Dashboard** page:



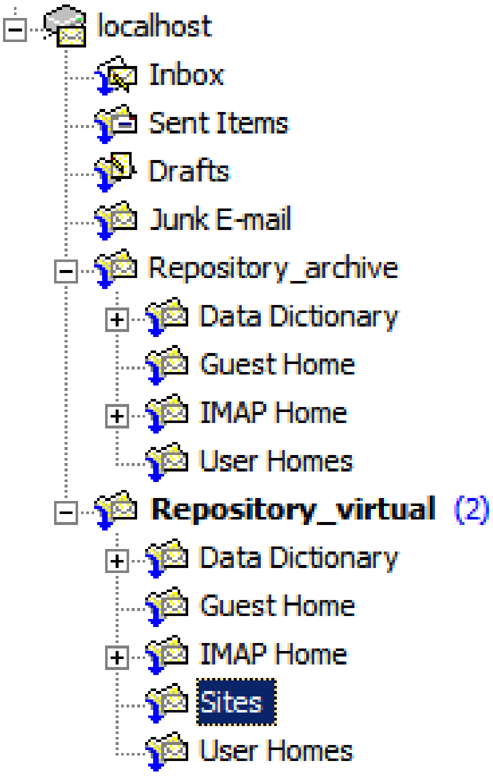
Welcome to the Content Community Site

Click the IMAP Favorite to make the site available via IMAP



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 externally resolvable host name of the UR Alfresco web application. 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 externally resolvable port number of the Alfresco web application URL. The default is `8080`.

alfresco.protocol
Specifies the protocol component of the Alfresco web application. The default is `http`.

share.context
Specifies context path component of the Share web application URL The default is `share`.

share.host
Specifies the externally resolvable host name of the Share web application URL. The default value is `${localname}`.

share.port
Specifies the externally resolvable port number of the Share web application URL. The default is `8080`.

share.protocol
Specifies the protocol to use. The default is `http`.

Configuring the repository

Tuning the JVM

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).

Hardware

Alfresco degrades gracefully on low-powered hardware, and small installations can run well on any modern server. However, for optimum performance, we recommend the following:

- Avoid 32 bit systems. Our benchmarks show a significant performance gain when using 64 bit hardware and a 64 bit JRE.

- Use a system with a clock speed above 2.5 GHz.
- Reserve enough RAM for your operating system beyond the memory required for your JVM.
- Keep search indexes on your local disk instead of on network storage.

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 allowances 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.

Virtualization

Alfresco runs well when virtualized, but you should expect a reduction in performance. When using the rough sizing requirements below, it may be necessary to allocate twice as many resources for a given number of users when those resources are virtual. Para-virtualization, or virtualized accesses to native host volumes do not require as many resources. Benchmarking your environment is necessary to get a precise understanding of what resources are required.

JVM memory and CPU hardware for multiple users

The repository L2 Cache, plus initial VM overhead, plus basic Alfresco system memory, is setup with a default installation to require a maximum of approximately 1024MB.

This means that you can run the Alfresco repository and web client with many users accessing the system with a basic single CPU server and only 1024MB 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.

Note that 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	1.5 GB JVM RAM 2x server CPU (or 1xDual-core)
For 100 concurrent users or up to 1000 casual users	1.5 GB JVM RAM 4x server CPU (or 2xDual-core)
For 200 concurrent users or up to 2000 casual users	2.5 GB JVM RAM 8x server CPU (or 4xDual-core)

JVM settings

```
-XX:MaxPermSize=256M
-Xss1024K
```

```
-Xms1G
-Xmx2G
-Dcom.sun.management.jmxremote
```

Tune the JVM using the following three steps:

1. Use as much RAM as possible for the JVM (`-Xmx32GB`).
2. Set the Permanent Generation to 256M (`-XX:MaxPermSize:256m`).
3. Do not add any other configuration settings.

To avoid memory swapping, `-Xmx` should never exceed the available RAM in the system. Remember to leave room for memory used by the operating system and other applications, like OpenOffice using JOD (JOD often uses 1GB of RAM per OO instance).

In general, if you do not give the JVM enough heap, adjusting the other JVM settings will not make any difference. Once the JVM has enough heap, you should not need to change the other JVM settings. The 1.6 JVM is generally excellent at memory optimization and is capable of functioning without adjustment.

The remaining information on this page may help in exceptional circumstances only. It is unlikely to apply to your use case, and we advise against JVM tuning beyond what has already been discussed in this section.

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 quite easy to overflow the PermGen using configuration changes or with the addition of custom extensions, and so it is recommended that you increase the PermGen to avoid OutOfMemory errors. For example, `-XX:MaxPermSize=160M` 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.

Example

The following settings are used on a high-volume clustered 64-bit, dual 2.6GHz Xeon / dual-core per CPU, 8GB RAM environment. Note the different memory ratios and try to preserve them on different environments. A minimum MaxPermSize of 128M is recommended but may sometimes require 256M.

```
-Xmx3G -XX:MaxPermSize=256M
```

Command line configuration

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

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
-Dindex.tracking.cronExpression='0/5 * * * * ?'
-Dindex.recovery.mode=AUTO
-Dalfresco.cluster.name=ALFRESCO_DEV
```

Configuring Alfresco to work with a web proxy

This topic describes the standard JVM system properties that you can use to set proxies for various protocol handlers, such as `HTTP` and `HTTPS`. These properties are used by Surf and all other parts of the system that make `http` call-outs.

All proxies are defined by a host name and a port number. The port number is optional and if not specified, a standard default port will be used.

The following two properties can be set to specify the proxy that will be used by the `HTTP` protocol handler:

System Properties	Description
`http.proxyHost`	Specifies the host name for the proxy server.
`http.proxyPort`	Specifies the port number for the proxy server. The default port number is 80.

The following two properties can be set to specify the proxy that will be used by the `HTTPS` protocol handler:

System Properties	Description
`https.proxyHost`	Specifies the host name for the proxy server when using https (http over SSL).
`https.proxyPort`	Specifies the port number for the proxy server when using https (http over SSL). The default port number is 443.

For example, the following command directs all http connections to go through the proxy server with the IP address 172.21.1.130, and the port number 8080:

```
java -Dhttp.proxyHost=172.21.1.130 -Dhttp.proxyPort=8080
```

In addition, you can also set the following non-standard properties for authenticated proxies:

Non-standard Properties	Description
`http.proxyUser`	Specifies the user name to use with an authenticated proxy used by the `HTTP` protocol handler. It should be left unset if the proxy does not require authentication.
`http.proxyPassword`	Specifies the password to use with an authenticated proxy used by the `HTTP` protocol handler. It should be left unset if the proxy does not require authentication.

Non-standard Properties	Description
`https.proxyUser`	Specifies the user name to use with an authenticated proxy used by the `HTTPS` protocol handler. It should be left unset if the proxy does not require authentication.
`https.proxyPassword`	Specifies the password to use with an authenticated proxy used by the `HTTPS` protocol handler. It should be left unset if the proxy does not require authentication.

Controlling JVM system properties

This topic describes how to control JVM system properties.

In a standard Linux/Unix installation, system properties can be specified in `-Dname=value` format (separated by spaces) in the JAVA_OPTS variable set by the script:

```
tomcat/scripts/ctl.sh
```

In a standard Windows installation, system properties can be listed in `-Dname=value` format (separated by semicolons) before `;-Dalfresco.home` in:

```
tomcat/bin/service.bat
```

Once edited, the commands:

```
tomcat/scripts/serviceinstall.bat REMOVE
tomcat/scripts/serviceinstall.bat INSTALL
```

must be run to re-register the Alfresco service with the new options.

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.

Individual cache settings

⚠ Do not directly modify this file.

Some comments may show the approximate amount of Java heap memory required by the cache. Some object references are shared by the caches, so the amount of memory used is not as high as the approximate value may suggest. Cache tracing can show which caches fill quickly for your specific server usage pattern.

Each cache is configured in an XML block with the following parameters:

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. To target specific caches, you can append the cache name or package, for example:

   ```
   org.alfresco.repo.cache.EhCacheTracerJob.org.alfresco
   ```
3. Open to the `<configRoot>\alfresco\scheduled-jobs-context.xml` file.
4. Locate the following bean:

   ```
   ehCacheTracerJob
   ```

   Override this bean to change the trigger schedule.
5. Uncomment the `scheduler` property to activate the trigger.

   When `ehCacheTracerJob` is 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.
6. To ensure that caches use statistics, set the statistics property for all caches.

   ```
   statistics="true"
   ```

   Use the cluster cache sample file or copy the default cache configuration file into the `<extensions>` directory.

Adding a MIME type

There are two files that contain the default MIME type definitions:

- `<configRoot>\classes\alfresco\mimetype\mimetype-map.xml`

- `<configRoot>\classes\alfresco\mimetype\mimetype-map-openoffice.xml`

Do not edit these files directly; instead, add new definitions to an extension file.

1. Open the `<extension>\mimetype\mimetypes-extension-map.xml.sample` file.
2. Modify the inserted MIME type to match your requirements.
3. Save the file without the `.sample` extension.
4. In the `content-services-context.xml` file, ensure that the `mimetypeConfigService` bean points to the MIME type extension file.

   This bean points to your modified file using the following line:

   ```
   <value>classpath*:alfresco/extension/mimetype/*-map.xml</value>
   ```
5. Restart the Alfresco server.

   When the Alfresco server starts, the new MIME type is registered in the system.

Configuring metadata extraction

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

For more information on the extractors, see the Javadocs for the org.alfresco.repo.content.metadata package.

Also, see the sample configurations in the following files:

- `<extension-samples>/alfresco/extension/custom-metadata-extrators-context.xml.sample`
- `<extension-samples>/alfresco/extension/wcm-xml-metadata-extracter-context.xml.sample`.

About aspects

Aspects are a fundamental concept related to content modeling in Alfresco. They allow the addition of functionality to existing content types.

Aspects use properties to enhance content types. You can attach behaviors and workflows to aspects. The following table lists the aspects available in Alfresco, along with a description.

Aspects	Description	Changes in Behavior /Share Interface
Classifiable	Enables categories to be assigned to a content item. For example, content items can be categorized under Languages, Region, Software Document Classification, and so on.	Adding Classifiable aspect displays an additional **Categories** property on the **Edit properties** page.
Complianceable	This aspect is no longer valid. For compliance-related behavior, use the Alfresco Record Management module.	

Aspects	Description	Changes in Behavior /Share Interface
Dublin Core	Enables metadata (such as publisher, contributor, identifier, and so on) to be added to a content item.	Adding Dublin Core aspect displays the following additional metadata properties on the **Edit Properties** page: • Publisher • Contributor • Type • Identifier • Source • Coverage • Rights • Subject
Effectivity	This aspect is no longer valid. For compliance-related behavior, use the Alfresco Record Management module.	
Summarizable	Enables the addition of a brief description about the content item.	Adding the Summarizable aspect displays additional **Summary** property on the **Edit Properties** page.
Versionable	Enables versioning of a content item each time it is edited (checked out and checked in or updated). In Alfresco Share, content items are versionable by default.	Adding Versionable aspect displays the version history of a content item in the **Version History** section.
Templatable	Enable template view. ✎ This aspect is only available in Alfresco Explorer.	
Emailed	Captures email-related information of the content item, if it is received as an email attachment.	Adding Emailed aspect displays additional properties (such as Originator, Addressee, Addresses, Sent Date and Subject) on the **Edit Properties** page.
Aliasable (Email)	Enables the emailing of content directly into a node.	Adding Aliasable (Email) displays additional **Aliasable** field in the **Edit Properties** page.
Inline Editable	Enables content items to be edited directly within the document library.	Adding Inline Editable aspect displays the **Inline Edit** link in the **Document Actions** section.
Google Docs Editable	This aspect is applied automatically to library items created using the Google Docs option in the **Create Content** menu. The aspect can also be added manually.	Adding Google Docs Editable aspect displays the **Edit in Google Docs** link in the **Document Actions** section.
Taggable	Enables tagging of content items using keywords. In Alfresco Share, content items are taggable by default.	Adding Taggable aspect displays the tagged keywords in the **Tags** section. You can also search for content items in the Document Library using the keywords displayed.

Aspects	Description	Changes in Behavior /Share Interface
Geographic	Enables a content item to be geographically tagged using latitude and longitude information. The location of content item is displayed as a marker on Google Maps. Click on the marker to display the **Document Details** page for that content item.	Adding Geographic aspect displays additional **Latitude** and **Longitude** properties on the **Edit Properties** page. Also, the **View on Google Maps** link is displayed in the **Document Actions** section.
EXIF	Enables capturing and viewing of additional image-related metadata of a content item. ✎ This aspect is automatically applied to an image content item.	Adding EXIF aspect displays additional information (such as Camera Model, Camera Software, Resolution Unit, and so on) about the image in the **Edit Properties** page.
Audio	Enables capturing and viewing of additional audio-related metadata of a content item. ✎ This aspect is automatically applied to an audio content item.	Adding Audio aspect displays additional information (such as Album, Artist, Composer, Track Number, and so on) about the audio file in the **Edit Properties** page.
Index Control	Enables control over how a content item is indexed.	Adding Index Control aspect displays additional Is Indexed and Is Content Indexed in the **Edit Properties** page.
Restrictable	Enables control over how a content item is distributed from the Alfresco Mobile iOS v1.5.	Adding Restrictable aspect displays additional **Offline Expires After** in the **Edit Properties** page. This defines the maximum amount of time for which the content is available in Alfresco Mobile since the user last authenticated with the Alfresco server. It's recommended that you use full hours when setting offline expiry because, in Alfresco Mobile, part-hours are rounded up to the nearest hour.

About 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 `alfresco-global.properties` file.
2. Add the following property:

   ```
   version.store.enableAutoVersioning=true
   ```

   When this property is set to false, the `VersionableAspect` will not respond to any events; even if the aspect is present, it will not create versions.
3. Save the global properties file
4. Restart the Alfresco server.

Setting up database 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.

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`.

Controlling Indexes

This section provides instructions on how to index content using the `cm:indexControl` aspect.

You can use the `cm:indexControl` aspect to set up indexes in Alfresco Share. This aspect enables you to control indexes for the control items. The aspect exposes two properties that allow configuration of indexing of nodes to which it is applied.

Property	Allowed values	Default	Description
`cm:isIndexed ((content + metadata))`	True or False	True	Controls whether the node is indexed or not.
`cm:isContentIndexed`	True or False	True	Controls whether the node content (binary) is indexed or not. Setting this to false inhibits full text indexing of the document binary.

Using this aspect you can choose to disable repository-wide indexing. This can prove useful in situations, such as bulk loading.

For more information on working with aspects, see Managing aspects section.

Setting up Alfresco authentication and security

The first time you access a vanilla Alfresco installation through Alfresco Explorer, Alfresco identifies you as a ‘guest’ user. You can identify yourself as another user by clicking the Login link and entering a new user name and password in the Login window. If you log in with the credentials of a user with administrator privileges (Alfresco uses `admin` as the default user name and password), you can use the Administration Console to create additional users and assign them passwords.

In this out-of-the-box set up, you can manage the user base and their passwords manually from within Alfresco, and unauthenticated users still have limited access as the ‘guest’ user.

From here, there are a number of common customizations you might want to make to scale up to the needs of a larger enterprise. For example, you might want to:

- Disable unauthenticated guest access
- Enable automatic sign-on using operating system credentials or a Single Sign-On (SSO) server to remove the need for a Login page
- Delegate authentication responsibility to a central directory server to remove the need to set up users manually in the Administration Console

Alfresco security

Alfresco security comprises a combination of authentication and authorization.

Authentication is about validating that a user or principal is who or what they claim to be. Alfresco normally refers to users. A user’s credentials can take many forms and can be validated in a number ways. For example, a password validated against an LDAP directory, or a Kerberos ticket validated against a Microsoft Active Directory Server.

Alfresco includes:

- An internal, password-based, authentication implementation
- Support to integrate with many external authentication environments

- The option to write your own authentication integration and to use several of these options simultaneously

Alfresco can integrate with LDAP, Microsoft Active Directory Server, the Java Authentication and Authorization Service (JASS), Kerberos, and NTLM. A user ID can also be presented as an HTML attribute over HTTPS to integrate with web-based single-sign-on solutions.

Authorization determines what operations an authenticated user is allowed to perform. There are many authorization models. Popular ones include: Role Based Access Control (RBAC), UNIX-style Access Control Lists (ACLs) and extended ACLs, Windows-style ACLs, and many more. Authorization requirements for the management of records are more detailed and include additional requirements, for example, enforcing access based on security clearance or record state.

Alfresco authorization is based on UNIX-extended ACLs. Each node in the repository has an ACL that is used to assign permissions to users and groups. Operations, such as creating a new node, describe what permissions are required to carry out the operation. ACLs are then used to determine if a given user may execute the operation based on the permissions that have been assigned directly to the user or indirectly through a group. An operation in Alfresco is invoking a method on a public service bean. For example, creating a user's home folder requires invoking methods on several public services; to create the folder, set permissions, disable permission inheritance, and so on. Each public service method invocation will check that the user is allowed to execute the method.

By convention, public service beans are the beans whose names start with capital letters, such as the NodeService. You configure the security requirements for public service beans in XML. A given method on a particular service may be available to all users, all users in a specified group, all users with a specified role, or users who have particular permissions on specified arguments to the method or its return value. In addition, for methods that return collections or arrays, their content may be filtered based on user permissions. If the authorization requirements for a method call are not met, the method call will fail and it will throw an AccessDeniedException. Non-public beans, such as nodeService, do not enforce security; use these only when the enforcement of authorization is not required.

Permission assignments are made in *Access Control Lists* (ACLs), which are lists of *Access Control Entries* (ACEs). An ACE associates an authority (group or user) with a permission or set of permissions, and defines whether the permission is denied or allowed for the authority. Every node has a related ACL. When you create a node, it automatically inherits an ACL from its parent. You can alter this behavior after node creation by breaking inheritance or modifying the ACL.

The XML configuration for permissions also defines a context-free ACL for ACEs that apply to all nodes. For example, you could use this to assign everyone Read access to all nodes regardless of what individual ACLs any node has set. (See the Permissions section in this chapter for more details on how to modify the permission model.)

```
<!-- Extension to alfresco\model\permissionDefinitions.xml -->
<globalPermission permission="Read" authority="GROUP_EVERYONE" />
```

A check that a user has Read permission for a node is done in two stages. First, the context-free ACL is checked to see if it allows access. If not, the ACL assigned or inherited by the node is checked. A user may be allowed to perform an operation because of permissions assigned to the context-free ACL, assigned to the node's ACL, inherited by the node from its parent, or a combination of all three.

Authentication subsystems

Authentication is one of the categories of the Alfresco subsystem. An authentication subsystem is a coordinated stack of compatible components responsible for providing authentication and identity-related functionality to Alfresco.

Alfresco offers multiple implementations of the 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.

An authentication subsystem provides the following functions to Alfresco:

- Password-based authentication for web browsing, Microsoft SharePoint protocol, FTP, and WebDAV
- CIFS and NFS file system authentication
- Web browser, Microsoft SharePoint protocol, and WebDAV Single Sign-On (SSO)
- User registry export (the automatic population of the Alfresco user and authority database)

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

Functions such as NTLM SSO and CIFS authentication can only be targeted at a single subsystem instance in the authentication chain. This is a restriction imposed by the authentication protocols themselves. For this reason, Alfresco targets these ‘direct’ authentication functions at the first member of the authentication chain that has them enabled.

Authentication subsystem types

A number of alternative authentication subsystem types exist for the most commonly used authentication protocols. These are each identified by a unique type name.

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`	Authentication and user registry export through the LDAP protocol (for example, OpenLDAP)	No	No	Yes

Type	Description	Single sign-on (SSO) support	CIFS authentication	User registry entry
`ldap-ad`	Authentication and user registry export from Active Directoy through the LDAP protocol	No	No	Yes
`passthru`	Authentication through a Windows domain server	Yes, NTLM	Yes	No
`kerberos`	Authentication through a Kerberos realm	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

This section describes the main components of an authentication subsystem.

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 chains

The authentication subsystem types 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.

Authentication and identity management functionality is provided by a prioritized list, or chain, of configurable subsystems. The built-in 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.

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 subsystems enable you to tie Alfresco to some of the most widely used authentication infrastructures. If you include more than one of these subsystems in the chain, you can create complex authentication scenarios.

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.

Configuring authentication

A number of examples demonstrate how to express various authentication configuration requirements in subsystem instances in the authentication chain. They also explain how the authentication chain integrates the functions of multiple subsystem instances into a more powerful conglomerate, letting you cater for even the most complex authentication scenarios. These examples demonstrate the flexibility and power of an Alfresco authentication chain. You can combine the strengths of a variety of different authentication protocols and keep the Alfresco user database synchronized almost transparently.

The authentication configuration examples adopt the following structured approach:

1. Decide the authentication chain composition (required subsystem types, instance names, order of precedence) and express this in the alfresco-global.properties file.
2. For each subsystem instance:
    a. Locate the properties files for its subsystem type. These properties files define the configurable properties for that subsystem type and their default values.
    b. Create a folder named after the subsystem instance under the Alfresco extension folders.
    c. Copy the properties files into your new folder.
    d. Edit the properties files to record the desired configuration of the subsystem instance.

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, you can 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 with the **Users** tool.

Alfresco authentication chain

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 subsystems enable you to tie Alfresco to some of the most widely used authentication infrastructures. If you include more than one of these subsystems in the chain, you can create complex authentication scenarios.

Example of disabling the Guest user login page

This section gives an example of how to set the authentication configuration in JConsole to disable the unauthenticated Guest user login using alfrescoNtlm. This example uses JConsole, however you can set the properties using the `alfresco-global.properties` file.

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 the `alfresco.log` file in the directory from where Alfresco was started. You see the following 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. Once the subsystem starts up again and reads its properties, the new settings take effect.
5. Log out from Alfresco Explorer.

   After logging out, the log in screen appears immediately, and whenever you access Alfresco, you are always directed to the login page, not the guest access.

For more information on disabling guest login for other subsystems, see the relevant authentication section. For example:

- Pass-through uses the `passthru.authentication.guestAccess` property (false by default)
- LDAP/AD uses the `ldap.authentication.allowGuestLogin` property (true by default)

Removing the login page

This section describes how to set the authentication in JConsole not to display the login page.

1. Login to Alfresco Explorer using the Administration user (`admin`).
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.

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.

Configuring Alfresco Share SSO to use NTLM

This section describes how to configure NTLM with Alfresco Share SSO.

Alfresco Share 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 web application, or it can run on a completely separate application server instance on a different machine. Share uses HTTP(S) to communicate with the configured Alfresco repository.

1. 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`

2. Edit the file, and then uncomment the following section:

```
 <!--
        SSO authentication config for Share
        NOTE: 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.alfresco.web.site.servlet.SlingshotAlfrescoConnector</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>
```

3. Change the `<endpoint-url>http://localhost:8080/alfresco/wcs</endpoint-url>` value to point to your Alfresco server location.

4. Set the `maxThreads` option in the `<TOMCAT_HOME>/conf/server.xml` file.

   ```
   <Connector port="8080" protocol="HTTP/1.1"
                connectionTimeout="20000"
                redirectPort="8443"
                maxThreads="200"
    />
   ```

   If Share and Alfresco are installed on the same Tomcat, it is important to set the `maxThreads` option to 2*(expected number of concurrent requests). This is because each Share request spawns an Alfresco request.

5. Restart Share.

If you have configured alfrescoNtlm or `passthru` in your Alfresco authentication chain and enabled SSO, NTLM will be the active authentication mechanism.

Share SSO login bypass

1. Enable NTLM SSO.
2. To login with another user to Share, use: http://localhost:8080/share/page?f=default&pt=login.
3. To logout from Share back to the NTLM, use: http://localhost:8080/share/logout.

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. Specify the file server subsystem settings in the `alfresco-global.properties` file.

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 following example defines mappings for two domains: `ALFRESCO` and `OTHERDOM`.

```
filesystem.domainMappings=ALFRESCO,OTHERDOM
filesystem.domainMappings.value.ALFRESCO.subnet=192.168.1.0
filesystem.domainMappings.value.ALFRESCO.mask=192.168.1.0
filesystem.domainMappings.value.OTHERDOM.rangeFrom=192.168.1.0
filesystem.domainMappings.value.OTHERDOM.rangeTo=192.168.1.100
```

The mask value masks the IP address to get the subnet part, and in this example, the mask value is 192.168.1.0. An alternative is to use 255.255.255.0. A value of 255.255.255.0 will get the subnet, which is then checked against the subnet value. If there were two subnets, 192.168.1.0 and 192.168.2.0, then a mask value of 255.255.255.0 and subnet value of 192.168.1.0 would only match addresses in the 192.168.1.0 range.

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
```

Example: customizing the pass-through subsystem

The authentication capabilities offered by the ldap-ad subsystem type cannot support CIFS and NTLM authentication. Instead, you would have to use form-based login for all users, and only Alfresco internal users could access CIFS. This is the compromise you would have to make if the directory server did not support any other authentication protocol. But for Active Directory, which also supports NTLM and Kerberos authentication, you can overcome this limitation by using either the Pass-through or the Kerberos subsystem types.

The Pass-through subsystem supports SSO, CIFS, and password authentication against a Windows domain server using the NTLM v1 protocol. Many prefer Kerberos for its enhanced security and you could consider it as an alternative.

1. Append an instance of `passthru` to the authentication chain.
2. Name the instance `passthru1`, and declare it by changing the `authentication.chain` property in `alfresco-global.properties` as follows:

   ```
   alfresco.authentication.authenticateCIFS=false
   ```

   Functions such as NTLM SSO and CIFS authentication can only be targeted at a single subsystem instance in the authentication chain. This is a restriction imposed by the authentication protocols themselves. For this reason, Alfresco targets these 'direct' authentication functions at the first member of the authentication chain that has them enabled. By disabling CIFS in `alfinst` earlier, `passthru1` has a chance to handle CIFS authentication for its larger user base. SSO is also left disabled in `alfinst`, which means that you can enable it in `passthru1`.

3. Stop `ldap1` from performing authentication.

   You can leave that to `passthru1`, which will be authenticating against the same server using more secure protocols. This leaves the `ldap1` user registry export capabilities still active, which you still rely on for account synchronization.
4. Edit the `ldap.authentication.active` property in the `ldap-ad-authentication.properties` file located in your `ldap1` directory as follows:

   `ldap.authentication.active=false`
5. Create the properties files to configure `passthru1`.

```
mkdir <installLocation>\tomcat\shared\classes\alfresco\extension
\subsystems\
Authentication\passthru\passthru1

cd /d <installLocation>\tomcat\shared\classes\alfresco\extension
\subsystems\
Authentication\passthru\passthru1

copy <installLocation>\tomcat\webapps\alfresco\WEB-INF\classes\alfresco
\subsystems\
Authentication\passthru\*.properties
```

After running the previous commands, two separate properties files should appear in your `passthru1` directory. These are:

- `passthru-authentication-context.properties`
- `ntlm-filter.properties`

Using a similar distinction to the `alfrescoNtlm` subsystem type, `passthru-authentication-context.properties` contains properties relating to core authentication capabilities, whereas `ntlm-filter.properties` groups those properties relating to automatic sign on. Unlike the `alfrescoNtlm` subsystem type, SSO is enabled by default in `passthru` subsystems so there is no need to edit `ntlm-filter.properties`.

The following lines show the set of properties you would need to edit and how to set them:

```
passthru.authentication.servers=DOMAIN1\\host1.com,DOMAIN2\
\host2.com,host1.com
passthru.authentication.domain=# Leave blank
passthru.authentication.guestAccess=false
passthru.authentication.defaultAdministratorUserNames=Administrator,alfresco
```

The following list is a summary of the settings that have been changed:

- `passthru.authentication.servers` — A comma-separated list of domain controller host names, each prefixed by the name of the NetBIOS domain they correspond to and a double backslash. The last member of the list should be a host name without a domain prefix, and this host will be used when a client does not include a domain name in an authentication request.
- `passthru.authentication.domain` — A property that is a less-reliable alternative to `passthru.authentication.servers` and should be left empty.
- `passthru.authentication.defaultAdministratorUserNames` — A list of user IDs who should be given Alfresco administrator privileges by default. Additional users can be made administrators by another administrator if they add those users to the ALFRESCO_ADMINISTRATORS group.

Applying the Pass-through example

Restart the Alfresco server.

The main differences to notice are:

- All Active Directory users can point their browser to the Alfresco server and be signed on automatically. (In Internet Explorer, this requires adding the Alfresco server to the Local Intranet security zone.)
- All Active Directory users can access Alfresco as a CIFS file system using their Active Directory credentials.

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 in OpenLDAP tips.

LDAP configuration properties

Both the `ldap` and `ldap-ad` subsystem types support the following configurable properties.

The defaults for `ldap` are typical for Open LDAP, and the 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 subsystem 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 user name 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.java.naming.read.timeout

Specifies the read timeout in milliseconds for LDAP operations. If Alfresco cannot get a LDAP response within that period, it aborts the read attempt. The integer should be greater than zero. If the integer is less than or equal to zero, no read timeout is specified, which is equivalent to waiting for the response infinitely until it is received.

ldap.authentication.userNameFormat

Specifies how to map the user identifier entered by the user to that passed through to LDAP.

If set to an empty string (the default for the ldap subsystem), an LDAP query involving `ldap.synchronization.personQuery` and `ldap.synchronization.userIdAttributeName` will be performed to resolve the DN from the user ID dynamically. This allows directories to be structured and does not require the user ID to appear in the DN.

If set to a non-empty value, the substring %s in this value will be replaced with the entered user ID to produce the ID passed to LDAP. This restricts LDAP user names to a fixed format. The recommended format of this value 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>`. 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 multiple 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
```

When authenticating against LDAP, users are not always in the same subtree of LDAP. In this situation, it is necessary to support authentication against multiple branches of LDAP. For example, some users who can authenticate using `cn=%s,ou=myCity,ou=myState,o=myCompany` but others can authenticate using `cn=%s,ou=ANOTHERCity,ou=myState,o=myCompany`. Set `ldap.authentication.userNameFormat` to be empty (the default), and then it will derive a query from your personQuery to look up a user by UID. This ensures that you can support users in any branch structure.

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 to be considered administrators by default. If you are using LDAP for all your users, this maps an LDAP user to be an administrator user. This administrator user can then configure the other admin users or groups by add users and/or groups to the `ALFRESCO_ADMINISTRATORS` group using the Share Admin Console or Explorer Administration Console.

If you already have a group of administrators in LDAP, you can add the required LDAP group(s)to the `ALFRESCO_ADMINISTRATORS` group. This can be set without a server restart.

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.authentication

The authentication mechanism used to connect to the LDAP server when performing user registry exports. In versions earlier than 3.4 versions, this property was the same as `ldap.authentication.java.naming.security.authentication`. The property should use one of the standard values covered in the Sun documentation http://java.sun.com/javase/6/docs/technotes/guides/jndi/spec/jndi/properties.html#pgfId=999247] or one of the values supported by the LDAP provider. Sun's LDAP provider supports the SASL mechanisms documented in http://java.sun.com/javase/6/docs/technotes/guides/jndi/jndi-ldap.html#SASL. Recommended values are:

none

Use this option if your LDAP server supports connection without a password. Set to none to allow synchronization via anonymous bind (Note that you will not also need to set the following two properties).

simple

This option is the basic LDAP authentication mechanism requiring the user name and password to be passed over the wire unencrypted. You may be able to add SSL for secure access; otherwise, use this option for testing only.

DIGEST-MD5

This option provides a more secure [ftp://ftp.isi.edu/in-notes/rfc2831.txt RFC 2831] digest authentication. With Active Directory, this requires your user accounts to be set up with reversible encryption, not the default setting.

ldap.synchronization.java.naming.security.principal

The LDAP user to connect as for the export operation, if one is required by the `ldap.synchronization.java.naming.security.authentication` authentication mechanism. This should be in the same format as `ldap.authentication.userNameFormat` but with a real user ID instead of `%s`.

This is the default principal to use (only used for LDAP sync when `ldap.synchronization.java.naming.security.authentication=simple`): `ldap.synchronization.java.naming.security.principal=cn\=Manager,dc \=company,dc\=com`

ldap.synchronization.java.naming.security.credentials

The password for this user, if required. The password for the default principal (only used for LDAP sync when `ldap.synchronization.java.naming.security.authentication=simple`) ldap.synchronization.java.naming.security.credentials=secret

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 organizational ID property in Alfresco.

ldap.synchronization.defaultHomeFolderProvider

The default home folder provider to use for people created using 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
```

Example: authentication and synchronization with one ldap-ad subsystem

This example addresses the more advanced goal of delegating 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, thus opening up Alfresco to your entire enterprise. This avoids the need for an administrator to manually set up user accounts or to store passwords outside of the directory server.

To integrate Alfresco with a directory server, you simply need to include an instance of the ldap or ldap-ad subsystem types in the authentication chain. Both subsystem types offer exactly the same capabilities and should work with virtually any directory server supporting the LDAP protocol. Their only differences are the default values configured for their attributes. The ldap type is preconfigured with defaults appropriate for OpenLDAP, whereas ldap-ad is preconfigured with defaults appropriate for Active Directory.

There are two choices in this scenario: replace or add to the authentication chain.

- Replace the authentication chain

  You could remove `alfinst` from the previous example and instead add an instance of `ldap-ad`. This would hand over all authentication responsibility to Active Directory and would mean that the built-in accounts, such as admin and guest, could not be used.

  In this scenario, it would be 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 were required. Furthermore, because ldap-ad cannot support CIFS authentication (as it requires an MD5 password hash exchange), it would rule out use of the CIFS server for all users and the CIFS server would be disabled.
- Add to the authentication chain

  You could instead supplement the existing capabilities of `alfinst` by inserting an `ldap-ad` instance before or after `alfinst` in the chain. This means that you could use the built-in accounts alongside those accounts in the directory server. Furthermore, the built-in accounts could access Alfresco through the CIFS server, since alfrescoNtlm is able to drive CIFS authentication.

  In this scenario, where you chose to position your ldap-ad instance in the chain determines how overlaps or collisions between user accounts are resolved. If an admin account existed in both Alfresco and Active Directory, then admin would be Alfresco if `alfinst` came first, or Active Directory if the ldap-ad instance came first.

This example uses an Active Directory server and configures an instance of the ldap-ad subsystem.

1. This example uses the second option to append an instance of ldap-ad to the authentication chain. This instance name is ldap1 and is declared by changing the `authentication.chain` property in the `alfresco-global.properties` file. In addition to the `authentication.chain` property, you need to add the `ntlm.authentication.sso.enabled` property to the `alfresco-global.properties` file.
2. Undo any previous modifications to alfinst and disable NTLM-based SSO.

   This is done because the ldap-ad and ldap subsystem types cannot participate in the NTLM handshake, so leaving SSO enabled would prevent any of the Active Directory users from logging in.
3. Disable SSO by opening the `alfresco-global.properties` file in a text editor and editing the `ntlm.authentication.sso. enabled` property as follows:

```
authentication.chain=alfinst:alfrescoNtlm,ldap1:ldap-ad

ntlm.authentication.sso.enabled=false

ldap.authentication.allowGuestLogin=false
ldap.authentication.userNameFormat=%s@domain.com
ldap.authentication.java.naming.provider.url=ldap://
domaincontroller.domain.com:389
ldap.authentication.defaultAdministratorUserNames=Administrator,alfresco
ldap.synchronization.java.naming.security.principal=alfresco@domain.com
ldap.synchronization.java.naming.security.credentials=secret
ldap.synchronization.groupSearchBase=ou=Security Groups,ou=Alfresco\
,dc=domain,dc=com
```

```
ldap.synchronization.userSearchBase=ou=User
 Accounts,ou=Alfresco,dc=domain,dc=com
```

The large number of configurable properties for ldap-ad may alarm you. This demonstrates the flexibility of Alfresco's LDAP infrastructure. Luckily, because ldap-ad already has sensible defaults configured for a typical Active Directory set up, there are only a few edits you must make to tailor the subsystem instance to your needs.

The following list is a summary of the settings that have been changed:

- `ldap.authentication.allowGuestLogin` — Enables / disables unauthenticated access to Alfresco
- `ldap.authentication.userNameFormat` — A template that defines how Alfresco user IDs are expanded into Active Directory User Principal Names (UPNs) containing a placeholder `%s`, which stands for the unexpanded user ID. A UPN generally consists of the user's account ID followed by an `@` sign and then the domain's UPN suffix. You can check the appropriate UPN suffix for your domain by connecting to the directory with an LDAP browser, browsing to a user account, and looking at the value of the `userPrincipalName` attribute.
- `ldap.authentication.java.naming.provider.url` — An LDAP URL containing the host name and LDAP port number (usually 389) of your Active Directory server
- `ldap.authentication.defaultAdministratorUserNames` — A list of user IDs who should be given Alfresco administrator privileges by default. Another administrator can include more users as administrators by adding those users to the ALFRESCO_ADMINISTRATORS group.
- `ldap.synchronization.java.naming.security.principal` — The UPN for an account with privileges to see all users and groups. This account is used by Alfresco to retrieve the details of all users and groups in the directory so that it can synchronize its internal user and authority database. Passwords are never compromised and remain in the directory server.
- `ldap.synchronization.java.naming.security.credentials` — The password for the previous account
- `ldap.synchronization.groupSearchBase` — The Distinguished Name (DN) of the Organizational Unit (OU) below which security groups can be found. You can determine the appropriate DN by browsing to security groups in an LDAP browser.
- `ldap.synchronization.userSearchBase` — The distinguished name (DN) of the Organizational Unit (OU) below which user accounts can be found. You can determine the appropriate DN by browsing to user accounts in an LDAP browser.

Applying the ldap-ad example

This example demonstrates how you can further delegate authentication responsibility to Active Directory, but you still do not have the automatic sign-on and CIFS browsing capabilities that are available to internal Alfresco users.

1. Restart the Alfresco server.

   If you watch the output from Tomcat in the `alfresco.log` in the installation directory, you will eventually see lines similar to the following:

   ```
   13:01:31,225 INFO
   [org.alfresco.repo.management.subsystems.ChildApplicationContextFactory]
   Starting 'Synchronization' subsystem, ID: [Synchronization, default]

   …

   13:01:49,084 INFO
   ```

```
[org.alfresco.repo.security.sync.ChainingUserRegistrySynchronizer]
Finished synchronizing users and groups with user registry 'ldap1'

13:01:49,084 INFO
[org.alfresco.repo.security.sync.ChainingUserRegistrySynchronizer]
177 user(s) and 19 group(s) processed

13:01:49,131 INFO
[org.alfresco.repo.management.subsystems.ChildApplicationContextFactory]
Startup of 'Synchronization' subsystem, ID: [Synchronization, default]
 complete
```

This is output is from the Synchronization subsystem, which is another Alfresco subsystem responsible for synchronizing the Alfresco internal user and authority database with all user registries in the authentication chain. Since the authentication chain now provides a user registry, the Synchronization subsystem has some work to do when Alfresco starts up.

2. From the example logs, notice that the Synchronization subsystem automatically created 177 users and 19 groups using attributes, such as email address and group memberships, retrieved from Active Directory through an LDAP query. This reduces the workload of the administrator user.

   The Synchronization subsystem uses an incremental timestamp-based synchronization strategy, meaning that it only queries for changes since the last synchronization run. So after the first start up, further synchronization runs can be almost instantaneous. Because synchronization runs are also triggered by a scheduled nightly job and whenever an unknown user successfully authenticates, you should find that Alfresco always stays synchronized with hardly any effort.

   Now, if you enter the Alfresco Explorer URL: http://localhost:8080/alfresco/ into your browser, you can log in using the ID and password of any of the Active Directory users.

   Passwords are validated through an LDAP bind operation on Active Directory in real time. Passwords for Active Directory users are not stored locally.

3. Navigate to a user profile.

   Notice that attributes such as email address were populated automatically from Active Directory.

Example: authentication and synchronization with two ldap-ad subsystems

This example uses one Active Directory server and shows authentication as well as user registry export (synchronization) from two ldap-ad subsystems.

The two ldap-ad subsystems used are 'ad1' and 'ad2'. Both these subsystems use the same Active Directory server but different locations within it (search bases).

1. Add the following properties to the `alfresco-global.properties` file.

```
authentication.chain=alfinst:alfrescoNtlm,ldap1:ldap-ad
ntlm.authentication.sso.enabled=false
```

2. Create the properties files to configure ad1:

```
mkdir <installLocation>\tomcat\shared\classes\alfresco\extension
\subsystems\
Authentication\ldap-ad\ad1

cd /d <installLocation>\tomcat\shared\classes\alfresco\extension
\subsystems\
Authentication\ldap-ad\ad1

copy <installLocation>\tomcat\webapps\alfresco\WEB-INF\classes\alfresco
\subsystems\
Authentication\ldap-ad\*.properties
```

A single file called `ldap-ad-authentication.properties` now appears in your ad1 directory. You can edit this file to define your LDAP set up.

The following lines show the set of properties you will typically need to edit and how you might set them for a domain controller for a fictitious domain called `domain.com` for ldap-ad subsystem, 'ad1'.

```
ldap.authentication.allowGuestLogin=false
ldap.authentication.userNameFormat=%s@domain.com
ldap.authentication.java.naming.provider.url=ldap://
domaincontroller.domain.com:389
ldap.authentication.defaultAdministratorUserNames=Administrator,alfresco
ldap.synchronization.java.naming.security.principal=alfresco@domain.com
ldap.synchronization.java.naming.security.credentials=secret
ldap.synchronization.groupSearchBase=ou=ad1,ou=Alfresco\
,dc=domain,dc=com
ldap.synchronization.userSearchBase=ou=ad1,ou=Alfresco,dc=domain,dc=com
```

3. Now, create the properties files to configure ad2:

```
mkdir <installLocation>\tomcat\shared\classes\alfresco\extension
\subsystems\
Authentication\ldap-ad\ad2

cd /d <installLocation>\tomcat\shared\classes\alfresco\extension
\subsystems\
Authentication\ldap-ad\ad2

copy <installLocation>\tomcat\webapps\alfresco\WEB-INF\classes\alfresco
\subsystems\
Authentication\ldap-ad\*.properties
```

A single file called `ldap-ad-authentication.properties` now appears in your ad2 directory. You can edit this file to define your LDAP set up.

The following lines show the set of properties you will typically need to edit and how you might set them for a domain controller for a fictitious domain called `domain.com` for ldap-ad subsystem, 'ad2'.

```
ldap.authentication.allowGuestLogin=false
ldap.authentication.userNameFormat=%s@domain.com
ldap.authentication.java.naming.provider.url=ldap://
domaincontroller.domain.com:389
ldap.authentication.defaultAdministratorUserNames=Administrator,alfresco
ldap.synchronization.java.naming.security.principal=alfresco@domain.com
ldap.synchronization.java.naming.security.credentials=secret
ldap.synchronization.groupSearchBase=ou=ad2,ou=Alfresco\
,dc=domain,dc=com
ldap.synchronization.userSearchBase=ou=ad2,ou=Alfresco,dc=domain,dc=com
```

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.

kerberos.authentication.stripUsernameSuffix
A Boolean which when true strips the @domain sufix from Kerberos authenticated usernames in CIFS, SPP, WebDAV and the Web Client when false, it enables a multi-domain customer to use the @domain sufix.

For Kerberos to work with user names that contain non-ASCII characters, add the following option to JAVA_OPTS for the Share JVM:

```
-Dsun.security.krb5.msinterop.kstring=true
```

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, and then select **Properties**.
   g. Select the **Account** tab and enable the **Do not require Kerberos preauthentication** option in the **Account Options** section.
2. Create a user account for the Alfresco SSO authentication filters, following the instructions in step one, using the full name `Alfresco HTTP` and the user login name as `alfrescohttp`.
3. Use the `ktpass` utility to generate key tables for the Alfresco CIFS and web server.

   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 -kvno 0
   ```

   ```
   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 -kvno 0
   ```

   a. Specify the `realm` as the domain in upper case. For example, if the domain is `alfresco.org` then the realm is `ALFRESCO.ORG`.
   b. `<web-server-name>` is the host name that is running the Alfresco server.
   c. Specify `<cifs-server-name>` as 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.
   d. Specify `<domain>` as the DNS domain. For example `alfresco.org`.
   e. Specify `<domainnetbios>` as the netbios name. For example `alfresco`.

   Some versions of the `ktpass` command can generate invalid `keytab` files. Download the latest version of the support 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 Windows 2003 Support Tools download.

   ```
   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 %WINDIR%\krb5.ini, where %WINDIR% is the location of your Windows directory, for example C:\Windows\krb5.ini.

```
[libdefaults]
 default_realm = ALFRESCO.ORG
 default_tkt_enctypes = rc4-hmac
 default_tgs_enctypes = rc4-hmac

[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.

   For JBoss 5, open the $JBOSS_HOME/server/default/conf/login-config.xml file. Add the following entries inside the <policy> tag:

```
 <application-policy name="Alfresco">
   <authentication>
     <login-module code="com.sun.security.auth.module.Krb5LoginModule"
 flag="sufficient"/>
   </authentication>
 </application-policy>

 <application-policy name="AlfrescoCIFS">
   <authentication>
     <login-module code="com.sun.security.auth.module.Krb5LoginModule"
 flag="required">
       <module-option name="debug">true</module-option>
       <module-option name="storeKey">true</module-option>
       <module-option name="useKeyTab">true</module-option>
       <module-option name="isInitiator">false</module-option>
       <module-option name="keyTab">C:/etc/alfrescocifs.keytab</module-
option>
       <module-option name="principal">cifs/<cifs-server-name>.domain</
module-option>
     </login-module>
   </authentication>
 </application-policy>

 <application-policy name="AlfrescoHTTP">
   <authentication>
     <login-module code="com.sun.security.auth.module.Krb5LoginModule"
 flag="required">
       <module-option name="debug">true</module-option>
       <module-option name="storeKey">true</module-option>
       <module-option name="isInitiator">false</module-option>
       <module-option name="useKeyTab">true</module-option>
       <module-option name="keyTab">C:/etc/alfrescohttp.keytab</module-
option>
       <module-option name="principal">HTTP/<web-server-name>.<domain></
module-option>
     </login-module>
   </authentication>
 </application-policy>
```

For other environments, in the `JRE\lib\security` folder (for example, `/usr/local/jdk1.6.0_03/jre/lib/security`), 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 by adding the following line to the main Java security configuration file, usually at `JRE\lib\security\java.security`.

```
login.config.url.1=file:${java.home}/lib/security/java.login.config
```

Kerberos client configuration

To make Internet Explorer negotiate Kerberos authentication, rather than NTLM, ensure that:

- Alfresco web server is in the Local Intranet security zone.

  Check **Tools > Internet Options > Security > Local Intranet**, and then ensure that a pattern matching the protocol and domain name is included, for example, http://server.com or http://*.company.com (the IP address does not work).

- Automatic log on is enabled.

  Check **Tools > Internet Options > Security > Custom Level**, and then ensure Automatic log on with the current user name and password is selected.

1. When using Firefox on Windows as your client, you need to add your Alfresco server name to the `network.negotiate-auth.trusted-uris` variable.

   Access the variable from the special URL: `about:config`.

2. When using Firefox on Linux, you 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 Share Kerberos SSO

1. Configure the Alfresco server.
2. Configure Share.
   a. Open the Share <web-extension> directory.
   b. Copy or rename the share-config-custom.xml.sample file to be called share-config-custom.xml.
   c. Replace the password, realm, and endpoint-spn option with the correct values with the correct values for the AlfrescoHTTP user (used to create the keytab files). The realm value should be capitalized.
   d. Uncomment both the <config evaluator="string-compare" condition="Remote"> sections.

```
   <!-- example port config used to access remote Alfresco server
 (default is 8080) -->

   <config evaluator="string-compare" condition="Remote">
      <remote>
         <endpoint>
            <id>alfresco-noauth</id>
            <name>Alfresco - unauthenticated access</name>
            <description>Access to Alfresco Repository WebScripts that
 do not require authentication</description>
            <connector-id>alfresco</connector-id>
            <endpoint-url>http://localhost:8080/alfresco/s</endpoint-
url>
            <identity>none</identity>
         </endpoint>

         <endpoint>
            <id>alfresco</id>
            <name>Alfresco - user access</name>
            <description>Access to Alfresco Repository WebScripts that
 require user authentication</description>
            <connector-id>alfresco</connector-id>
            <endpoint-url>http://localhost:8080/alfresco/s</endpoint-
url>
            <identity>user</identity>
         </endpoint>

         <endpoint>
            <id>alfresco-feed</id>
            <name>Alfresco Feed</name>
            <description>Alfresco Feed - supports basic HTTP
 authentication via the EndPointProxyServlet</description>
            <connector-id>http</connector-id>
            <endpoint-url>http://localhost:8080/alfresco/s</endpoint-
url>
            <basic-auth>true</basic-auth>
            <identity>user</identity>
         </endpoint>

         <endpoint>
            <id>activiti-admin</id>
            <name>Activiti Admin UI - user access</name>
            <description>Access to Activiti Admin UI, that requires
 user authentication</description>
            <connector-id>activiti-admin-connector</connector-id>
            <endpoint-url>http://localhost:8080/alfresco/activiti-
admin</endpoint-url>
            <identity>user</identity>
         </endpoint>
      </remote>
   </config>


   <!--
        Overriding endpoints to reference an Alfresco server with
 external SSO enabled
        NOTE: If utilising a load balancer between web-tier and
 repository cluster, the "sticky
              sessions" feature of your load balancer must be used.
        NOTE: If alfresco server location is not localhost:8080 then
 also combine changes from the
              "example port config" section below.
        *Optional* keystore contains SSL client certificate + trusted
 CAs.
        Used to authenticate share to an external SSO system such as
 CAS
        Remove the keystore section if not required i.e. for NTLM.

        NOTE: For Kerberos SSO rename the "KerberosDisabled" condition
 above to "Kerberos"

        NOTE: For external SSO, switch the endpoint connector to
 "AlfrescoHeader" and set
              the userHeader to the name of the HTTP header that the
 external SSO
              uses to provide the authenticated user name.
   -->

   <config evaluator="string-compare" condition="Remote">
      <remote>
         <keystore>
             <path>alfresco/web-extension/alfresco-system.p12</path>
             <type>pkcs12</type>
             <password>alfresco-system</password>
         </keystore>

         <connector>
            <id>alfrescoCookie</id>
            <name>Alfresco Connector</name>
            <description>Connects to an Alfresco instance using
 cookie-based authentication</description>
```

```
  <class>org.alfresco.web.site.servlet.SlingshotAlfrescoConnector</
class>
         </connector>

         <connector>
            <id>alfrescoHeader</id>
            <name>Alfresco Connector</name>
            <description>Connects to an Alfresco instance using header
 and cookie-based authentication</description>

  <class>org.alfresco.web.site.servlet.SlingshotAlfrescoConnector</
class>
            <userHeader>SsoUserHeader</userHeader>
         </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>
```

e. Locate the `<!-- Kerberos settings -->` section and replace `condition=KerberosDisabled` with `condition=Kerberos`.

```
<!-- Kerberos settings -->
   <!-- To enaable kerberos rename this condition to "Kerberos" -->
   <config evaluator="string-compare" condition="Kerberos"
 replace="true">
      <kerberos>
```

f. In the (Sun Java) `jre/lib/security/java.login.config` file, add a new section:

```
ShareHTTP {
   com.sun.security.auth.module.Krb5LoginModule required
   storeKey=true
   useKeyTab=true
   keyTab="/etc/keys/alfrescohttp.keytab"
   principal="HTTP/madona.example.foo";
};
```

g. Restart the Alfresco server.

3. Configure Active Directory.

   a. Modify the `alfrescohttp` user created during the Alfresco Kerberos setup.

   b. In the user **Delegation** tab, tick the **Trust this user for delegation to any service (Kerberos only)** check box.

      If you do not see the delegation tab, follow the `Allow a user to be trusted for delegation for specific services` instruction on the Microsoft http://technet.microsoft.com website.

   c. If you cannot see the **Delegation** tab, do one or both of the following:

      - Register a Service Principal Name (SPN) for the user account with the Setspn utility in the support tools on your CD. Delegation is only intended to be used by service accounts, which should have registered SPNs, as opposed to a regular user account which typically does not have SPNs.

- Raise the functional level of your domain to Windows Server 2003.

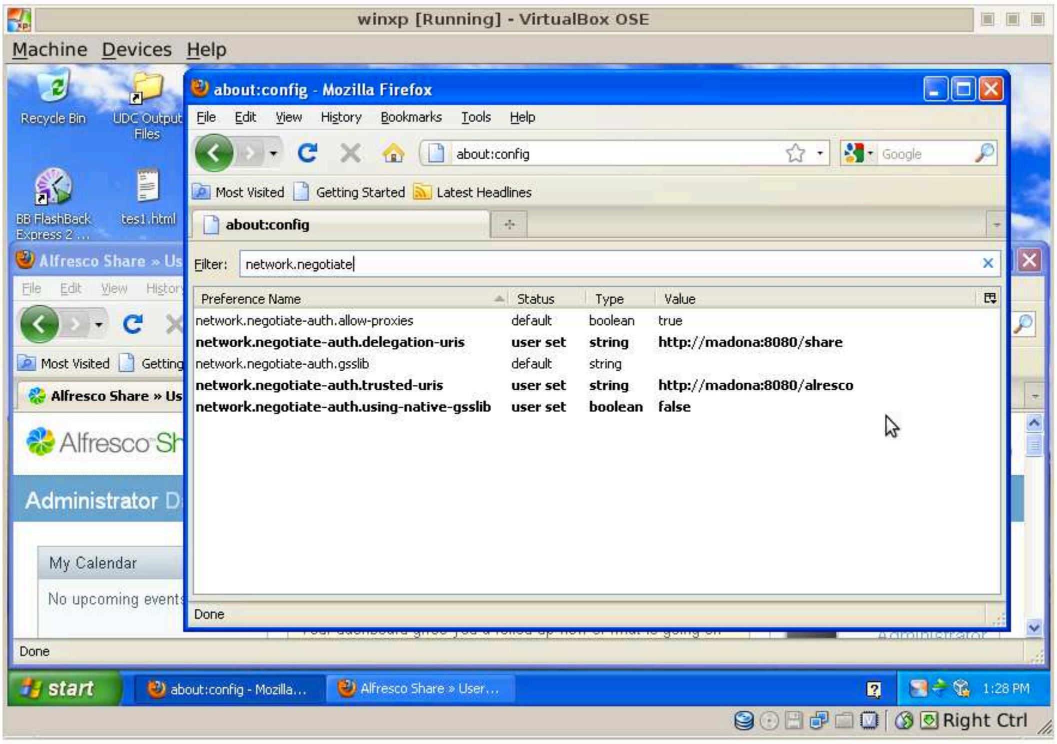
To raise the domain functional level:

1. Open **Active Directory Domains and Trusts**.
2. In the console tree, right-click the domain for which you want to raise functionality, and then click **Raise Domain Functional Level**.
3. In **Select an available domain functional level**, do one of the following:
   - To raise the domain functional level to Windows 2000 native, click **Windows 2000 native**, and then click **Raise**.
   - To raise domain functional level to Windows Server 2003, click **Windows Server 2003**, and then click **Raise**.

4. Configure the client.
   a. For Windows client configuration, Internet Explorer configured as described in Kerberos client configuration should work without modifications.
   b. To ensure that Firefox works with Windows on the share URL with Kerberos SSO, modify the following variables in the about:config special URL:

```
network.negotiate-auth.delegation-uris
network.negotiate-auth.trusted-uris
network.negotiate-auth.using-native-gsslib
```

For example:


winxp [Running] - VirtualBox OSE
Machine Devices Help
about:config - Mozilla Firefox
File Edit View History Bookmarks Tools Help
about:config

Filter: network.negotiate

Preference Name	Status	Type	Value
network.negotiate-auth.allow-proxies	default	boolean	true
network.negotiate-auth.delegation-uris	**user set**	**string**	**http://madona:8080/share**
network.negotiate-auth.gsslib	default	string	
network.negotiate-auth.trusted-uris	**user set**	**string**	**http://madona:8080/alresco**
network.negotiate-auth.using-native-gsslib	**user set**	**boolean**	**false**

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.

Activating external authentication makes Alfresco accept external authentication tokens, make sure that no untrusted direct access to Alfresco's HTTP or AJP ports is allowed.

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 Alfresco Share to use an external SSO

This section describes how to configure Alfresco Share to use an external Single Sign on (SSO).

Alfresco Share can be configured to accept a user name from an HTTP header provided by an external authentication system. Complete the following steps to achieve this configuration.

Activating external authentication makes Alfresco accept external authentication tokens. Ensure that no untrusted direct access to Alfresco's HTTP or AJP ports is allowed.

1. Configure Share.
   a. Open the Share `<web-extension>` directory.
   b. Copy or rename the `share-config-custom.xml.sample` file to be called `share-config-custom.xml`.
   c. Uncomment the `<config evaluator="string-compare" condition="Remote">` section.

2. Change the connector used by the endpoint in the second section to use `alfrescoHeader` rather than `alfrescoCookie`.
3. Set the name of the header used by the external SSO in the `userHeader` element of the `alfrescoHeader` connector.
4. Change the `endpoint-url` value to point to your Alfresco Server location.

```
  <!-- example port config used to access remote Alfresco server
(default is 8080)
    -->

  <config evaluator="string-compare" condition="Remote">
     <remote>
        <endpoint>
           <id>alfresco-noauth</id>
           <name>Alfresco - unauthenticated access</name>
           <description>Access to Alfresco Repository WebScripts that do
not
           require authentication
       </description>
           <connector-id>alfresco</connector-id>
           <endpoint-url>http://localhost:8080/alfresco/s</endpoint-url>
           <identity>none</identity>
        </endpoint>

        <endpoint>
           <id>alfresco</id>
           <name>Alfresco - user access</name>
           <description>Access to Alfresco Repository WebScripts that
                        require user authentication
       </description>
           <connector-id>alfresco</connector-id>
           <endpoint-url>http://localhost:8080/alfresco/s</endpoint-url>
           <identity>user</identity>
        </endpoint>

        <endpoint>
           <id>alfresco-feed</id>
           <name>Alfresco Feed</name>
           <description>Alfresco Feed - supports basic HTTP
authentication via
                        the EndPointProxyServlet</description>
           <connector-id>http</connector-id>
           <endpoint-url>http://localhost:8080/alfresco/s</endpoint-url>
           <basic-auth>true</basic-auth>
           <identity>user</identity>
        </endpoint>

        <endpoint>
           <id>activiti-admin</id>
           <name>Activiti Admin UI - user access</name>
           <description>Access to Activiti Admin UI, that requires user
                        authentication</description>
           <connector-id>activiti-admin-connector</connector-id>
           <endpoint-url>http://localhost:8080/alfresco/activiti-admin
           </endpoint-url>
           <identity>user</identity>
        </endpoint>
     </remote>
   </config>


  <!--
       Overriding endpoints to reference an Alfresco server with
external SSO
       enabled
```

```
        NOTE: If utilising a load balancer between web-tier and
 repository
        cluster,the "sticky sessions" feature of your load balancer must
 be used.

        NOTE: If alfresco server location is not localhost:8080 then also
 combine
        changes from the"example port config" section below.
        *Optional* keystore contains SSL client certificate + trusted
 CAs.
        Used to authenticate share to an external SSO system such as CAS
        Remove the keystore section if not required i.e. for NTLM.

        NOTE: For Kerberos SSO rename the "KerberosDisabled" condition
 above to
        "Kerberos"

        NOTE: For external SSO, switch the endpoint connector to
 "AlfrescoHeader"
              and set the userHeader to the name of the HTTP header
              that the external SSO uses to provide the authenticated
 user name.
   -->

   <config evaluator="string-compare" condition="Remote">
      <remote>
         <keystore>
             <path>alfresco/web-extension/alfresco-system.p12</path>
             <type>pkcs12</type>
             <password>alfresco-system</password>
         </keystore>

         <connector>
            <id>alfrescoCookie</id>
            <name>Alfresco Connector</name>
            <description>Connects to an Alfresco instance using cookie-
based
                          authentication
            </description>

  <class>org.alfresco.web.site.servlet.SlingshotAlfrescoConnector</class>
         </connector>

         <connector>
            <id>alfrescoHeader</id>
            <name>Alfresco Connector</name>
            <description>Connects to an Alfresco instance using header
 and
             cookie-based authentication
            </description>

  <class>org.alfresco.web.site.servlet.SlingshotAlfrescoConnector</class>
            <userHeader>SsoUserHeader</userHeader>
         </connector>

         <endpoint>
            <id>alfresco</id>
            <name>Alfresco - user access</name>
            <description>Access to Alfresco Repository WebScripts that
 require user
             authentication
            </description>
            <connector-id>alfrescoHeader</connector-id>
            <endpoint-url>http://localhost:8080/alfresco/wcs</endpoint-
url>
            <identity>user</identity>
            <external-auth>true</external-auth>
         </endpoint>
```

```
    </remote>
</config>
```

5. Set the `external.authentication.proxyHeader` property to the same value as the `userHeader` value. By doing this, you have configured both Share and the Alfresco repository to use the same HTTP header value.

   ```
   external.authentication.proxyHeader=SsoUserHeader
   ```

6. Restart Share.

You have now configured Alfresco Share to use an external SSO.

Authorities

Authorities are people (or persons) or groups.

A group may contain people or other groups as members. The authorities assigned to a user at any time are the userName from their associated Person node, all of the groups in which the user is a direct or indirect member, and any appropriate dynamic authorities. Dynamic authorities are used for internal roles.

Dynamic authorities and roles

Alfresco uses some custom roles. To implement a custom role, you create a dynamic authority for that role and assign global permissions to it. The Alfresco internal roles have not been assigned any object-specific rights. The internal roles are:

- ROLE_ADMINISTRATOR is assigned to the default administrators for the configured authentication mechanisms or members of the administration groups defined on the AuthorityServiceImpl bean. This role has all rights.
- ROLE_OWNER is assigned to the owner of a node. If there is no explicit owner, this role is assigned to the creator. This role has all rights on the owned node.
- ROLE_LOCK_OWNER is assigned to the owner of the lock on a locked node. This supports a lock owner’s right to check in, cancel a check out, or unlock the node.

The Alfresco Explorer and Alfresco Share currently support the assignment of permissions only to the owner role. You can use such things as the Java API and scripting to make other assignments.

Hierarchical and zoned roles may be added to Alfresco in the future to avoid the hidden group implementation for true roles.

People and users

When a user logs in, Alfresco validates the user’s identifier and password. Alfresco uses the identifier to look up the appropriate person details for the user, using the `userName` property on the Person type. You can configure this look-up to be case sensitive or case insensitive. The `userName` property on the matching Person node is used as the actual user authority; it may differ in case from the user identifier presented to the authentication system. After the Person node look-up, Alfresco is case sensitive when matching authorities to permissions, group membership, roles, and for all other authorization tests.

Any user, who authenticates by any mechanism, must have an associated person node in Alfresco. Person nodes may be:

- Explicitly created
- Created on demand with some default entries
- Created from LDAP synchronization

Person nodes are explicitly created when using the administration pages of the Alfresco Explorer and Alfresco Share web clients to manage users.

By default, person nodes are auto-created if not present. If an external authentication system is configured, such as NTLM, when any user authenticates, an appropriate person node may not exist. If a person node does not exist and auto-creation is enabled, a person node will then be created using the identifier exactly as presented by the user and validated by the authentication system. The auto-created Person node's userName will have the same case as typed by the user. LDAP synchronization will create person nodes with the userName, as provided from the LDAP server.

It is possible that LDAP synchronization can change the userName associated with a Person node. For example, this can happen with a system that uses NTLM authentication, LDAP synchronization, or a system that creates person nodes on demand, or uses case-insensitive authentication. For example, Andy could log in as “Andy” and the associated Person node is created with the userName “Andy.” Later, the LDAP synchronization runs and changes the userName to “andy”.

From version 3.2, changes to Person node userNames will cause updates to other related data in Alfresco, such as ACL assignment.

Groups

Groups are collections of authorities with a name and display name.

Groups may include other groups or people. You can include a group in one or more other groups, as long as this inclusion does not create any cyclic relationships.

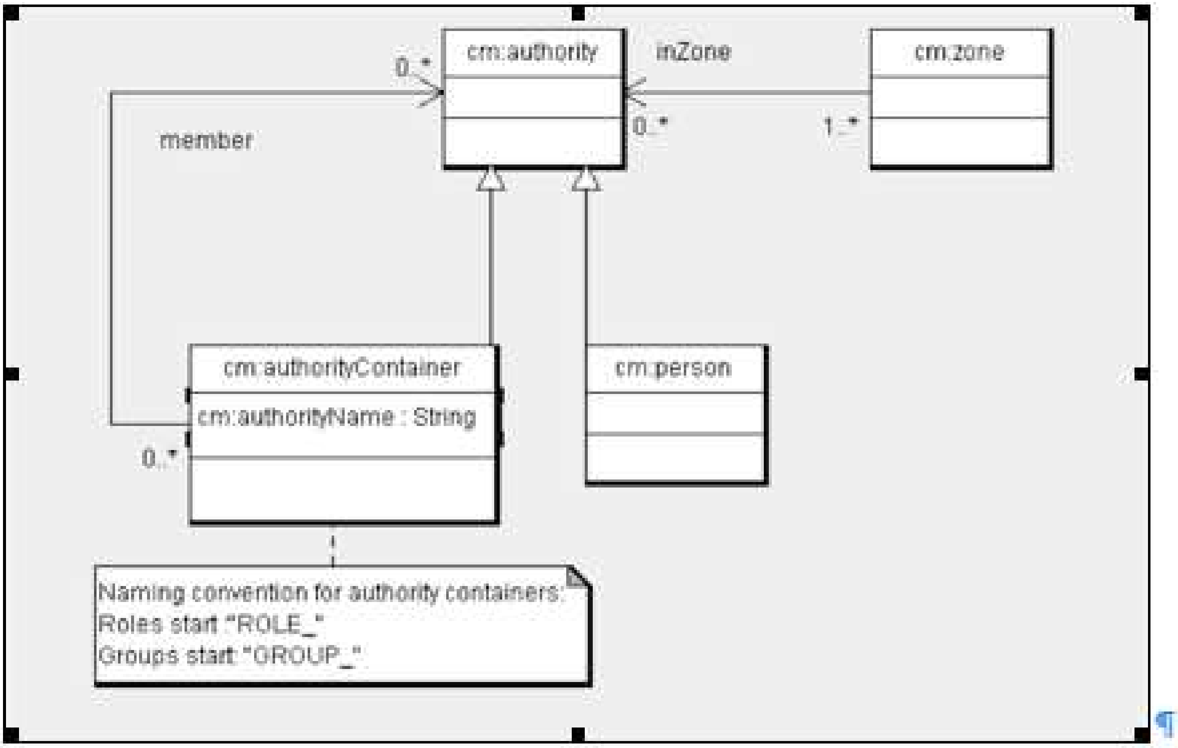
Zones

All person and group nodes are in one or more zones. You can use zones for any partitioning of authorities. For example, Alfresco synchronization uses zones to record from which LDAP server users and groups have been synchronized. Zones are used to hide some groups that provide Role Based Access Control (RBAC) role-like functionality from the administration pages of the Alfresco Explorer and Alfresco Share web clients. Examples of hidden groups are the roles used in Alfresco Share. Only users and groups in the default zone are shown for normal group and user selection on the group administration pages. Zones cannot be managed from the administration pages of Alfresco Explorer and Alfresco Share.

Zones are intended to have a tree structure defined by naming convention. Zones are grouped into two areas: Application-related zones and authentication-related zones.

Within a zone, a group is considered to be a root group if it is not contained by another group in the same zone.

Alfresco uses a model for persisting people, groups, and zones. A Person node represents each person, and an AuthorityContainer represents groups, which can be used for other authority groupings such as roles. AuthorityContainer and Person are sub-classes of Authority and as such can be in any number of Zones.



Application-related zones

Application-related zones, other than the default, hide groups that implement RBAC like roles. Application zones, by convention, start APP. and include:

- APP.DEFAULT is for person and group nodes to be found by a normal search. If no zone is specified for a person or group node, they will be a member of this default zone.
- APP.SHARE is for hidden authorities related to Alfresco Share.
- APP.RM will be added for authorities related to RM.

Authorization-related zones

Zones are also used to record the primary source of person and group information. They may be held within Alfresco or some external source. While authorities can be in many zones, it makes sense for an authority to be in only one authentication-related zone.

- AUTH.ALF is for authorities defined within Alfresco and not synchronized from an external source. This is the default zone for authentication.
- AUTH.EXT.<ID> is for authorities defined externally, such as in LDAP.

Defining permissions

Permissions and their groupings are defined in an XML configuration file. The default file is found in the distribution configuration directory as `<installLocation>\tomcat\webapps\alfresco\WEB-INF\classes\alfresco\model\permissionDefinitions.xml`. This configuration can be replaced or extended and has a structure as described in `<installLocation>\tomcat\webapps\alfresco\WEB-INF\classes\alfresco\model\permissionSchema.dtd`.

The following example uses the permission definitions related to the Ownable aspect.

```
<!-- ============================================ -->
   <!-- Permissions associated with the Ownable aspect -->
   <!-- ============================================= -->

   <permissionSet type="cm:ownable" expose="selected">

      <!-- Permission control to allow ownership of the node to be taken from
 others -->
      <permissionGroup name="TakeOwnership" requiresType="false"
 expose="false">
        <includePermissionGroup permissionGroup="SetOwner" type="cm:ownable" />
      </permissionGroup>

      <permissionGroup name="SetOwner" requiresType="false" expose="false"/>

      <!-- The low level permission to control setting the owner of a node -->
      <permission name="_SetOwner" expose="false" requiresType="false">
        <grantedToGroup permissionGroup="SetOwner" />
        <requiredPermission on="node" type="sys:base" name="_WriteProperties" /
>
      </permission>

</permissionSet>
```

Permissions and permission groups are defined in a permission set, which is a sub-element of the permissions root element. A permission set is associated with a type or aspect and applies only to that type and sub-types, or aspect and sub-aspects.

A permission has a name. By convention, the names of permissions start with an underscore character. They may be exposed in the administration pages of Alfresco Explorer and Alfresco Share but, usually, are not. A permission, in its definition, may be granted to any number of permission groups. This means that those permission groups will include the permission. The permission may require that the type or aspect specified on the permission set be present on the node. If a permission is associated with an aspect and the requiresType property is set to true then if that aspect is not applied to a node, the permission does not apply to that node either. If an aspect-related permission definition has the requiresType property set to false, the permission applies to any node, even if the aspect has not been applied to the node.

An aspect can be applied at any time and there are no restrictions as to which aspects can be applied to a type. A permission may also require other permissions be tested on the same node, its children, or its parent. In the example, _SetOwner requires _WriteProperties. This means you cannot set ownership on a node if you are not allowed to write to its properties. You can also use this to check that all children can be deleted before deleting a folder, or to enforce that you can only read nodes for which you can read all the parents; neither are normally required in Alfresco. The configuration to do this is present in the standard configuration file but is commented out. The _DeleteNode permission definition (as shown in the following code snippet) is an example. If permission A requires permission B and this requirement is implied (by setting the implies attribute of the requiredPermission element to true), assigning an authority permission A will also give them permission B (as opposed to checking they have permission B).

```
<permission name="_DeleteNode" expose="false" >
    <grantedToGroup permissionGroup="DeleteNode" />
    <!-- Commented out parent permission check ...
    <requiredPermission on="parent" name="_ReadChildren" implies="false"/>
    <requiredPermission on="parent" name="_DeleteChildren" implies="false"/>
    <requiredPermission on="node" name="_DeleteChildren" implies="false"/>
     -->
    <!-- Recursive delete check on children -->
    <!--  <requiredPermission on="children" name="_DeleteNode" implies="false"/
>  -->
</permission>
```

Permissions are normally hidden inside permission groups. Permission groups are made up of permissions and other permission groups. By convention, each permission has a related permission group. Permission groups can then be combined to make other permission groups.

As for permissions, a permission group may be exposed by the administration pages of Alfresco Explorer and Alfresco Share and may require the presence of a type or aspect to apply to a particular node. In addition, a permission group may allow full control, which grants all permissions and permission groups. As a type or aspect may extend another, a permission group defined for a type or aspect can extend one defined for one of its parent types and be assigned more permissions, include more permission groups, or change what is exposed in the administration pages of the Alfresco Explorer and Alfresco Share web clients.

It is unusual to extend or change the default permission model unless you are adding your own types, aspects, and related public services or you wish to make minor modifications to the existing behavior. The following code snippets show how to extend and replace the default permission model.

```
<bean id='permissionsModelDAO'
class="org.alfresco.repo.security.permissions.impl.model.PermissionModel" init-
method="init">
        <property name="model">
<-- <value>alfresco/model/permissionDefinitions.xml</value> -->
<value>alfresco/extension/permissionDefinitions.xml</value>
        </property>
        <property name="nodeService">
            <ref bean="nodeService" />
        </property>
        <property name="dictionaryService">
            <ref bean="dictionaryService" />
        </property>
</bean>
```

The preceding code example shows how to replace the default permission model with one located in the `alfresco/extension` directory. The following code snippet shows how to extend the existing model.

```
<bean id="extendPermissionModel" parent="permissionModelBootstrap">
   <property name="model" value="alfresco/extension/
permissionModelExtension.xml" />
</bean>
```

Access Control Lists

An Access Control List (ACL) is an ordered list of Access Control Entries (ACEs). An ACE associates a single authority to a single permission group or permission, and states whether the permission is to be allowed or denied. All nodes have an associated ACL. There is one special, context-free, ACL defined in the XML configuration to support global permissions. An ACL specifies if it should inherit ACEs from a parent ACL. The parent ACL is associated with the primary parent node. When a new node is created it automatically inherits all ACEs defined on the parent within which it is created. Linking a node to a secondary parent has no effect on ACE inheritance; the node will continue to inherit permission changes from its primary parent (defined when it was first created).

By default, ACL inheritance is always from the primary parent. The underlying design and implementation does not mandate this. ACL inheritance does not have to follow the parent child relationship. It is possible to change this through the Java API but not via the administration pages of Alfresco Explorer and Alfresco Share.

There are several types of ACL defined in ACLType. The main types are:

- DEFINING
- SHARED
- FIXED
- GLOBAL

A node will be associated with an ACL. It will have a DEFINING ACL if any ACE has been set on the node. DEFINING ACLs include any ACEs inherited from the node's primary parent and above, if inheritance is enabled. All DEFINING ACLs are associated with one SHARED ACL. This SHARED ACL includes all the ACEs that are inherited from the DEFINING ACL. If the primary children of a node with a DEFINING ACL do not themselves have any specific ACEs defined then they can be assigned the related SHARED ACL. For the primary children of a node with a SHARED ACL that also have no specific ACEs set they can use the same SHARED ACL. A single SHARED ACL can be associated with many nodes. When a DEFINING ACL is updated, it will cascade update any related ACLs via the ACL relationships rather than walk the node structure. If a DEFINING ACL inherits ACEs, then these will come from the SHARED ACL related to another DEFINING ACL.

ACLs and nodes have two linked tree structures.

FIXED ACLs are not associated with a node but found by name. A node ACL could be defined to inherit from a fixed ACL. A GLOBAL ACL is a special case of a FIXED ACL with a well known name. It will be used to hold the global ACE currently defined in XML.

ACEs comprise an authority, a permission, and a deny/allow flag. They are ordered in an ACL.

ACL ordering and evaluation

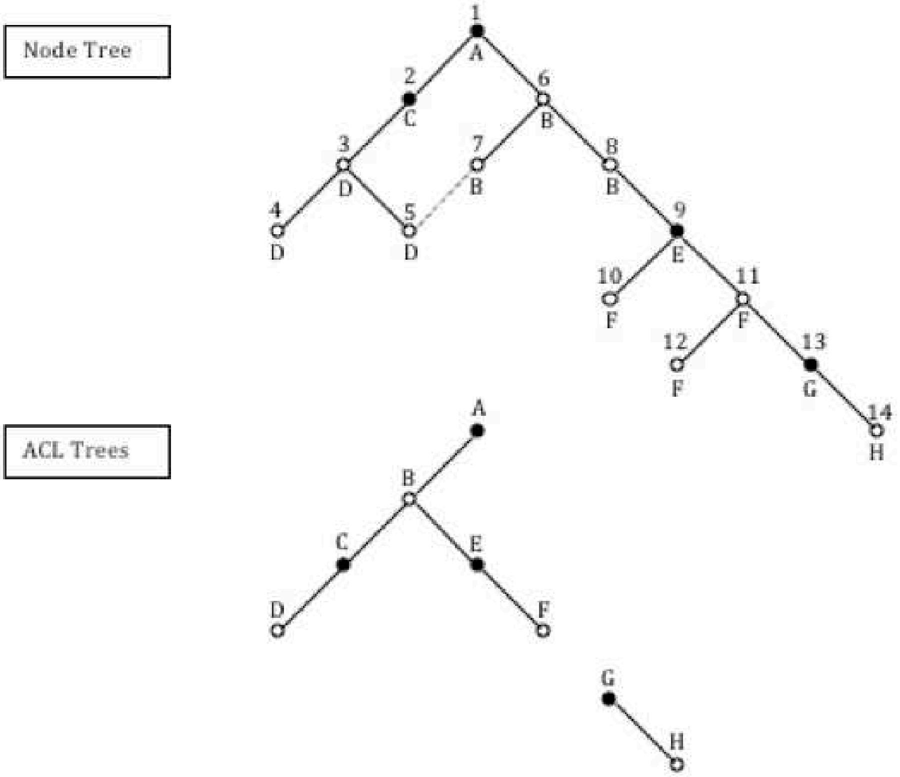
The ACEs within an ACL are ordered and contain positional information reflecting how an ACE was inherited. DEFINING ACLs have entries at even positions; SHARED ACLs have entries at odd positions. For a DEFINING ACL, any ACEs defined for that ACL have position 0, any inherited from the parent ACL have position two, and so on. For a SHARED ACL, ACEs defined on the ACL from which it inherits will have position one.

When Alfresco makes permission checks, ACEs are considered in order with the lowest position first. Deny entries take precedence over allow entries at the same position. The default configuration is that “any allow allows”. Once a deny entry is found for a specific authority and permission combination, any matching ACE, at a higher position from further up the inheritance chain, is denied. A deny for one authority does not deny an assignment for a different authority. If a group is denied Read permission, a person who is a member of that group can still be assigned Read permission using another group or directly with their person userName. However, if an authority is granted Read (made up of ReadContent and ReadProperties) and the same authority denied ReadContent, they will just be granted ReadProperties permission. The administration pages of Alfresco Explorer and Alfresco Share do not expose deny.

You can alter the configuration to support “any deny denies”.

An ACL example

This example relates a tree of nodes to two corresponding trees of ACLs. The nodes in the node tree are identified by number and are shown filled in black if they have any ACEs set, or white/clear if not. Primary child relationships are drawn as black lines and secondary child relationships as dashed lines. ACLs in the ACL trees are identified by letter, DEFINING ACLs are shown filled in black, and SHARED ALCs are shown as clear. Under each node on the node tree the related ACL is referenced.



The table describes the ACEs in each ACL and their position.

Table 1: ACL formats

ACL format	Authority	Permission	Allow/Deny	Position
ACL A (Defining, no inheritance)	All	Read	Allow	0
ACL B (Shared, inherits from ACL A)	All	Read	Allow	1
ACL C (Defining, inherits from ACL B)	All	Read	Allow	2
	ROLE_OWNER	All	Allow	0
	GROUP_A	Write	Allow	0
	GROUP_A	CreateChildren	Allow	0
ACL D (Shared, inherits from ACL C)	ALL	Read	Allow	3
	ROLE_OWNER	All	Allow	1
	GROUP_A	Write	Allow	1
	GROUP_A	CreateChildren	Allow	1
ACL E (Defining, inherits from ACL B)	All	Read	Allow	2
	Andy	All	Allow	0
	Bob	Write	Allow	0
	Bob	WriteContent	Deny	0
ACL F (Shared, inherits from ACL E)	All	Read	Allow	3
	Andy	All	Allow	1
	Bob	Write	Allow	1
	Bob	WriteContent	Deny	1
ACL G (Defining, no inheritance)	Bob	All	Allow	0

ACL format	Authority	Permission	Allow/Deny	Position
ACL H (Shared, inherits from ACL G)	Bob	All	Allow	1

ACL A, and any ACL that inherits from it, allows Read for everyone (All) unless permissions are subsequently denied for everyone (All). If ACL A is changed, all the ACLs that inherit from ACL A in the ACL tree will reflect this change. In the example, nodes 1-12 would be affected by such a change. Nodes 13 and 14 would not inherit the change due to the definition of ACL G.

ACL C adds Contributor and Editor permissions for any authority in GROUP_A.

The GROUP_ prefix is normally hidden by the administration pages of Alfresco Explorer and Alfresco Share.

Anyone in GROUP_A can edit existing content or create new content. The owner ACE means that anyone who creates content then has full rights to it. The ACE assignment for owner is not normally required as all rights are given to node owners in the context-free ACL defined in the default permission configuration.

ACL E adds some specific user ACEs in addition to those defined in ACL A. As an example, it allows Bob Write but also denies WriteContent. Write is made up of WriteContent and WriteProperties. Bob will only be allowed WriteProperties.

ACL G does not inherit and starts a new ACL tree unaffected by any other ACL tree unless an inheritance link is subsequently made.

If a new node was created beneath node 13 or 14 it would inherit ACL H. If a new node was created beneath nodes 1, 6, 7, or 8 it would inherit ACL B.

If a node that has a shared ACL has an ACE set, a new defining ACL and a related shared ACL are inserted in the ACL tree. If a defining ACL has all its position 0 ACEs removed, it still remains a defining ACL: There is no automatic clean up of no-op defining ACLs.

Modifying access control

Modifying access control may involve:

- Changing the definition of existing security interceptors to check for different conditions
- Adding new public services and related security interceptors
- Defining new types and aspects and their related permissions
- Adding new definitions to the security interceptor by implementing an ACEGI AccessDecisionVoter and/or AfterInvocationProvider (in extreme cases)

A few constraints and design patterns should be observed when modifying access control. Permissions apply to the node as whole. In particular, the same Read rights apply to all properties and content. You should check that methods can be executed and not that a user has a particular permission. The access control restrictions for a public service method may change. Follow the design pattern to implement RBAC roles.

When modifying access control, do not try to split ReadProperties and ReadContent. This does not make sense for search. A node and all of its properties, including content, are indexed as one entity. Splitting the evaluation of access for content and properties is not possible. Search would have to apply both criteria so as to not leak information. Other services, such as copy, may not behave as expected or may produce nodes in an odd state.

Permissions are assigned at the node level, not at the attribute level. Again, this makes sense with the search capabilities. Search results need to reflect what the user performing the search can see. It makes sense that all properties have the same Read access as the node, as nodes are indexed for searching and not individual properties. Applying Read ACLs at the property level would require a change to the indexing implementation or a complex post analysis to work out how nodes were found by the search. If not, the values of properties could be deduced by how a readable node was found from a search on restricted properties.

Fine grain attribute permissions could be implemented by using children nodes to partition metadata. Queries would have to be done in parts and joined by hand, as there is no native support for SQL-like join.

Check that method execution is allowed and not that the user has a fixed permission. Rather than checking for Read permission in code, check that the appropriate method can be called using the PublicServiceAccessService bean. This avoids hard coding to a specific permission implementation and is essential if you intend to mix records management and the content repository. The access restrictions for public service methods may change. The PublicServiceAccessService bean allows you to test if any public service method can be invoked successfully with a given set of arguments. It checks all the entry criteria for the method and, assuming these have not changed, the method can be called successfully. The method call may still fail if the conditions for the returned object are not met or some security configuration has changed, such as an ACE is removed, a user is removed from a group, or the method fails for a non-authorization reason.

For those coming from an RBAC background, Alfresco has roles in the RBAC sense only for limited internal use. To implement RBAC in Alfresco, use zoned groups. These groups will not appear in the administration pages of the Alfresco Explorer and Alfresco Share web clients as normal groups (unless you also add them to the APP.DEFAULT zone) but can be used to assign users and groups to roles. This approach has been taken in Alfresco to support roles in Alfresco Share. To map RBAC terminology to Alfresco: operations map to method calls on public service beans, objects map to method arguments including nodes (folders, documents, and so on). Users and permissions/privileges map directly. Alfresco allows the assignment of permissions to users or groups.

By default, the owner of an object can manage any aspect of its ACL. Users with ChangePermissions rights for a node can also change its ACL. If users have the ability to alter the ACL associated with an object, they can allow other users to do the same. There is no restriction on the permissions they may assign. The Alfresco model supports liberal discretionary access control with multi-level grant. A user who can grant access can pass on this right without any restriction. In addition, anyone who can change permissions can carry out the revocation of rights: it is not restricted to the original granter. Normally, when someone can perform an operation you would not expect it is because they own the node and therefore have all permissions for that node.

Public services

Security is enforced around public services. Web services, web scripts, Alfresco Explorer and Alfresco Share, CIFS, WebDAV, FTP, CMIS, and more, all use public services, and therefore include security enforcement. Public services are defined in <installLocation>\tomcat\webapps \alfresco\WEB-INF\classes\alfresco\public-services-context.xml.

Access control allows or prevents users or processes acting on behalf of a user, from executing service methods on a particular object by checking if the current user, or any of the authorities granted to the current user, has a particular permission or permission group, or that the user has a particular authority.

For example, on the NodeService bean, the `readProperties` method checks that the current user has Read permission for the node before invoking the method and returning the node's properties. On the SearchService query method, the results are restricted to return only the nodes for which a user has Read permission.

Public services configuration

Security is enforced in the Spring configuration by defining proxies for each internal service implementation and adding a method interceptor to enforce security for each public service proxy. These interceptors also have other roles. When a method is called on a public service, the security interceptor is called before the method it wraps. At this stage, the interceptor can examine the function arguments to the method and check that the user has the appropriate rights for each argument in order to invoke the method. For example, a method delete(NodeRef nodeRef) exists on the node service. The security interceptor can see the nodeRef argument before the underlying delete(...) method is called. If configured correctly, the interceptor could check that the current user has "Delete" permission for the node. If they do not have the permission, a security exception is raised. If all the entry criteria are met, the method goes ahead.

In a similar manner, after a method has executed the interceptor can examine the returned object and decide if it should return it to the caller. For example, a search method could return a list of nodes. The security interceptor could filter this list for only those nodes for which the current user has Read permission.

It is also possible to configure a method so that it can be called by all users, only by users with the admin role, or only by specific users or groups. This can also be enforced by the security method interceptor.

Access control interceptor definitions for public services are included in <installLocation>\tomcat \webapps\alfresco\WEB-INF\classes\alfresco\public-services-security-context.xml along with any other supporting beans. This configuration file also defines the location from which the permission model is loaded. The interceptors are wired up to the public services in <installLocation>\tomcat \webapps\alfresco\WEB-INF\classes\alfresco\public-services-context.xml. The public services are the only Spring beans to have access control.

Method-level security definition

The beans required to support Spring ACEGI-based security around method invocation are defined in <installLocation>\tomcat\webapps\alfresco\WEB-INF\classes\alfresco\public-services-security-context.xml. This configures two Alfresco-specific beans: A voter that can authorize method execution based on the permissions granted to the current user for specific arguments to the method, and an after invocation provider to apply security to objects returned by methods.

Method access is defined in the normal ACEGI manner with some additions.

For the following information detailing preconditions and postconditions, these factors are all relevant:

<authority>
Represents an authority (user name or group).

<#>
Represents a method argument index.

<permission>
Represents the string representation of a permission.

Preconditions take one of the following forms:

ACL_METHOD.<authority>
Restricts access to the method to those with the given authority in Alfresco. This could be a user name or group. Dynamic authorities are not supported.

ACL_NODE.<#>.<permission>
Restricts access control to users who have the specified permission for the node at the identified argument. If the argument is a NodeRef, it will be used; if it is a StoreRef, the root node for the store will be used; if it is a ChildAssociationRef, the child node will be used.

ACL_PARENT.<#>.<permission>
Restricts access control to users who have the specified permission for the parent of the node on the identified argument. If the argument is a NodeRef, the parent of the node will be used; if it is a ChildAssociationRef, the parent node will be used.

ROLE
Checks for an authority starting with ROLE_.

GROUP
Checks for an authority starting with GROUP_.

If more than one ACL_NODE.<#>.<permission> , ACL_PARENT.<#>.<permission>, or ACL_METHOD.<permission> entry is present, then all of the ACL_NODE and ACL_PARENT permissions must be present and any one of the ACL_METHOD restrictions, if present, for the method to execute.

Post-conditions take the forms:

AFTER_ACL_NODE.<permission>
Similar to ACL_NODE.<#>.<permission> but the restriction applies to the return argument.

AFTER_ACL_PARENT.<permission>
Similar to ACL_PARENT.<#>.<permission> but the restriction applies to the return argument.

The support return types are:

- StoreRef
- ChildAssociationRef
- Collections of StoreRef, NodeRef, ChildAssociationRef, and FileInfo
- FileInfo
- NodeRef
- Arrays of StoreRef, NodeRef, ChildAssociationRef, and FileInfo
- PagingLuceneResultSet
- QueryEngineResults
- ResultSet

The post-conditions will create access denied exceptions for return types such as NodeRef, StoreRef, ChildAssociationRef, and FileInfo. For collections, arrays, and result sets, their members will be filtered based on the access conditions applied to each member.

Continuing the example from the permissions defined for the Ownable aspect, the definition for the security interceptor for the related OwnableService is shown in the following code snippet.

```
<bean id="OwnableService_security"

 class="org.alfresco.repo.security.permissions.impl.acegi.MethodSecurityInterceptor">
   <property name="authenticationManager"><ref bean="authenticationManager"/></
property>
   <property name="accessDecisionManager"><ref local="accessDecisionManager"/
></property>
   <property name="afterInvocationManager"><ref local="afterInvocationManager"/
></property>
   <property name="objectDefinitionSource">
     <value>

 org.alfresco.service.cmr.security.OwnableService.getOwner=ACL_NODE.0.sys:base.ReadPrope

 org.alfresco.service.cmr.security.OwnableService.setOwner=ACL_NODE.0.cm:ownable.SetOwne

 org.alfresco.service.cmr.security.OwnableService.takeOwnership=ACL_NODE.0.cm:ownable.Ta
 org.alfresco.service.cmr.security.OwnableService.hasOwner=ACL_NODE.0.sys:base.ReadPrope
      org.alfresco.service.cmr.security.OwnableService.*=ACL_DENY
      </value>
    </property>
</bean>
```

Security for the four methods on the OwnableService is defined. To invoke the OwnableService getOwner() method on a node, the invoker must have permission to read the properties of the target node. To set the owner of a node, a user must have been explicitly assigned the SetOwner permission or have all rights to the node. A user may have all rights to a node via the context-free ACL or be assigned a permission, which grants all permission or includes SetOwner. With the default configuration, a user will own any node they create and therefore be able to give ownership to anyone else and possibly not have the right to take ownership back.

The last entry catches and denies access for any other method calls other than those listed. If any additional methods were added to this service and no security configuration explicitly defined for the new methods, these methods would always deny access.

Implementation and services

The following key services are involved in access control:

- PersonService
- AuthorityService
- PermissionService
- OwnableService

The PersonService and the AuthorityService are responsible for managing authorities. The PermissionService is responsible for managing ACLs and ACEs and for checking if a user has been assigned a permission for a particular node. The OwnableService manages object ownership and is used in evaluation the dynamic ROLE_OWNER authority.

The protection of public services methods is implemented using Spring method interceptors defined as part of the related ACEGI 0.8.2 security package. The Alfresco implementation adds new implementations of the ACEGI interfaces AccessDecisionVoter and AfterInvocationProvider, which support the configuration elements that have already been described (for example, ACL_NODE.<#>.<permission>). These extension classes make use of the key services.

Person service

The PersonService interface is the API by which nodes of the person type, as defined in <installLocation>\tomcat\webapps\alfresco\WEB-INF\classes\alfresco\model\contentModel.xml, should be accessed.

The PersonService is responsible for all of the following:

- Obtaining a reference to the Person node for a given user name
- Determining if a person entry exists for a user
- Potentially creating missing people entries with default settings on demand
- Supplying a list of mutable properties for each person
- Creating, deleting, and altering personal information

The beans to support the PersonService and its configuration can be found in <installLocation>\tomcat\webapps\alfresco\WEB-INF\classes\alfresco\authentication-services-context.xml. The principle configuration options are around how people are created on demand if users are managed via NTLM or some other external user repository.

Authority service

The AuthorityService is responsible for:

- Creating and deleting authorities
- Querying for authorities
- Structuring authorities into hierarchies
- Supporting queries for membership
- Finding all the authorities that apply to the current authenticated user
- Determining if the current authenticated user has admin rights
- Managing zones and the assignment of authorities to zones

The authority service does not support user authentication or user management. This is done by the AuthenticationService. Person nodes are managed via the PersonService.

The default implementation allows a list of group names to define both administration groups and guest groups. Each authentication component defines its own default administrative user(s), which can also be set explicitly. The default service is defined in <installLocation>\tomcat\webapps\alfresco\WEB-INF\classes\alfresco\authority-services-context.xml.

Permission service

The PermissionService is responsible for:

- Providing well known permissions and authorities
- Providing an API to read, set, and delete permissions for a node
- Providing an API to query, enable, and disable permission inheritance for a node
- Determining if the current, authenticated user has a permission for a node

The PermissionService interface defines constants for well-known permissions and authorities.

The default implementation coordinates implementations of two service provider interfaces: a ModelDAO and a PermissionsDAO. A permission is simply a name scoped by the fully qualified name of the type or aspect to which it applies. The beans are defined and configured in <installLocation>\tomcat\webapps\alfresco\WEB-INF\classes\alfresco\public-services-security-context.xml. This file also contains the configuration for security enforcement.

The ModelDAO interface defines an API to access a permissions model. The default permission model is in XML and defines permission sets, and their related permission groups and permissions. Global permissions are part of the permission model. There may be more than one permission model defined in XML; they are in practice merged into one permission model. A module can extend the permission model.

The available permissions are defined in the permission model. This is defined in <installLocation>\tomcat\webapps\alfresco\WEB-INF\classes\alfresco\model\permissionDefinitions.xml. This configuration is loaded in a bean definition in <installLocation>\tomcat\webapps\alfresco\WEB-INF\classes\alfresco\public-services-security-context.xml. This file also defines global permissions. The definition file is read once at application start-up. If you make changes to this file, you will have to restart the repository in order to apply the changes.

Ownable service

The idea of file ownership is present in both UNIX and Windows. In Alfresco, the repository has the concept of node ownership. This ownership is optional and is implemented as an aspect.

The owner of a node may have specific ACLs granted to them. Ownership is implemented as the dynamic authority, ROLE_OWNER, and is evaluated in the context of each node for which an authorization request is made. The Ownable aspect, if present, defines a node's owner by storing a userName; if the Ownable aspect is not present, the creator is used as the default owner. If the userName of the current user matches, including case, the userName stored as the owner of the node, the current user will be granted all permissions assigned to the authority ROLE_OWNER.

The OwnableService is responsible for all of the following:

- Determining the owner of a node
- Setting the owner of a node
- Determining if a node has an owner
- Allowing the current user to take ownership of a node

The OwnableService is supported by an Ownable aspect defined in <installLocation>\tomcat \webapps\alfresco\WEB-INF\classes\alfresco\model\contentModel.xml.

There are permissions and permission groups associated with the Ownable aspect in the permission model and related access controls applied to the methods on the public OwnableService.

Setting up 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 set up of two tomcat nodes on two separate machines, talking to shared content store, shared database, but each with their own indexes. This is the simplest cluster to set up, gives redundancy due to the two machines, and can load-balance for performance or use the second node as a "hot spare" for fail over.

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). The common store is simpler and more prevalent.

When using multiple content stores, each cluster stores content 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. The index store cannot be shared between servers. To keep the indexes up to date, a timed thread updates each index directly from the database when running in a cluster.

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 on the other nodes in the cluster, the indexes on those nodes are 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 updated 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

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.

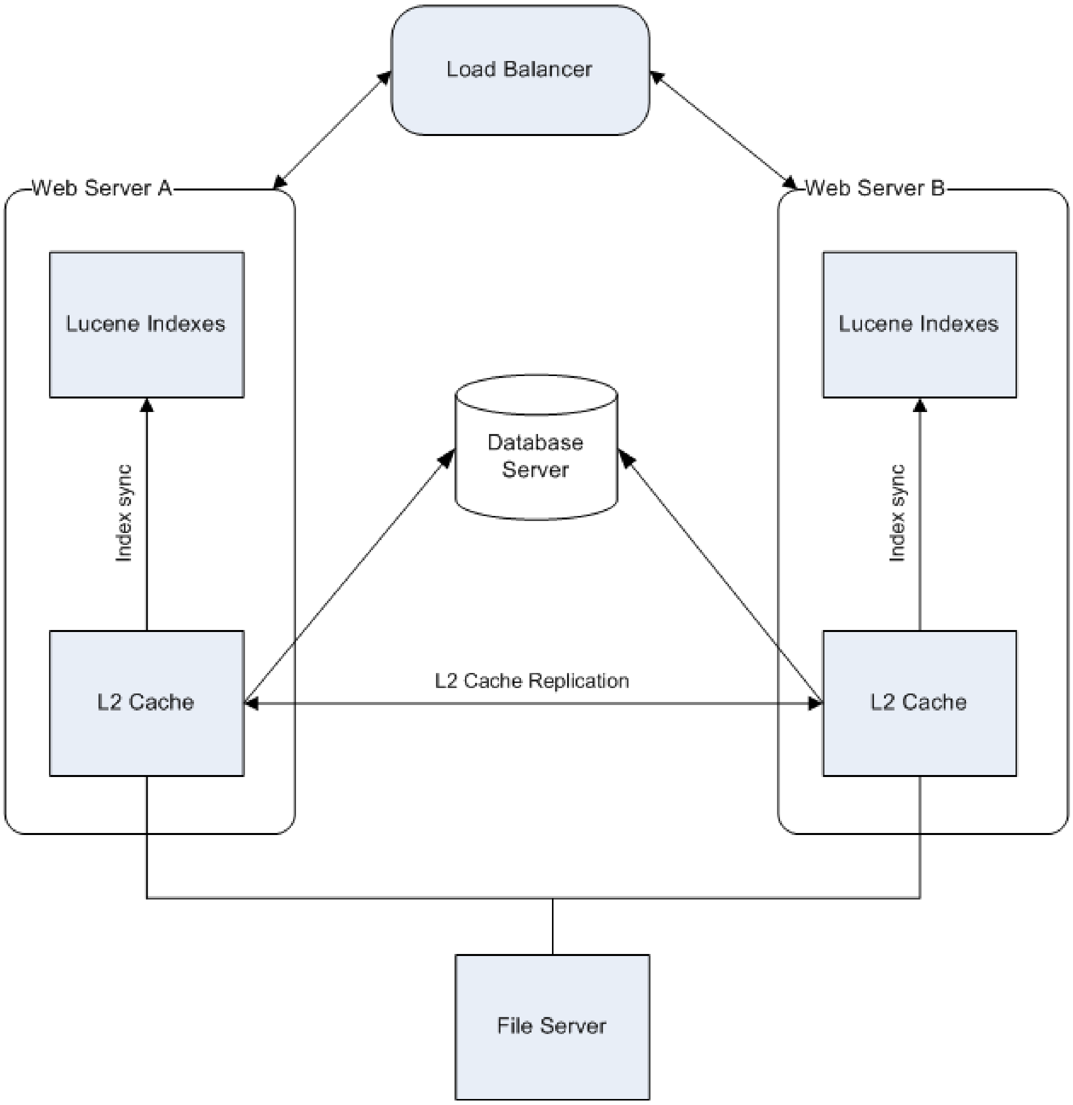
The L2 cache objects are stored in memory attached to the application scope of the server. Sticky sessions must be used to keep a user that has already established a session on one server for the entire session. By default, the cache replication makes use of RMI to replicate changes to all nodes in the cluster using the Peer Cache Replicator. Each replicated cache member notifies all other cache instances when its content has changed.

If you have issues with the replication of information in clustered systems, that is, the cache cluster test fails, you may want to confirm this by setting the following properties to true in the `alfresco-global.properties` file:

```
system.cache.disableMutableSharedCaches=
system.cache.disableImmutableSharedCaches=
```

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.

Initiating clustering

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. For details, refer to Using EHCache multicast discovery.

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.

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. To initiate clustering in Alfresco:

1. Locate the `ehcache-custom.xml.sample.cluster` file.
2. Copy the file to `<classpathRoot>/alfresco/extension/ehcache-custom.xml`.
3. Remove the following default definition of `cacheManagerPeerListenerFactory`:

```
 <cacheManagerPeerListenerFactory

 class="net.sf.ehcache.distribution.RMICacheManagerPeerListenerFactory"
            properties="socketTimeoutMillis=10000"
            />
```

4. Uncomment the extended definition by removing the comment lines `<!--` and `--!>` before and after the following section:

```
<cacheManagerPeerListenerFactory

 class="net.sf.ehcache.distribution.RMICacheManagerPeerListenerFactory"
            properties="hostName=${alfresco.ehcache.rmi.hostname},
            port=${alfresco.ehcache.rmi.port},

          remoteObjectPort=${alfresco.ehcache.rmi.remoteObjectPort},
          socketTimeoutMillis=
${alfresco.ehcache.rmi.socketTimeoutMillis}" />
```

5. Save the `ehcache-custom.xml` file.
6. Set the following properties in the `alfresco-global.properties` file:

Parameter	Description
`alfresco.cluster.name`	Specifies the name of the cluster. This property is mandatory. Without it, neither JGroups nor index tracking will be enabled.
`index.recovery.mode`	Set this to AUTO to ensure indexes are refreshed properly on startup.
`alfresco.ehcache.rmi.hostname`	The externally resolvable DNS name or IP address that other cluster members should use to send this one cache invalidation messages. This would normally be the server's host name. If you use a public IP v4 address, see the section on Using IP v4 addresses.
`alfresco.rmi.services.external.host`	The externally resolvable DNS name or IP address that external JMX and RMI clients, such as JConsole should use to contact this one. Set this to the same value as `alfresco.ehcache.rmi.hostname`.
`alfresco.ehcache.rmi.port`	Specifies the fixed port number to which to bind the ehcache RMI registry.
`alfresco.ehcache.rmi.remoteObjectPort`	Specifies the fixed port number through which to receive cache invalidation messages.

For example:

```
alfresco.cluster.name=cluster1
alfresco.jgroups.defaultProtocol=TCP
alfresco.tcp.start_port=7800
alfresco.tcp.initial_hosts=server1.company.com[7800]
alfresco.ehcache.rmi.hostname=server2.company.com
alfresco.rmi.services.external.host=server2.company.com
alfresco.ehcache.rmi.port=40001
alfresco.ehcache.rmi.remoteObjectPort=45001
```

You need to set any JGroups properties, especially if the TCP stack is required.

7. Restart each Alfresco server in the cluster and test that they are communicating using the test described in the Testing cache clustering topic.
8. Configure the content stores.

Configuring JGroups

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
```

1. Open the `alfresco-global.properties` file.
2. Set the general properties for the required 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 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`.
3. To select the TCP communication method, add the following property:

   ```
   alfresco.jgroups.defaultProtocol=TCP
   ```

   To select the FILE_PING communication method, add the following property in the `alfresco-global.properties` file:

   ```
   alfresco.jgroups.defaultProtocol=TCP-FPING
   ```
4. Set the property definitions for the chosen stack.

   The protocol stacks can be redefined using any of the standard JGroups transport protocols.

   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]` Note that `HOST_A` and `HOST_B` must be IP addresses, rather than host names.

TCP property	Description
`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 property for the FILE_PING stack is:

FILE_PING property	Description
`alfresco.fping.shared.dir`	Specifies the shared directory used by all cluster members to record their membership of the cluster and discover other members of the cluster. The default is `${dir.contentstore}`.

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.

5. (Optional) To specify your own JGroups configuration file, add the following properties:

```
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.

6. Save the `alfresco-global.properties` file.

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.

Using EHCache multicast discovery

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.

Configuring Hazelcast for JLAN Clustering

This section describes the configuring options available for JLAN clustering.

The Hazelcast library provides support for distributed data structures and communications channels that allow certain components of the Alfresco content repository server to function properly within a clustered environment. While configuring JLAN clustering, if you wish to cluster CIFS, FTP or NFS protocols with an Alfresco repository, you will need to configure the `hazelcastConfig.xml` file. This file is located at `alfresco/tomcat/shared/classes/alfresco/extension`.

You can configure the `hazelcastConfig.xml` file to create separate clusters by specifying the group-name and group-password. Also, you need to specify the network interface and multicast discovery option that Hazelcast should use. For more information, see the Hazelcast Documentation.

```
<group>
   <name>dev</name>
   <password>dev-pass</password>
</group>
<network>
   <port auto-increment="true">5701</port>
   <join>
      <multicast enabled="true">
        <multicast-group>224.2.2.3</multicast-group>
        <multicast-port>54327</multicast-port>
      </multicast>
      <tcp-ip enabled="false">
        <interface>127.0.0.1</interface>
      </tcp-ip>
   </join>
      <interfaces enabled="true">
        <interface>10.244.10.119</interface>
      </interfaces>
```

To enable hazelcast clustering, you need to set the following in your `alfresco-global.properties` file:

```
filesystem.cluster.enabled=true
filesystem.cluster.configFile=c:\\temp\\hazelcastConfig.xml
```

where `filesystem.cluster.enabled` allows you to enable or disable the filesystem cluster and `filesystem.cluster.configFile` is the location of Hazelcast configuration file.

Verifying the cluster

Testing cache clustering

We do not recommend using a clustered Alfresco installation until the following tests pass. If the tests don't pass, carefully review all the configuration settings and check that appropriate ports have been opened in any firewalls. Use the additional troubleshooting tips in Troubleshooting configuration.

If you need technical support, contact Alfresco Customer Support by email at *support@alfresco.com*.

1. For each cluster node, connect with a JMX client, such as JConsole.

   The following are typical connection details:

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

   username: `controlRole`

   password: `change_asap`

   If you are unable to connect externally via JConsole, the RMI communications necessary to support clustering are also unlikely to work. For details, see Troubleshooting configuration settings.
2. In each JConsole, navigate to **MBeans > Alfresco > RepoServerMgmt > Attributes**. Check the value of `TicketCountAll` attribute for each cluster node.
3. Using Alfresco Share, log in to one of the nodes.
4. Using JConsole, refresh the `TicketCountAll` attribute by clicking **Refresh**.
5. Check the value of `TicketCountAll` or `TicketCountNonExpired` attribute on all nodes again using JConsole. The values on each node should have increased in step with each other.

Index 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`.
8. On M1, perform an advanced search of the description field for `abcdef` and verify that the tutorial document is returned.
9. 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.
3. On Mx, perform a simple search for `abcdef` and ensure that the new document is retrieved.

This relies on index tracking, so it may take a few seconds for the document to be visible.

4. Open the document and ensure that the correct text is visible.

Testing WCM clustering

Testing 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.

Testing 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.

Testing 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.

Testing JGroups

This topic describes the steps used to test if JGroups is working correctly and to confirm that all cluster nodes are communicating successfully.

1. Enable `DEBUG` level logging for the following log4j logger classes:

   ```
   org.alfresco.enterprise.repo.cache.jgroups.JGroupsKeepAliveHeartbeatReceiver=debug
   org.alfresco.enterprise.repo.cache.jgroups.JGroupsKeepAliveHeartbeatSender=debug
   ```
2. Start up the servers.
3. Check for the presence of log messages beginning with `Sending cache peer URLs heartbeat:` in the logs.
4. Check for the presence of log messages beginning with `Received cache peer URLs heartbeat:` in the logs.

   If the cluster nodes can communicate with each other, the `Received cache peer URLs heartbeat:` log messages will appear in the log file in response to the heartbeat messages in Step 3. The log message will also display the host names of the peer nodes as the `src` attribute in the message.

   However, if there is no communication between the cluster nodes, only the `Sending cache peer URLs heartbeat:` messages will appear in the log file.
5. Once you are sure that all the cluster members are communicating successfully, it is advised that you enable `INFO` level (or greater) logging for the following log4j logger classes:

   ```
   org.alfresco.enterprise.repo.cache.jgroups.JGroupsKeepAliveHeartbeatReceiver=info
   org.alfresco.enterprise.repo.cache.jgroups.JGroupsKeepAliveHeartbeatSender=info
   ```

Configuring Share clustering

These steps are required for cluster configuration for Share. If you are using an HTTP load-balancing mechanism in front of a clustered installation, ‘sticky’ routing must be enabled for the HTTP requests made by the Share tier to the repository tier (the `/alfresco` application).

This can be achieved in one of two ways:

1. Hard-wire each `/share` instance to its own `/alfresco` instance, bypassing the load balancer.

   This can be achieved by populating each `share-config-custom.xml` file with a host name and port number that is not behind your load balancing mechanism.

2. If NTLM or Kerberos authentication is enabled with SSO, or external authentication is enabled, then Share will use cookie-based sessions and you can configure your load balancer to use sticky routing using the JSESSIONID cookie.

   Depending on whether you want to enable NTLM or Kerberos authentication, or external authentication, perform either of the following two steps:

   - **To enable NTLM or Kerberos authentication:** Refer to the instructions in Configuring authentication to configure alfrescoNtlm, passthru, or Kerberos authentication, and set either `ntlm.authentication.sso.enabled=true` or `kerberos.authentication.sso.enabled=true`.
   - **To enable external authentication:** Alternatively, if you enable external authentication, Share will still support manual log ins in the absence of any SSO mechanism, but will still use the necessary cookie-based sessions. To enable external authentication, follow the instructions in Configuring the Share default port and then set the following in `alfresco-global.properties`:

     ```
     authentication.chain=alfrescoNtlm1:alfrescoNtlm,external1:external
     external.authentication.proxyUserName=
     ```

If you are configuring a cluster, refer to Setting up high availability systems.

Configuring the cache peer URLs

If the cache clustering tests fail, enable debug so that you can track the issues. Refer to Tracking clustering issues.

If the cache peer URLs generated do not contain the correct host name or IP address of the sending server, then other machines will not be able to connect back. This occurs when the JVM call to get the current host name produces an incorrect result. That is, a host name that is not recognized in the cluster (this is typically `localhost`). This can be resolved at a system level, but can also be changed for the Alfresco cluster.

To change the host name and/or port values that the caches broadcast to the cluster:

1. Open the `ehcache-custom-xml` file.

   This file is created when initiating the cluster by making a copy of the `ehcache-custom.xml.sample.cluster` file and calling it `ehcache-custom-xml`.

2. Locate the cacheManagerPeerProviderFactory definition.
3. Comment out the `cacheManagerPeerListenerFactory` definition, and uncomment the second `cacheManagerPeerListenerFactory` definition.

   This extended definition allows additional properties to be set.

4. Set the additional properties in the `alfresco-global.properties` file.

   For example, set the host name of the current server using the following property setting:

   ```
   alfresco.ehcache.rmi.hostname=1.2.3.4
   ```

This should be set to the IP address that other cluster members should use to communicate with this node.

The other properties available are:

remoteObjectPort
This is the port number on which the remote nodes registry receive messages. The default is 0 (zero) which will select a free port.

port
The port that this node listens on. The default is 0 (zero) which will select a free port.

socketTimeoutMillis
The number of ms client sockets will stay open when sending messages to remote nodes. The default is 5 seconds, which should be long enough.

5. If the debug logging shows that the IP address or host name being sent or received by JGroups is wrong, set the JGroups bind address to the address that other members of the cluster should use to communicate with this node (`alfresco.jgroups.bind_address`). Refer to Configuring JGroups.

Tracking clustering issues

1. Enable the following log categories:
   - `log4j.logger.net.sf.ehcache.distribution=DEBUG`

     Check that heartbeats are received 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.enterprise.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
```

Configuring search

This section describes how to configure Alfresco search. It provides an overview of the Solr server and instructions on configuring it. This section also describes how to configure search in Share.

Configuring search in Alfresco Share

The following sections describe how to configure search in Alfresco Share.

Controlling permissions checking on search results in Share

This topic provides instructions on controlling permissions checking on search results in Share.

You can limit the amount of time Alfresco spends on ensuring that the user executing the search has the necessary permissions to see each result. Setting this limit increases search speed and reduces the use of resources.

You can limit both the time spent and the number of documents checked before Alfresco returns a search query. Alfresco uses the `system.acl.maxPermissionCheckTimeMillis` and the `system.acl.maxPermissionChecks` properties to set a limit to these values. The default values are 10000 and 1000 respectively.

To change these values:

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 the `system.acl.maxPermissionCheckTimeMillis` property and the value you want it to contain.

   For example, `system.acl.maxPermissionCheckTimeMillis=20000`.
4. Add the `system.acl.maxPermissionChecks` property and the value you want it to contain.

   For example, `system.acl.maxPermissionChecks=1000`.

   - If you increase these values and have a query that returns a very large number of results, (a) the search results will take longer to be returned to the user, and (b) the system will spend longer to check permissions, leading to the possibility of performance degradation.
   - If you set these values to a low number, you run the risk of inconsistent search results every time you run the same search.

Controlling search results in Share

This topic provides instructions on controlling the maximum number of items that a Share search returns.

By default, the Share search returns a maximum of 250 search results. If you want the search to return more than 250 entries, complete the following steps.

1. Ensure that the `<web-extension>\share-config-custom.xml` file exists. If the file does not exist:
   a. Locate the following `.sample` configuration override file:

      `<web-extension>\share-config-custom.xml.sample`

   b. Copy and rename the file to:

      `<web-extension>\share-config-custom.xml`
2. Copy the `<config evaluator="string-compare" condition="Search">` section from the `<configRoot>\classes\alfresco\share-config.xml` file.
3. Paste the copied section into the `<web-extension>\share-config-custom.xml` file.
4. In the `<web-extension>\share-config-custom.xml` file, edit the value for the `<max-search-results>` property to reflect the number of search results you want Share to return.
5. For the changes to take effect, refresh the Alfresco web scripts. To refresh the web scripts:

a. Navigate to the Alfresco web scripts Home page.

   For example, go to: http://<your-host>:8080/share/page/index.

b. Click on **Refresh Web Scripts**.

   You have now refreshed the web scripts and set a limit to the number of items a search in Share returns.

6. Test the search configuration.

   a. Browse to the location of your Alfresco installation.

      For example, `http://<your-host>:8080/share`.

   b. Search for folders or documents in the Share repository.

   Notice that the number of search items returned is not more than what you specified in the `<web-extension>\share-config-custom.xml` file.

   Custom searches and searches from the node browser use the `solr.query.maximumResultsFromUnlimitedQuery` property to control search results. For more information, see Solr core configuration properties.

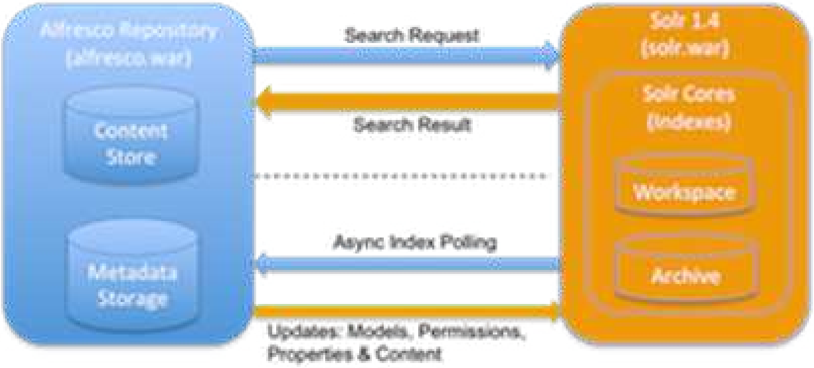
Solr overview

Alfresco supports use of the Solr Enterprise search platform for searching within the Alfresco repository. Also, the existing embedded Lucene index remains available.

Solr is an open source enterprise search platform that uses lucene as indexing and search engine. Solr is written in Java and runs as a standalone search server within the Tomact application server. Alfresco sends HTTP and XML input to Solr and searches for content. Solr updates the cores or indexes and returns the result of the query in XML or JSON format.

There are two cores or indexes in Solr version 1.4:

- **WorkspaceSpacesStore**: used for searching all live content stored at `alfresco/alf_data/contentstore`within the Solr search server.
- **ArchiveSpacesStore**: used for searching content that has been marked as deleted at `alfresco/alf_data/contentstore`within the Solr search server.



Alfresco only supports Solr version 1.4, so any latest version of Solr (for example Solr 3.6.1) will not be supported by Alfresco.

Solr is the default search mechanism for new installations installed with the Setup Wizard. Also, the Solr server is supported only when running in a Tomcat application server. Therefore, if you are running Alfresco within a different application server and you wish to use Solr search, you must install Tomcat.

Advantages of Solr search over Lucene search

Solr provides improvements on the search capabilities within Alfresco over the embedded Lucene index that improved the performance, scalability, and general support and configuration.

In particular, the Solr search server offers the following advantages over an embedded Lucene search engine:

- Fixed tokenization, in addition to local specific tokenization, to support better cross-language support
- Uses the date time analyzer for `d:datetime` properties variable resolution search, for example, `cm:created:2010`
- Improved performance on the PATH implementation
- Evaluates READ access at query time
- No in-transaction indexing
- Cross-locale ordering for `d:text` and `d:mltext` properties
- Full integration with the Alfresco data model, including tracking model changes
- Support using the Search Service for simple field-based faceting - faceting is after read access enforcement
- No need to duplicate indexes on every machine in a cluster
- Search support can be scaled separately from the Alfresco repository (for example, two Solr instances for a four cluster node)
- Solr built-in administration http://localhost:8080/solr/alfresco/admin/ for checking tokenization behavior, terms in the index, and so on

Eventual Consistency

Alfresco 4 introduces the concept of eventual consistency to overcome the scalability limitations of in-transaction indexing.

An Alfresco installation configured to use Solr does not include any transactional indexing operations, and therefore, the database and indexes does not need to be in perfect synchronization at any given time. It relies on an index that is updated at a configurable interval. By default, this interval is 15 seconds. See Solr core configuration properties.

The index tracker records all new transactions for Alfresco and updates the indexes. In this way, the indexes will be eventually consistent with the database.

When not to use Solr

Solr can be used for search in Alfresco but it is not appropriate to use it for some functionality.

If you require the following functionality, you will not be able to use Solr for search:

- When you do not have a Tomcat application server in which to run the Solr server
- In-transaction indexing when you are dissatisfied with the eventually consistent results, or you cannot resolve this requirement in another way - for example, when taking advantage of changes to the Node Service or writing canned data base queries. For more information, see Eventual Consistency.

Configuring Solr

The way that you configure Alfresco to use Solr depends on how you have installed Alfresco.

If you install Alfresco using the setup wizard, Solr is installed and enabled automatically. Solr is installed in the same Tomcat container as Alfresco, and the connection URL is unchanged from the default. The Solr home is in the Alfresco data directory, which also contains the Solr data files.

If you have an existing Alfresco installation, and you wish to configure it to use Solr search, you need to apply the Solr archive to your web application.

Installing and Configuring Solr

This section describes how to install and configure Solr using the distribution archive file on an existing Alfresco installation using Tomcat.

The distribution archive file is called `alfresco-enterprise-solr-4.1.5.zip`.

This file contains the following artifacts:

- a template SOLR home directory containing `solr.xml`, which is expected by Solr
- Solr WAR file
- an example context to wire up in Tomcat
- a `lib` directory with all the required Alfresco and other JARs
- two Solr core configurations: one to track the live SpacesStore and one to track items archived from the SpacesStore

You can install Solr either to the same Tomcat application server as Alfresco or a separate Tomcat. The Solr server indexes data in Alfresco by periodically tracking the changes made to Alfresco. It does so by calling a RESTful API that describe the latest transactions to it. The Alfresco server performs searches through the Solr server by issuing queries through a special API. For this reason, there needs to be two-way communication between the Alfresco server and the Solr server. For security reasons, the communication channel between the Alfresco server and Solr server must be secured by means of https encryption and mutual client certificate authentication.

The following instructions use `<ALFRESCO_TOMCAT_HOME>` to refer to the tomcat directory where Alfresco is installed and `<SOLR_TOMCAT_HOME>` to refer to the tomcat directory where Solr is installed. These may be the same or different directories, depending on whether you have chosen to install Solr on a standalone server. The `<ALFRESCO_HOME>` refers to the directory where Alfresco is installed.

1. Extract the `alfresco-enterprise-solr-4.1.5.zip` file to a location. For example, `<SOLR-ARCHIVE>`.
2. Copy the `solr-tomcat-context.xml` file to `<SOLR_TOMCAT_HOME>\conf\Catalina\localhost\solr.xml`.
3. Edit `docBase` in the `<SOLR_TOMCAT_HOME>\conf\Catalina\localhost\solr.xml` file to point to `<SOLR-ARCHIVE>\apache-solr-1.4.1.war`.
4. Edit `solr/home` in XML to point to `<SOLR-ARCHIVE>`.

   For example:

   ```
   <?xml version="1.0" encoding="utf-8"?>
   <Context docBase="<SOLR-ARCHIVE>\apache-solr-1.4.1.war" debug="0"
    crossContext="true">
      <Environment name="solr/home" type="java.lang.String" value="<SOLR-
   ARCHIVE>" override="true"/>
   </Context>
   ```
5. For each core, edit the `solrcore.properties` file:

   - `<SOLR-ARCHIVE>/archive-SpacesStore/conf/solrcore.properties`
   - `<SOLR-ARCHIVE>/workspace-SpacesStore/conf/solrcore.properties`

   Set the `data.dir.root` property to the location where the Solr indexes will be stored. You can set the same value for the both cores, and the cores will create the sub-directories.

6. Ensure that Alfresco has already been started at least once and the `<ALFRESCO_TOMCAT_HOME>/webapps/alfresco/WEB-INF` directory exists.
7. Create and populate a keystore directory for the Alfresco and Solr servers. By default, the keystore directory is created in `<ALFRESCO_HOME>/alf_data/keystore`. Note that at this stage the keystore directory will just be a template, containing standard keys. To secure the installation, you must follow the steps to generate new keys as explained in the Generating Secure Keys for Solr Communication section.

   For example:

   For Unix:

   ```
   mkdir -p <ALFRESCO_HOME>/alf_data/keystore

   cp <ALFRESCO_TOMCAT_HOME>/webapps/alfresco/WEB-INF/classes/alfresco/
   keystore/* <ALFRESCO_HOME>/alf_data/keystore
   ```

   For Windows:

   ```
   mkdir <ALFRESCO_HOME>\alf_data\keystore

   copy <ALFRESCO_TOMCAT_HOME>\webapps\alfresco\WEB-INF\classes\alfresco
   \keystore\* <ALFRESCO_HOME>\alf_data\keystore
   ```
8. Configure the Alfresco and Solr tomcat application servers to use the keystore and truststore for `https` requests by editing the specification of the connector on port 8443 in `<ALFRESCO_TOMCAT_HOME>/conf/server.xml` and `<SOLR_TOMCAT_HOME>/conf/server.xml` as shown below:

   Remember to replace `<ALFRESCO_HOME>/alf_data/keystore` with the full path to your keystore directory. Also, make sure that the Tomcat connectors is configured with `maxSavePostSize="-1"` to support SSL.

   For example:

   ```
   <Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
         SSLEnabled="true" maxThreads="150" scheme="https"
         keystoreFile="<ALFRESCO_HOME>/alf_data/keystore/ssl.keystore"
         keystorePass="kT9X6oe68t" keystoreType="JCEKS" secure="true"
    connectionTimeout="240000"
         truststoreFile="<ALFRESCO_HOME>/alf_data/keystore/ssl.truststore"
         truststorePass="kT9X6oe68t" truststoreType="JCEKS"
    clientAuth="false" sslProtocol="TLS" maxSavePostSize="-1"/>
   ```
9. Configure Alfresco to use the keystore and truststore for client requests to Solr by specifying `dir.keystore` in `<ALFRESCO_TOMCAT_HOME>/shared/classes/alfresco-global.properties`.

   Remember to replace `<ALFRESCO_HOME>/alf_data/keystore` with the full path to your keystore directory.

   For example:

   ```
   dir.keystore=<ALFRESCO_HOME>/alf_data/keystore
   ```
10. Configure an identity for the Alfresco server. In `<SOLR_TOMCAT_HOME>/conf/tomcat-users.xml`, add the following:

    Remember, you can choose a different username, such as the host name of the Alfresco server, but it must match the `REPO_CERT_DNAME` that you will later specify in the keystore in the Generating Secure Keys for Solr Communication section.

    For example:

    ```
    <user username="CN=Alfresco Repository, OU=Unknown, O=Alfresco Software
     Ltd., L=Maidenhead, ST=UK, C=GB" roles="repository" password="null"/>
    ```

11. Configure an identity for the Solr server. In `<ALFRESCO_TOMCAT_HOME>/conf/tomcat-users.xml`, add the following:

    Remember, you can choose a different username but it must match the `SOLR_CLIENT_CERT_DNAME` that you will later specify in the keystore in the Generating Secure Keys for Solr Communication section.

    For example:

    ```
    <user username="CN=Alfresco Repository Client, OU=Unknown, O=Alfresco
     Software Ltd., L=Maidenhead, ST=UK, C=GB" roles="repoclient"
     password="null"/>
    ```
12. To complete the installation, it is necessary to secure the two-way communication between Alfresco and Solr by generating your own keys. See the Generating Secure Keys for Solr Communication topic.

Generating Secure Keys for Solr Communication

This task describes how to replace or update the keys used to secure communication between Alfresco and Solr, using secure keys specific to your Alfresco installation.

The following instructions assume that Solr has been extracted and a keystore directory has already been created, either automatically by the Alfresco installer or manually by following the instructions in the Configuring Solr section.

If you are applying these instructions to a clustered installation, the steps should be carried out on a single host and then the generated `.keystore` and `.truststore` files must be replicated across all other hosts in the cluster.

1. Obtain the file `generate_keystores.sh` (for Linux and Solaris) or `generate_keystores.bat` (for Windows) from the Alfresco Customer Support website under **Online Resources > Downloads > Alfresco Enterprise 4.0 > <Alfresco Version> generate_keystores.x**.
2. Edit the environment variables at the beginning of the file to match your environment.
    a. If you are updating an environment created by the Alfresco installer, you only need to edit `ALFRESCO_HOME` to specify the correct installation directory.
    b. For manual installations, carefully review `ALFRESCO_KEYSTORE_HOME`, `SOLR_HOME, JAVA_HOME`, `REPO_CERT_DNAME` and `SOLR_CLIENT_CERT_DNAME` and edit as appropriate. For WebLogic installations, it is necessary to edit the `CERTIFICATE_VALIDITY` variable so that the certificate expires before the year 2105.
3. Run the edited script.

    You should see the message “Certificate update complete” and another message reminding you what `dir.keystore` should be set to in the `alfresco-global.properties` file.

Solr Directory Structure

After you have installed Alfresco, several directories and configuration files related to Solr will be available in the Alfresco home directory. This section explains the Solr directory structure.

`alfresco\alf_data\solr`

This is the Solr home directory. It contains the Solr cores: `archive-SpacesStore`(for deleted content) and `workspace-SpacesStore`(for live content). It also contains two configurations files: `solr-tomcat-context.xml` and `solr.xml`.
A Sole core holds one Lucene index and the supporting cnfiguration for that index.

The Solr directory contains the following sub-folders and files:

- `archive`: This directory contains the Lucene index for the archive core.
- `archive-SpacesStore`: This is the configuration directory for the archive core.
- `docs`: This directory contains example update, delete, and commit XML documents.
- `lib`: This directory contains extra libraries that Solr loads on start up. These libraries are used to communicate with Alfresco via CMIS, Alfresco data model or Spring Surf Web Scripts.
- `workspace`: This directory contains the Lucene index for the workspace core.
- `workspace-SpacesStore`: This is the configuration directory for the workspace core.
- `apache-solr-1.4.1.war`: This is the patched version of Apache Solr 1.4.1 Web Application by Alfresco.
- `apache-solr-1.4.1.war.unpatched`: This is the original unpatched version of Apache Solr 1.4.1.
- `CreateSSLKeystores.txt`: This file contains instructions for generating Solr SSL keystores.
- `HowToSetUpSolr.txt`: This file contains instructions on setting up Solr on a dedicated server.
- `solr.xml`: This configuration file specifies the cores to be used by Solr.
- `solr-tomcat-context.xml`: This configuration file specifies the Solr web application context template to use when installing Solr in separate tomcat server.

`alfresco\alf_data\solr\workspace-SpacesStore` and `alfresco\alf_data\solr\archive-SpacesStore`

Both these directories are instance directories for Solr core.
The Solr directory contains the following sub-folders and files:

- `alfrescoModels`: This directory contains all the content models that come out of the box with Alfresco. Any new custom content model added to Alfresco are synced to this directory so that Solr knows about it.
- `alfrescoResources`: This directory contains resource files that specifies what Data Type Index Analyzers should be used for different languages. For example, the default analyzer for Alfresco data type is defined in the **dataTypeAnalyzers.properties** file as:

  ```
  d_dictionary.datatype.d_text.analyzer=org.alfresco.repo.search.impl.lucene.anal
  ```
- `conf`: This is the main configuration directory for Solr core.

`alfresco\alf_data\solr\workspace-SpacesStore\conf` and `alfresco\alf_data\solr\archive-SpacesStore\conf`

This is the configuration directory for Solr core. Both these directories are instance directories for Solr core.
The `conf` directory contains the following configuration files: `schema.xml`, `solrconfig.xml`, `solrcore.properties`, `ssl.repo.client.keystore`, `ssl.repo.client.truststore`, `ssl-keystore-passwords.properties` and `ssl-truststore-passwords.properties`. To know more about these configuration files, see Solr Configuration Files.

Solr Configuration Files

When you install Alfresco 4.0, several Solr-related configuration files are made available to you. The section lists the Solr configuration files, their location in the Alfresco directory structure and description.

Configuration File	Location	Description
repository.properties	alfresco\tomcat\webapps\alfresco\WEB-INF\classes\alfresco\	This file specifies the Solr-related properties for configuring how Alfresco connects to the Solr server. As the Solr server runs in the same Tomcat instance as Alfresco, the connection properties are setup to connect to a locally running Solr server.
schema.xml	alfresco\alf_data\solr\<core>, where <core> is the location of core's configuration directory, for example alfresco\alf_data\solr\workspace-SpacesStore\conf or alfresco\alf_data\solr\archive-SpacesStore\conf	This file defines the schema for the index including field type definitions with associated analyzers. It contains details about the fields that you can include in your document and also describes how those fields can be used when adding documents to the index or when querying those fields.
solr.xml	alfresco\tomcat\conf\catalina\localhost\	This file defines the Solr web application context. It specifies the location of the Solr war file and sets up the Solr home directory.
solr.xml	alfresco\alf_data\solr	This file specifies the cores to be used by Solr.
solrconfig.xml	alfresco\alf_data\solr\workspace-SpacesStore\conf or alfresco\alf_data\solr\archive-SpacesStore\conf	This file specifies the parameters for configuring Solr. Also, the Solr search components are added to this file.
solrcore.properties	alfresco\alf_data\solr\workspace-SpacesStore\conf or alfresco\alf_data\solr\archive-SpacesStore\conf	This is the property configuration file for a core. Solr supports system property substitution, so properties that need substitution can be put in to this file. There is one solrcore.properties file in each core's configuration directory. For details, see the Solrcore Configuration Properties section.
solr-tomcat-context.xml	alfresco\alf_data\solr	This file specifies the Solr web application context template to use when installing Solr in separate tomcat server.
ssl.repo.client.keystore	alfresco\alf_data\solr\workspace-SpacesStore\conf or alfresco\alf_data\solr\archive-SpacesStore\conf	This keystore contains the Solr public/private RSA key pair.
ssl-keystore-passwords.properties	alfresco\alf_data\solr\workspace-SpacesStore\conf or alfresco\alf_data\solr\archive-SpacesStore\conf	This file contains the password information for ssl.repo.client.keystore.

Configuration File	Location	Description
ssl.repo.client.truststore	alfresco\alf_data \solr\workspace-SpacesStore\conf or alfresco\alf_data \solr\archive-SpacesStore\conf	This keystore contains the trusted Alfresco Certificate Authority certificate (which has been used to sign both the repository and Solr certificates)
ssl-truststore-passwords.properties	alfresco\alf_data \solr\workspace-SpacesStore\conf or alfresco\alf_data \solr\archive-SpacesStore\conf	This file contains the password information for ssl.repo.client.truststore.

Solr core configuration properties

The solrcore.properties configuration file is the property configuration file for a Solr core. There is one solrcore.properties file in each core's configuration directory. This section lists the properties of this file, their description, and the default value.

Property Name	Description	Default Value
data.dir.root	This property specifies the top level directory path for the indexes managed by Solr.	C:/Alfresco/alf_data/solr
data.dir.store	This property specifies the directory relative to data.dir.root where the data for this core is stored.	workspace/SpacesStore
enable.alfresco.tracking	This property instructs Solr if it should index Alfresco content in the associated Alfresco repository store or not.	true
cache.alfresco.size	This property specifies the Alfresco cache size used internally for PATH looks up.	100
max.field.length	This property specifies the maximum number of tokens to include for each field. By default, all tokens are added.	2147483647
alfresco.host	This property specifies the host name for the Alfresco instance that Solr should track and index. In a default installation, both Alfresco and Solr runs in the same Tomcat instance and on the same host, so host would be set to local host.	localhost
alfresco.port	This property specifies the HTTP port for the Alfresco instance that Solr should track and index.	8080
alfresco.port.ssl	This property specifies the HTTPS port for the Alfresco instance that Solr should track and index.	8443

Property Name	Description	Default Value
`alfresco.cron`	This property specifies the cron expression that instructs Solr how often to track Alfresco and index new or updated content. The default value indicates that Solr tracks Alfresco every 15 seconds.	`0/15 * * * * ? *`
`alfresco.stores`	This property specifies the Alfresco repository store that this core should index.	`workspace://SpacesStore`
`alfresco.baseUrl`	This property configures the base URL to Alfresco web project.	`/alfresco`
`alfresco.lag`	When Solr tracking starts, it aims to get up to date to the current time (in seconds), less this lag.	`1000`
`alfresco.hole.retention`	Each track will revisit all transactions from the timestamp of the last in the index, less this value, to fill in any transactions that may have been missed.	`3600000`
`alfresco.batch.count`	This property indicates the number of updates that should be made to this core before a commit is executed.	`1000`
`alfresco.secureComms`	This property instructs Solr if it should talk to Alfresco over HTTP or HTTPS. Set to none if a plain HTTP connection should be used.	`https`
`alfresco.encryption.ssl.keystore.type`	This property specifies the CLIENT keystore type.	`JCEKS`
`alfresco.encryption.ssl.keystore.provider`	This property specifies the Java provider that implements the `type` attribute (for example, JCEKS type). The provider can be left unspecified and the first provider that implements the keystore type specified is used.	
`alfresco.encryption.ssl.keystore.location`	This property specifies the CLIENT keystore location reference. If the keystore is file-based, the location can reference any path in the file system of the node where the keystore is located.	`ssl.repo.client.keystore`

Property Name	Description	Default Value
`alfresco.encryption.ssl.keystore.passwordFileLocation`	This property specifies the location of the file containing the password that is used to access the CLIENT keystore, also the default that is used to store keys within the keystore.	`ssl-keystore-passwords.properties`
`alfresco.encryption.ssl.truststore.type`	This property specifies the CLIENT truststore type.	`JCEKS`
`alfresco.encryption.ssl.truststore.provider`	This property specifies the Java provider that implements the `type` attribute (for example, JCEKS type). The provider can be left unspecified and the first provider that implements the truststore type specified is used.	`&nbsp;`
`alfresco.encryption.ssl.truststore.location`	This property specifies the CLIENT truststore location reference. If the truststore is file-based, the location can reference any path in the file system of the node where the truststore is located.	`ssl.repo.client.truststore`
`alfresco.encryption.ssl.truststore.passwordFileLocation`	This property specifies the location of the file containing the password that is used to access the CLIENT truststore, also the default that is used to store keys within the truststore.	`ssl-truststore-passwords.properties`
`alfresco.enableMultiThreadedTracking`	This property enables/disables multi-threaded tracking.	`true`
`alfresco.corePoolSize`	This property specifies the pool size for multi-threaded tracking. It is used for indexing nodes.	`3`
`alfresco.maximumPoolSize`	This property specifies the maximum pool size for multi-threaded tracking.	`-1`
`alfresco.keepAliveTime`	This property specifies the time (in seconds) to keep non-core idle threads in the pool.	`120`
`alfresco.threadPriority`	This property specifies the priority that all threads must have on the scale of 1 to 10, where 1 has the lowest priority and 10 has the highest priority.	`5`
`alfresco.threadDaemon`	This property sets whether the threads run as daemon threads or not. If set to `false`, shut down is blocked else it is left unblocked.	`true`

Property Name	Description	Default Value
`alfresco.workQueueSize`	This property specifies the maximum number of queued work instances to keep before blocking against further adds.	-1
`alfresco.maxTotalConnections`	This property is used for HTTP client configuration.	40
`alfresco.maxHostConnections`	This property is used for HTTP client configuration.	40
`solr.filterCache.size`	This property specifies the maximum number of entries in the Solr filter cache.	512
`solr.filterCache.initialSize`	This property specifies the initial capacity (number of entries) of the Solr filter cache.	512
`solr.queryResultCache.size`	This property configures the Solr result cache.	1024
`solr.queryResultCache.initialSize`	This property configures the Solr result cache.	1024
`solr.documentCache.size`	This property configures the Solr document cache.	512
`solr.documentCache.initialSize`	This property configures the Solr document cache.	512
`solr.queryResultMaxDocsCached`	This property configures the Solr result cache.	200
`solr.maxBooleanClauses`	This property specifies the number of Boolean clauses in a query. It can affect range or wildcard queries that expand to big Boolean queries.	10000
`alfresco.transactionDocsBatchSize`	This property is used for batch fetching updates during tracking.	100
`alfresco.changeSetAclsBatchSize`	This property is used for batch fetching updates during tracking.	100
`alfresco.aclBatchSize`	This property is used for batch fetching updates during tracking.	10

Property Name	Description	Default Value
`solr.query.maximumResultsFromUnlimitedQuery`	This property is used to set the maximum number of results returned by Solr queries, therefore, limiting otherwise unconstrained Solr queries to return a finite number of results. This prevents such unconstrained queries from consuming excessive resources. By default, the value is set to the same value as the `system.acl.maxPermissionChecks` property. - This property is set in the `alfresco-global.properties` file, and not in the `solrcore.properties` file. - The Alfresco Share searches use the `<max-search-results>` property to control the number of search results. For more information, see Controlling search results in Share. - This property is used for queries without an explicit limit.	`1000`
`alfresco.index.transformContent`	If this property is set to false, the index tracker will not transform any content and only the metadata will be indexed.	`false`

Solr Subsystem

Search is contained within a subsystem, and it has an implementation of either `solr` or `lucene`.

The Alfresco Solr search subsystem supports the same query languages as the embedded Lucene subsystem, and the same fields (`ID` , `PARENT`) are also available. The following properties in the `alfresco-global.properties` file are related to Solr and are setup as follows, by default:

```
### Solr indexing ###
index.subsystem.name=solr
dir.keystore=${dir.root}/keystore
solr.port.ssl=8443
```

As shown above, note that search has been moved into a subsystem with a solr and lucene implementation.

Activating Solr

This section describes how to activate the Solr search mechanism in a manual Alfresco installation or an upgrade from a previous version.

Global properties file

1. Open the `<configRoot>alfresco-global.properties` file.
2. Set the following properties:

Property	Description
`index.subsystem.name`	The subsystem type value. The `index.subsystem.name` property values are either `solr` or `lucene`.
`solr.host`	The host name where the Solr instance is located.
`solr.port`	The port number on which the Solr instance is running.
`solr.port.ssl`	The port number on which the Solr SSL support is running.

   For example, some example properties for activating Solr are:

```
index.subsystem.name=solr
solr.host=localhost
solr.port=8080
solr.port.ssl=8443
```

3. Save the global properties file and restart the Alfresco server.

Share Admin Console

1. Open the Share Admin Console.
2. Edit the following properties:

Property	Description
`index.subsystem.name`	Select the subsystem type value as either `solr` or `lucene`.
`solr.host`	The host name where the Solr instance is located.
`solr.port`	The port number on which the Solr instance is running.
`solr.port.ssl`	The port number on which the Solr SSL support is running.

3. Click Save.

JMX client

1. To switch between Lucene and Solr in JMX, choose **MBeans > Alfresco > Configuration > Search**.
2. Set the manager `sourceBeanName` to either `solr` or `lucene`.

   The subsystems have their own related properties. The `managed - solr` instance exposes the `solr.base.url` property. The `lucene` subsystem exposes all the properties that had to be set at start up.
3. These can now be configured live and the subsystem redeployed.

Solr troubleshooting for SSL configurations

When you have an Alfresco installation that requires an SSL configuration, you may encounter some connection issues.

You may see a message on the Tomcat console similar to the following (and may find that Solr search and/or the Solr tracking is not working):

```
Aug 22, 2011 8:19:21 PM org.apache.tomcat.util.net.jsse.JSSESupport
 handShake     WARNING: SSL server initiated
renegotiation is disabled, closing connection
```

This message indicates that one side of the SSL connection is trying to renegotiate the SSL connection. This form of negotiation was found to be susceptible to man-in-the-middle attacks and it was disabled in the Java JSEE stack until a fix could be applied.

Refer to the following link for more information: http://www.oracle.com/technetwork/java/javase/documentation/tlsreadme2-176330.html.

Refer also to the following links: http://www.gremwell.com/enabling_ssl_tls_renegotiation_in_java o http://tomcat.apache.org/tomcat-6.0-doc/config/http.html.

If your version of Java does not have the fix, you need to re-enabled renegotiation by performing the following steps:

1. Add the `-Dsun.security.ssl.allowUnsafeRenegotiation=true` option to JAVA_OPTS.
2. Add the `allowUnsafeLegacyRenegotiation="true"` option to the Tomcat SSL connector.

Solr security

Communications between Alfresco repository and Solr are protected by SSL with mutual authentication.

Both the repository and Solr have their own public/private key pair, signed by an Alfresco Certificate Authority, which are stored in their own respective keystores. These keystores are bundled with Alfresco. You can also create your own keystores. For an overview on how to create an SSL public/private key and certificate for the repository, please see Generating Repository SSL Keystores.

Keystores are used also to protect repository and Solr communications using encryption and mutual authentication. To do this, keystores store RSA keys and certificates.

It is assumed that the keystore files are stored in `alf_data`. Place the keystore files from the directory `repository/config/alfresco/keystore` in the `$ALF_DATA/keystore` directory.

Repository SSL key stores

This section describes the key stores used by the repository for SSL.

The repository has two key stores it uses for SSL:

- `ssl keystore` contains a public/private RSA key pair for the repository
- `ssl truststore` contains the trusted Alfresco Certificate Authority certificate (which has been used to sign both the repository and Solr certificates)

These key stores can be stored in any location.

Update the following key store properties in the `alfresco-global.properties` file to specify the location of the key stores:

`ssl keystore`

Property	Description
`encryption.ssl.keystore.location`	Specifies the key store location.
`encryption.ssl.keystore.provider`	Specifies the key store provider.
`encryption.ssl.keystore.type`	Specifies the key store type.
`encryption.ssl.keystore.keyMetaData.location`	Specifies the key store metadata file location.

`ssl truststore`

Property	Description
`encryption.ssl.truststore.location`	Specifies the trust store location.
`encryption.ssl.truststore.provider`	Specifies the trust store provider.
`encryption.ssl.truststore.type`	Specifies the trust store type.
`encryption.ssl.truststore.keyMetaData.location`	Specifies the trust store metadata file location.

Solr SSL key stores

This section describes the key stores used by Solr for SSL.

Solr core has two key stores it uses for SSL:

- `ssl.repo.client.keystore` contains a Solr public/private RSA key pair
- `ssl.repo.client.truststore` contains the trusted Alfresco Certificate Authority certificate (which has been used to sign both the repository and Solr certificates)

Connecting to the SSL-protected Solr web application

All Solr URLs, which are bundled within Alfresco, are protected by SSL.

To use these URLs from a browser, you need to import a browser-compatible key store to allow mutual authentication and decryption to work. The following steps describe how to import the key store into your browser (these relate to Firefox, other browsers will have a similar mechanism):

1. Open the FireFox **Certificate Manager** by selecting **Tools > Options > Advanced > Encryption > View Certificates > Your Certificates**.
2. Import the browser keystore `browser.p12` that is located in your `WEB_INF/classes/alfresco/keystore` directory.
3. Enter the password `alfresco`.

   A window displays showing that the key store has been imported successfully. The **Certificate Manager** now contains the imported key store with the Alfresco repository certificate under the **Your Certificates** tab.
4. Close the **Certificate Manager** by clicking **OK**.
5. In the browser, navigate to a Solr URL.

   For example, use the URL http://localhost:8080/solr.

   The browser displays an error message window to indicate that the connection is untrusted. This is due to the Alfresco certificate not being tied to the server IP address. In this case, view the certificate and confirm that it is signed by the Alfresco Certificate Authority.
6. Expand **I understand the risks**.
7. Select **Add Exception**.
8. Click **View**.

   This displays the certificate.

9. Confirm that the certificate was issued by Alfresco Certificate Authority, and then confirm the **Security Exception**.

Access to Solr is then granted. The Solr Admin page is displayed with a list of cores:

- **Admin alfresco** (workspace core for live content)
- **Admin archive** (workspace core for archive content)

The **Solr Admin (alfresco)** page is displayed when you click on the **Admin alfresco** core. This page is useful for finding information about the Solr installation, such as deployed schemas, Solr configuration, indexed fields etc.

Generating Repository SSL Keystores

This task describes how to create an SSL public/private keystore and a certificate for the repository.

The following instructions creates an RSA public/private key pair for the repository with a certificate signed by the Alfresco Certificate Authority (CA). It also creates a truststore for the repository containing the CA certificate that is used to authenticate connections to specific repository URLs from Solr. The instructions assume the existence of the Alfresco CA key and certificate to sign the repository certificate. However, for security reasons these may not available. You can either generate your own CA key and certificate or use a recognised Certificate Authority, such as Verisign. To generate your own CA key and certificate, see Generating CA key and certificate.

`<store password>` is the keystore password. The file `C:\Alfresco\alf_data\keystore\ssl-keystore-passwords.properties` contains passwords for the SSL keystore, whereas, the file `C:\Alfresco\alf_data\keystore\ssl-truststore-passwords.properties` contains passwords for the SSL truststore.

1. Generate the repository public/private key pair in a keystore.

```
$ keytool -genkey -alias repo -keyalg RSA -keystore ssl.keystore -
storetype JCEKS -storepass <store password>
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  Alfresco Repository
What is the name of your organizational unit?
  [Unknown]:
What is the name of your organization?
  [Unknown]:  Alfresco Software Ltd.
What is the name of your City or Locality?
  [Unknown]:  Maidenhead
What is the name of your State or Province?
  [Unknown]:  UK
What is the two-letter country code for this unit?
  [Unknown]:  GB
Is CN=Alfresco Repository, OU=Unknown, O=Alfresco Software Ltd.,
 L=Maidenhead, ST=UK, C=GB correct?
  [no]:  yes

Enter key password for <repo>
 (RETURN if same as keystore password):
```

2. Generate a certificate request for the repository key.

```
$ keytool -keystore ssl.keystore -alias repo -certreq -file repo.csr -
storetype JCEKS -storepass <store password>
```

3. Alfresco CA signs the certificate request and creates a certificate that is valid for 365 days.

```
$ openssl x509 -CA ca.crt -CAkey ca.key -CAcreateserial -req -in repo.csr
 -out repo.crt -days 365
Signature ok
```

```
subject=/C=GB/ST=UK/L=Maidenhead/O=Alfresco Software Ltd./OU=Unknown/
CN=Alfresco Repository
Getting CA Private Key
Enter pass phrase for ca.key:
```

4. Import the Alfresco CA key into the repository keystore.

```
$ keytool -import -alias AlfrescoCA -file ca.crt -keystore ssl.keystore -
storetype JCEKS -storepass <store password>
Enter keystore password:
Owner: CN=Alfresco CA, O=Alfresco Software Ltd., L=Maidenhead, ST=UK,
 C=GB
Issuer: CN=Alfresco CA, O=Alfresco Software Ltd., L=Maidenhead, ST=UK,
 C=GB
Serial number: 805ba6dc8f62f8b8
Valid from: Fri Aug 12 13:28:58 BST 2011 until: Mon Aug 09 13:28:58 BST
 2021
Certificate fingerprints:
  MD5:  4B:45:94:2D:8E:98:E8:12:04:67:AD:AE:48:3C:F5:A0
  SHA1: 74:42:22:D0:52:AD:82:7A:FD:37:46:37:91:91:F4:77:89:3A:C9:A3
  Signature algorithm name: SHA1withRSA
  Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 08 42 40 DC FE 4A 50 87   05 2B 38 4D 92 70 8E 51  .B@..JP..+8M.p.Q
0010: 4E 38 71 D6                                        N8q.
]
]

#2: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#3: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 08 42 40 DC FE 4A 50 87   05 2B 38 4D 92 70 8E 51  .B@..JP..+8M.p.Q
0010: 4E 38 71 D6                                        N8q.
]

[CN=Alfresco CA, O=Alfresco Software Ltd., L=Maidenhead, ST=UK, C=GB]
SerialNumber: [    805ba6dc 8f62f8b8]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore
```

5. Import the CA-signed repository certificate into the repository keystore.

```
$ keytool -import -alias repo -file repo.crt -keystore ssl.keystore -
storetype JCEKS -storepass <store password>
Enter keystore password:
Certificate reply was installed in keystore
```

6. Convert the repository keystore to a pkcs12 keystore (for use in browsers, such as Firefox). Specify the keystore passowrd for pkcs12 keystore as 'alfresco'.

```
keytool -importkeystore -srckeystore ssl.keystore -srcstorepass <keystore
 password> -srcstoretype JCEKS -srcalias
repo -srckeypass kT9X6oe68t -destkeystore firefox.p12 -deststoretype
 pkcs12 -deststorepass alfresco -destalias repo
-destkeypass alfresco
```

7. Create a repository truststore containing the Alfresco CA certificate.

   ```
   keytool -import -alias AlfrescoCA -file ca.crt -keystore ssl.truststore -
   storetype JCEKS -storepass <store password>
   ```

8. Copy the keystore and truststore to the repository keystore location defined by the property `dir.keystore`.
9. Update the SSL properties (properties starting with the prefixes `alfresco.encryption.ssl.keystore` and `alfresco.encryption.ssl.truststore`).

Generating a Certificate Authority (CA) Key and Certificate

This task describes how to create Alfresco CA key and certificate to sign the repository certificate.

1. Generate the CA private key.

   ```
   $ openssl genrsa -des3 -out ca.key 1024
   Generating RSA private key, 1024 bit long modulus
   ..........++++++
   ..++++++
   e is 65537 (0x10001)
   Enter pass phrase for ca.key:
   Verifying - Enter pass phrase for ca.key:
   ```

2. Generate the CA self-signed certificate.

   ```
   $ openssl req -new -x509 -days 3650 -key ca.key -out ca.crt
   Enter pass phrase for ca.key:
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a
    DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [AU]:GB
   State or Province Name (full name) [Some-State]:UK
   Locality Name (eg, city) []:Maidenhead
   Organization Name (eg, company) [Internet Widgits Pty Ltd]:Alfresco
    Software Ltd.
   Organizational Unit Name (eg, section) []:
   Common Name (eg, YOUR name) []:Alfresco CA
   Email Address []:
   ```

Solr monitoring and troubleshooting

This section provides help for monitoring and resolving any Solr index issues that might arise as a result of a transaction.

Performing a full reindex with Solr

This task describes how to force a full reindex when using Solr.

This task assumes you are using only one Solr instance for all nodes in the Alfresco cluster. If not, then you need to repeat this process on each Solr instance used in the cluster.

1. Confirm the location of the Solr core directories for `archive-SpacesStore` and `workspace-SpacesStore` cores. This can be determined from the `solrcore.properties` file for both the cores.

   By default, the `solrcore.properties` file can be found at `C:\alfresco\alf_data\solr\workspace-SpacesStore\conf` and `C:\alfresco\alf_data\solr\archive-SpacesStore\conf`.

The Solr core location is defined in the `solrcore.properties` file as:

```
# data is in ${data.dir.root}/${data.dir.store}

data.dir.root=C:/alfresco/alf_data/solr
data.dir.store=workspace/SpacesStore
```

2. Shut down Solr. If Solr is running on the same server as an Alfresco instance, the Alfresco instance will also be shut down.
3. Delete the contents of the index data directories for each Solr core at `${data.dir.root}`.
    - `C:\alfresco\alf_data\solr\workspace\SpacesStore`
    - `C:\alfresco\alf_data\solr\archive\SpacesStore`
4. Delete all the Alfresco models for each Solr core at `${data.dir.root}`.
    - `C:\alfresco\alf_data\solr\workspace-SpacesStore\alfrescoModels`
    - `C:\alfresco\alf_data\solr\archive-SpacesStore\alfrescoModels`
5. Start up the application server that runs Solr.
6. Monitor the application server logs for Solr. You will get the following warning messages on bootstrap:

```
WARNING: [alfresco] Solr index directory 'c:/alfresco/alf_data/solr/
workspace/SpacesStore/index' doesn't exist. Creating new index...
09-May-2012 09:23:42
 org.apache.solr.handler.component.SpellCheckComponent inform
WARNING: No queryConverter defined, using default converter
09-May-2012 09:23:42 org.apache.solr.core.SolrCore initIndex
WARNING: [archive] Solr index directory 'c:/alfresco/alf_data/solr/
archive/SpacesStore/index' doesn't exist. Creating new index...
```

7. Use the Solr administration console to check the health of the Solr index.

The process of building the Solr indexes may take some time depending on the size of the repository. To monitor the reindexing progress, use the Solr administration console and check the logs for any issues during this activity. Whilst the index is rebuilding, searches from Alfresco may return incomplete or inaccurate results.

Unindexed Solr Transactions

You can check the status of the Solr index to identify the nodes to a transaction that failed to index.

To generate a general report, including the last transaction indexed and the time, use:

```
http://localhost:8080/solr/admin/cores?action=REPORT&wt=xml
```

The `REPORT` parameter compares the database with the index and generates an overall status report with the following details:

- `DB transaction count`: indicates the transaction count on the database
- `DB acl transaction count`: indicates the acl transaction count on the database
- `Count of duplicated transactions in the index`: indicates the number of transactions that appear more than once in the index. The value of this parameter should be zero. If not, there is an issue with the index.
- `Count of duplicated acl transactions in the index`: indicates the number of ACL transactions that appear more than once in the index. The value of this parameter should be zero. If not, there is an issue with the index.
- `Count of transactions in the index but not the database`: indicates the number of transactions in the index but not in the database. This count includes empty transactions

that have been purged from the database. The value of this parameter should be zero. If not, there may be an issue with the index.

- `Count of acl transactions in the index but not the DB`: indicates the number of acl transactions in the index but not in the database. The value of this parameter should be zero. If not, there is an issue with the index. Note that empty acl transactions are not currently purged from the database.
- `Count of missing transactions from the Index`: indicates the number of transactions in the database but not in the index. The value of this index should be zero when the index is up-to-date.
- `Count of missing acl transactions from the Index`: indicates the number of acl transactions in the database but not in the index. The value of this index should be zero when the index is up-to-date.
- `Index transaction count`: indicates the number of transactions in the index.
- `Index acl transaction count`: indicates the number of acl transactions in the index.
- `Index unique transaction count`: indicates the number of unique transactions in the index.
- `Index unique acl transaction count`: indicates the number of unique acl transactions in the index.
- `Index leaf count`: indicates the number of docs and folders in the index.
- `Count of duplicate leaves in the index`: indicates the number of duplicate docs or folders in the index. The value of this parameter should be zero. If not, there is an issue with the index.
- `Last index commit time`: indicates the time stamp for the last transaction added to the index. It also indicates that transactions after this time stamp have not yet been indexed.
- `Last Index commit date`: indicates the time stamp as date for the last transaction added to the index. It also indicates that transactions after this date have not yet been indexed.
- `Last TX id before holes`: indicates that transactions after this id will be checked again to make sure they have not been missed. This is computed from the index at start up time. By default, it is set an hour after the last commit time found in the index. Solr tracking, by default, goes back an hour from the current time to check that no transactions have been missed .
- `First duplicate` : indicates if there are duplicate transactions in the index. It returns the id of the first duplicate transaction.
- `First duplicate acl tx`: indicates if there are duplicate acl transactions in the index. It returns the id of the first duplicate acl transaction.
- `First transaction in the index but not the DB`: if the related count is > 0, it returns the id of the first offender.
- `First acl transaction in the index but not the DB` : if the related count is > 0, it retuns the id of the first offender.
- `First transaction missing from the Index`: if the related count is > 0, it returns the id of the first offender.
- `First acl transaction missing from the Index`: if the related count is > 0, it returns the id of the first offender.
- `First duplicate leaf in the index`: if the related count is > 0, it returns the id of the first offender.

To generate a summary report, use:

```
http://localhost:8080/solr/admin/cores?action=SUMMARY&wt=xml
```

With multi-threaded tracking, you can specify additional tracking details and tracking statistics:

- `detail=true`: provide statistics per tracking thread
- `hist=true`: provides a histogram of the times taken for tracking operations for each tracking thread
- `reset=true`: resests all tracking statistics
- `values=true`: reports (by default) the last 50 values recorded for each tracking operation for each thread

The `SUMMARY` parameter provides the status of the tracking index and reports the progress of each tracking thread. It generates a report with the following details:

- `Active`: indicates the tracker for the core active.
- `Last Index Commit Time`: indicates the time stamp for the last transaction that was indexed.
- `Last Index Commit Date`: indicates the time stamp as a date for the last transaction that was indexed. Changes made after this time are not yet in the index.
- `Lag`: indicates the difference in seconds between the last transaction time stamp on the server and the time stamp for the last transaction that was indexed.
- `Duration`: indicates the time lag as an XML duration.
- `Approx transactions remaining`: indicates the approximate number of transactions to index in order to bring the index up-to-date. It is calculated as the last transaction id on the server minus the last transaction id indexed. It includes all the missing and empty transactions.
- `Approx transaction indexing time remaining`: it is based on `Approx transactions remaining`, the average number of nodes per transaction and the average time to index a node (how long the index will take to be up-to-date). The estimate is in the most appropriate scale, for example, seconds, minutes, hours and days.
- `Model sync times (ms)`: indicates summary statistics for model sync time. It supports additional information with `&detail=true`, `&hist=true` and `&value=true`.
- `Acl index time (ms)`: indicates summary statistics for acl index time. It supports additional information with `&detail=true`, `&hist=true` and `&value=true`.
- `Node index time (ms)`: indicates summary statistics for node index time. It supports additional information with `&detail=true`, `&hist=true` and `&value=true`.
- `Acl tx index time (ms)`: indicates the summary statistics for acl transaction index time. It supports additional information with `&detail=true`, `&hist=true` and `&value=true`.
- `Tx index time (ms)`: indicates summary statistics for transaction index time. It specifies the estimated time required to bring the index up-to-date.
- `Docs/Tx`: indicates summary statistics for the number of documents per transaction. It supports additional information with `&detail=true`, `&hist=true` and `&value=true`.
- `Doc Transformation time (ms)`: indicates summary statistics for document transformation time. It supports additional information with `&detail=true`, `&hist=true` and `&value=true`.

Troubleshooting Solr Index

This section describes how to repair a transaction that failed to index.

To repair an unindexed or failed transaction (as identified by the `REPORT` option in the Unindexed Solr Transactions section), run the following report:

```
http://localhost:8080/solr/admin/cores?action=FIX
```

The FIX parameter compares the database with the index and identifies any missing or duplicate transactions. It then updates the index by either adding or removing transactions.

Use the PURGE parameter to remove transactions, acl transactions, nodes and acls from the index. It can also be used for testing wrong transactions and then to fix them.

```
http://localhost:8080/solr/admin/cores?
action=PURGE&txid=1&acltxid=2&nodeid=3&aclid=4
```

Use the REINDEX parameter to reindex a transaction, acl transactions, nodes and acls.

```
http://localhost:8080/solr/admin/cores?
action=REINDEX&txid=1&acltxid=2&nodeid=3&aclid=4
```

Use the INDEX parameter to create entries in the index. It can also be used to create duplicate index enteries for testing.

```
http://localhost:8080/solr/admin/cores?
action=PURGE&txid=1&acltxid=2&nodeid=3&aclid=4
```

Use the following setting to specify an option core for the report. If it is absent, a report is produced for each core. For example:

```
&core=alfresco
&core=archive
```

You can also fix index issues, check the index cache and backup individual indexes via JMX. The status of the index can be checked using the JMX client on the **JMX MBeans > Alfresco > solrIndexes > <store>** tabs. The default view is the Solr core summary. The operations run the same consistency checks that are available by URL.

Solr backup and restore

This section describes the process for backing up and restoring the Solr server.

Your backup strategy must be tested end-to-end, including restoration of backups that were taken previously. Ensure that you have adequately tested your backup scripts prior to deploying Alfresco to production.

Backing up Solr

This section describes how to back up the Solr indexes.

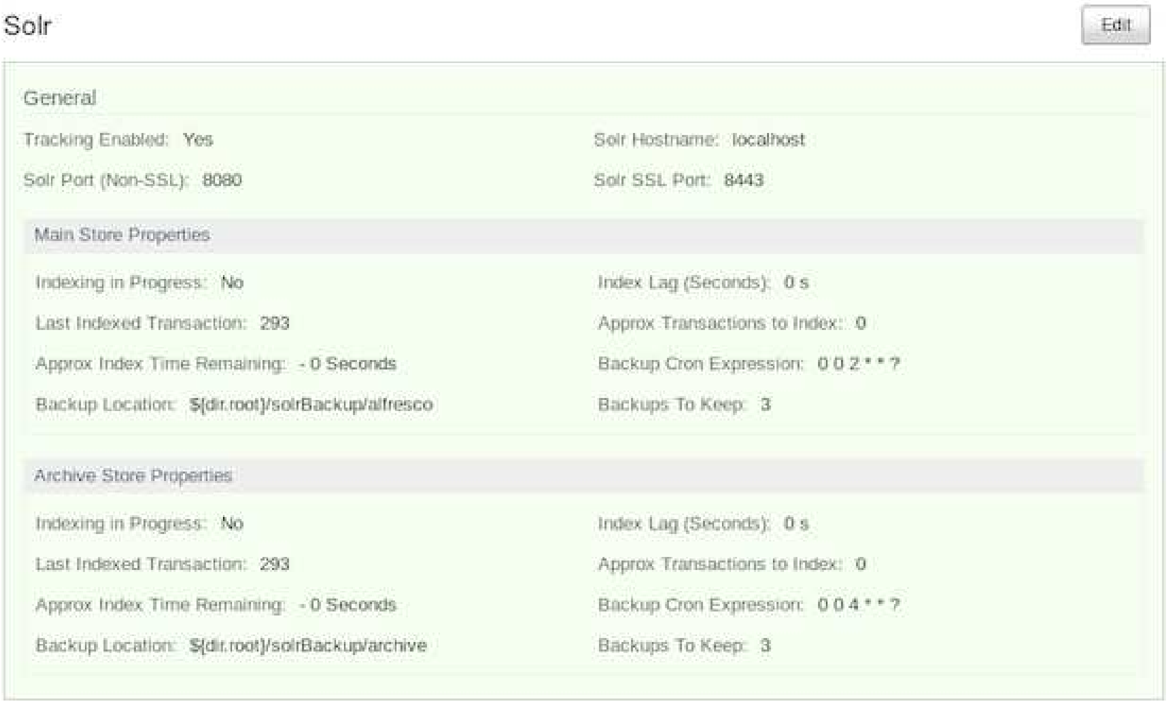
You can set the Solr indexes backup properties using the Admin Console in Share or by editing the alfresco-global.properties file. Also, you can backup the Solr indexes using a JMX client, such as JConsole.

Set up Solr backup properties using Share Admin Console

You can only see the Admin Console if you are an administrator user.

1. On the toolbar, expand the **More** menu, and then click **More** in the **Admin Tools...** list.
2. Under the **Tools** section on the left navigation bar, you see various tools available and the options that you can set. In the **Search** sub-section, click on **Solr**.

   The **Solr** window is displayed.



Solr

General

Tracking Enabled: Yes

Solr Hostname: localhost

Solr Port (Non-SSL): 8080

Solr SSL Port: 8443

Main Store Properties

Indexing in Progress: No

Index Lag (Seconds): 0 s

Last Indexed Transaction: 293

Approx Transactions to Index: 0

Approx Index Time Remaining: - 0 Seconds

Backup Cron Expression: 0 0 2 * * ?

Backup Location: ${dir.root}/solrBackup/alfresco

Backups To Keep: 3

Archive Store Properties

Indexing in Progress: No

Index Lag (Seconds): 0 s

Last Indexed Transaction: 293

Approx Transactions to Index: 0

Approx Index Time Remaining: - 0 Seconds

Backup Cron Expression: 0 0 4 * * ?

Backup Location: ${dir.root}/solrBackup/archive

Backups To Keep: 3

Here, you can specify the backup location and edit backup properties for each core of the Solr index: **Main Store** and **Archive Store**.

- **Backup Cron Expression**: Specifies a Quartz cron expression that defines when backups occur. Solr creates a timestamped sub-directory for each index back up you make.
- **Backup Location**: Specifies the full-path location for the backup to be stored.
- **Backups To Keep**: Specifies the maximum number of index backups that Solr should store.

3. Click **Edit**.
4. Specify the full location path on the Solr server file system to store the index backup in the **Backup Location** text box.
5. Click **Save**.

Specifying Solr backup directory via `alfresco-global.properties` file

This task shows how to specify the Solr backup directory via `alfresco-global.properties` file.

- To set the solr backup directory using the `alfresco-global.properties` file, set the value of the following properties to the full path where the backups should be kept:

```
solr.backup.archive.remoteBackupLocation=
solr.backup.alfresco.remoteBackupLocation=
```

Back up Solr indexes using JMX client

You can use the JMX client, JConsole to backup Solr indexes, edit Solr backup properties and setup the backup directory.

- You can set the backup of Solr indexes using the JMX client, such as JConsole on the **JMX MBeans > Alfresco > Schedule > DEFAULT > MonitoredCronTrigger > search.alfrescoCoreBackupTrigger > Operations > executeNow** tab. The default view is the Solr core summary. Alternatively, navigate to **MBeans > Alfresco >SolrIndexes >coreName >Operations >backUpIndex** tab. Type the directory name in the **remoteLocation** text box and click **backUpIndex**.

- Solr backup properties can be edited using the JMX client on the **JMX MBeans > Alfresco > Configuration > Search > managed > solr > Attributes** tab. The default view is the Solr core summary.
- To use JMX client to setup Solr backup directory, navigate to **MBeans tab > Alfresco > Configuration > Search > managed > solr > Attributes** and change the values for `solr.backup.alfresco.remoteBackupLocation` and `solr.backup.archive.remoteBackupLocation` properties.

Restoring Solr Indexes

This section describes the process for restoring the Solr indexes.

During the recovery process, Solr queries Alfresco to update the indexes restored from a backup. To restore Solr indexes, use the following steps:

1. Stop the Solr server.
2. Copy a backup index to the data directory for each core.

   Remember to use a backup created from the same Alfresco instance.
3. Restart the Solr server. Solr will start to track the indexes based on the state of the restored index.

Migrating from Lucene to Solr search

This section describes how to migrate from Alfresco Enterprise 3.x with Lucene search server to Alfresco Enterprise 4.x with Solr search server.

With Alfresco Enterprise 3.x, the default search engine was Lucene. When you upgrade to Alfresco Enterprise 4.x using the setup wizard, the default search engine is Solr. However, if you upgrade using the WAR file, the default search engine is Lucene.

To determine the current search server, navigate to the `Search Manager` page at **Alfresco Share Admin Console > Tools > Search > Search Manager**. Click the **Edit** button to change between the Solr and Lucene search subsystem.

Use the steps below to migrate from Alfresco Enterprise 3.x with Lucene search server to Alfresco Enterprise 4.x with Solr search server.

1. Upgrade to Alfresco Enterprise 4.x and continue to use Lucene search server as before.
2. Install and configure Solr to track the repository. For details, see the Installing and Configuring Solr topic.
3. Monitor progress using the `SUMMARY` report.

   ```
   http://localhost:8080/solr/admin/cores?action=SUMMARY&wt=xml
   ```

   For details, see the Unindexed Solr Transactions topic.
4. When the index is updated as reported by the `SUMMARY` report, you can use the `REPORT` option and check the following:

   - In the `REPORT` option, leaf count should match the number of live nodes in the repository (assuming nothing is changing and the index is updated). The index contains a leaf for failed nodes, so failures need to be considered separately.
   - Any missing transactions; if there are issues, use the `FIX` option.

     ```
     http://localhost:8080/solr/admin/cores?action=FIX
     ```

     For more information, see the Troubleshooting Solr Index topic.
   - Find errors with specific nodes using `EXCEPTIONMESSAGE:*` option.

```
https://localhost:8443/solr/alfresco/afts?
q=EXCEPTIONMESSAGE:*&wt=xml
```

- If there are any issues, use the `REINDEX` option with the relevant node id.

```
http://localhost:8080/solr/admin/cores?
action=REINDEX&txid=1&acltxid=2&nodeid=3&aclid=4
```

  For more information, see the Troubleshooting Solr Index topic.

5. When the Solr index is updated, enable the Solr subsystem and disable the Lucene subsystem.
6. Validate the query results.

   You can switch back to the Lucene search server at any time but there may be a pause while the index is being updated. Also, note that Lucene should be set to **auto**.

Full text search configuration properties for Solr and Lucene indexes

The `repository.properties` file defines the properties that influence how all indexes behave. This section describes the full text search properties, for the Solr and Lucene indexes, contained in the `repository.properties` file.

The main index and deltas all use the same configuration. The data dictionary settings for properties determine how individual properties are indexed.

Note that the following properties are set in the `repository.properties` file. However, if you wish to change them, we recommend that you add the relevant property to the `alfresco-global.properties` file and then make the changes. No changes should be done in the `repository.properties` file.

Solr index properties

solr.host=localhost
The host name where the Solr instance is located.

solr.port=8080
The port number on which the Solr instance is running.

solr.port.ssl=8443
The port number on which the Solr SSL support is running.

solr.solrUser=solr
The Solr user name.

solr.solrPassword=solr
The Solr password.

solr.secureComms=https
The HTTPS connection.

solr.solrConnectTimeout=5000
The Solr connection timeouts in ms.

solr.solrPingCronExpression=0 0/5 * * * ? *
The cron expression defining how often the Solr Admin client (used by JMX) pings Solr if it goes away.

Lucene index - general properties

dir.indexes=${dir.root}/lucene-indexes
The directory that contains all lucene indexes and deltas against those indexes.

dir.indexes.backup=${dir.root}/backup-lucene-indexes
The directory for index backups.

dir.indexes.lock=${dir.indexes}/locks
The directory that contains the locks for lucene indexes.

lucene.maxAtomicTransformationTime=100
Transformations of content that are likely to take longer than this time (in millis) will be done in the background. To force atomic content indexing, increase this value.

lucene.query.maxClauses=10000
Max Clauses (Lucene standard parameter)
Lucene queries limit the number of clauses in a boolean query to this value. Some queries are expanded into a whole set of boolean query with many clauses under the covers. For example, searching for luc* will expand to a boolean query containing an "OR" for every token the index knows about that matches luc*.

lucene.indexer.batchSize=1000000
Batch size (Alfresco indexing parameter)
The indexer stores a list of what it has to do as changes are made using the node service API. Typically, there are many events that would cause a node to be re-indexed. Keeping an event list means the actions can be optimized - the algorithm limits re-indexes to one per batch size, and will not index if a delete is pending, etc. When the list of events reaches this size, the whole event list is processed and documents are added to the delta index.

lucene.indexer.contentIndexingEnabled=true
This property controls whether or not the content of the documents is indexed. If false, content is not indexed.

Lucene caching properties

Caching is carried across an index, for the composite index reader.

lucene.indexer.cacheEnabled=true
Enable/disable index level caching
ALL

lucene.indexer.maxDocIdCacheSize=10000
Cache index doc id to NodeRef.
This cache is designed to avoid a disk read and synchronization when finding the noderef association with a lucene document
Lucene currently synchronizes on reading a document.
ALL

lucene.indexer.maxDocumentCacheSize=100
Cache for lucene documents by lucene index doc id
ALL

lucene.indexer.maxIsCategoryCacheSize=-1
Cache which lucene docs are Alfresco categories (-1 caches all)
CATEGORY

lucene.indexer.maxLinkAspectCacheSize=10000
Cache link aspects (increase to improve the performance of category queries)
Used in linking children to aspects
CATEGORY

lucene.indexer.maxParentCacheSize=10000
Cache parent lookups (increase to improve the performance of PATH queries)
Used in linking children to parents
CATEGORY and PATH

lucene.indexer.maxPathCacheSize=10000
Cache the first part of category PATH lookups
CATEGORY ONLY

lucene.indexer.maxTypeCacheSize=10000
Cache alfresco type for docs in the lucene index
CATEGORY ONLY

Lucene merger properties

Properties for merge.

lucene.indexer.mergerMaxMergeDocs=1000000

lucene.indexer.mergerMergeFactor=5

lucene.indexer.mergerMergeBlockingFactor=1

lucene.indexer.mergerMaxBufferedDocs=-1

lucene.indexer.mergerRamBufferSizeMb=16

Lucene writer properties

Properties specific to IndexWriters used for transactional indexes. These options only apply to the index process - the resulting index will be optimized.

lucene.indexer.writerMaxMergeDocs=1000000

lucene.indexer.writerMergeFactor=5

lucene.indexer.writerMaxBufferedDocs=-1

lucene.indexer.writerRamBufferSizeMb=16

Lucene merging behavior properties

Properties to control merging behavior.

lucene.indexer.mergerTargetIndexCount=5
Target for the number of indexes (more than this and we start merging)

lucene.indexer.mergerTargetOverlayCount=5
Target for the number of indexes that relate directly to a transaction.
More than this and we start merging deletions and transforming overlays to indexes.

lucene.indexer.mergerTargetOverlaysBlockingFactor=1
Throttle factor so the stack of indexes and overlays does not become too large and affect performance.

lucene.indexer.maxDocsForInMemoryMerge=10000
Control in-memory merges

lucene.indexer.maxRamInMbForInMemoryMerge=16
Control in-memory merges

lucene.indexer.maxDocsForInMemoryIndex=10000
Max size for indexes held in memory

lucene.indexer.maxRamInMbForInMemoryIndex=16
Max size for indexes held in memory

Other Lucene properties

lucene.indexer.termIndexInterval=128

lucene.indexer.useNioMemoryMapping=true

lucene.indexer.postSortDateTime=true

lucene.indexer.defaultMLIndexAnalysisMode=EXACT_LANGUAGE_AND_ALL

lucene.indexer.defaultMLSearchAnalysisMode=EXACT_LANGUAGE_AND_ALL

lucene.indexer.maxFieldLength=10000

lucene.write.lock.timeout=10000

lucene.commit.lock.timeout=100000

lucene.lock.poll.interval=100

Data dictionary options

The indexing behavior of each property can be set in the content model. By default, they are indexed atomically. The property value is not stored in the index, and the property is tokenized when it is indexed.

The following example shows how indexing can be controlled.

Enabled="false"
If this is false, there will be no entry for this property in the index.

Atomic="true"
If this is true, the property is indexed in the transaction, if not the property is indexed in the background.
Indexing of content that requires transformation before being indexed (for example, PDFs) will only obey `Atomic=true` if the transformation takes less time than the value specified for `lucene.maxAtomicTransformationTime`. Refer to the general properties.

Stored="true"
If true, the property value is stored in the index and may be obtained using the Lucene low-level query API.
This can be useful while debugging systems to see exactly what is being indexed, but do not set this to true on production systems.

Tokenised="true"
If "true", the string value of the property is tokenized before indexing.
if "false", it is indexed "as is" as a single string.
if "both" then both forms above are in the index.

The tokenizer is determined by the property type in the data dictionary. This is locale sensitive as supported by the data dictionary, so you could switch to tokenize all your content in German. At the moment you cannot mix German and English tokenization.

```
<type name="cm:content">
       <title>Content</title>
       <parent>cm:cmobject</parent>
       <properties>
          <property name="cm:content">
             <type>d:content</type>
             <mandatory>false</mandatory>
             <index enabled="true">
                <atomic>false</atomic>
                <stored>false</stored>
                <tokenised>true</tokenised>
             </index>
          </property>
       </properties>
```

```
     </type>
```

Indexing defaults

The effective indexing defaults for all properties are as follows:

```
<index enabled="true">
                 <atomic>true</atomic>
                 <stored>false</stored>
                 <tokenised>true</tokenised>
              </index>
              ...
```

Setting Solr log4j values

This topic describes the method of setting the Solr log4j values.

To set debug logging levels for Alfresco-Solr components, follow the steps below:

1. Copy `tomcat/webapps/solr/WEB-INF/classes/log4j.properties` to `<solrRootDir>/log4j.properties`.
2. Edit it to add your required logging setting. For example:

   ```
   log4j.logger.org.alfresco.solr.tracker.CoreTracker=DEBUG
   ```
3. Changes to the `log4j.properties` file will be re-read by Solr when it starts up. If you need to make changes to the logging level while the system is running, going to the following URL (either in a browser or for example, using curl) will cause Solr to re-load the properties file.

   ```
   https://<solrHostName>:<solrPort>/solr/admin/cores?
   action=LOG4J&resource=log4j.properties
   ```

Calculate the memory needed for Solr nodes

Solr can have high memory requirements. This topic describes the formula to calculate the memory needed for the Alfresco internal data structures used in Solr for PATH queries and read permission enforcement.

By default, there are two cores in Solr: `WorkspaceSpacesStore` and `ArchiveSpacesStore`. Normally, each core has one searcher but can have a maximum of two searchers.

In the calculation below:

- N = refers to the number of nodes in the store. Each core's value is calculated separately. If there are more than two cores, you will need to add additional queries to calculate the value for that core (as shown in the example code block below).
- T = refers to the number of transactions in the repository and this is same for each core
- A = refers to the number of ACLs in the repository and this is same for each core
- X = refers to the number of ACL transactions in the repository and this is same for each core

The values for N, T, A and X come from the database. Use the following commands to derive these values:

```
select * from
(select count( * ) N_Alfresco from alf_node where store_id = (select id from
 alf_store where protocol = 'workspace' and identifier = 'SpacesStore')) as
 N1 ,
(select count( * ) N_Archive from alf_node where store_id = (select id from
 alf_store where protocol = 'archive' and identifier = 'SpacesStore')) as N2 ,
(select count( * ) T from alf_transaction ) as T,
(select count( * ) A from alf_access_control_list ) as A,
(select count( * ) X from alf_acl_change_set) as X;
```

For example, if there are three cores, include additional queries to calculate the value for that core, as shown below:

```
select * from
(select count( * ) N_Alfresco from alf_node where store_id = (select id from
 alf_store where protocol = 'workspace' and identifier = 'SpacesStore')) as
 N1 ,
(select count( * ) N_Archive from alf_node where store_id = (select id from
 alf_store where protocol = 'archive' and identifier = 'SpacesStore')) as N2 ,
(select count( * ) N_Version2 from alf_node where store_id = (select id from
 alf_store where protocol = 'workspace' and identifier = 'version2Store'))as
 N3 ,
(select count( * ) T from alf_transaction ) as T,
(select count( * ) A from alf_access_control_list ) as A,
(select count( * ) X from alf_acl_change_set) as X;
```

Memory calculation for the Alfresco data structures associated with one searcher

For a store containing 100M nodes, 100M transactions, 100M ACLs and 100M ACL transactions, 21.6 G of memory is needed. Assuming there are not many ACLs or ACL changes, for 100M nodes, you will need 12G -16G of memory depending on the number of transactions. This calculation is based on the following formula: `120N + 32(T + A + X)` bytes.

Memory calculation for the Solr caches associated with one searcher

The Solr cache will use up to (2N + T + A + X)/8 bytes for an entry in any cache.

The formula to calculate the total memory needed for the caches for a single core is: `(solr.filterCache.size + solr.queryResultCache.size + solr.authorityCache.size + solr.pathCache.size) * (2N + T + A + X)/8` bytes

So, for 100M documents and 100M transactions, 96G of memory is needed using the out of box configuration.

```
(512 + 1024 + 512 + 512)(300M)/8 = 96G
```

The default cache values needs to change to accommodate a large repository. So, for 100M documents, 100M transactions and reduced cache size, 12G of memory is needed.

```
(64 + 128 + 64 + 64)(300M)/8 = 12G
```

Overall Solr memory use

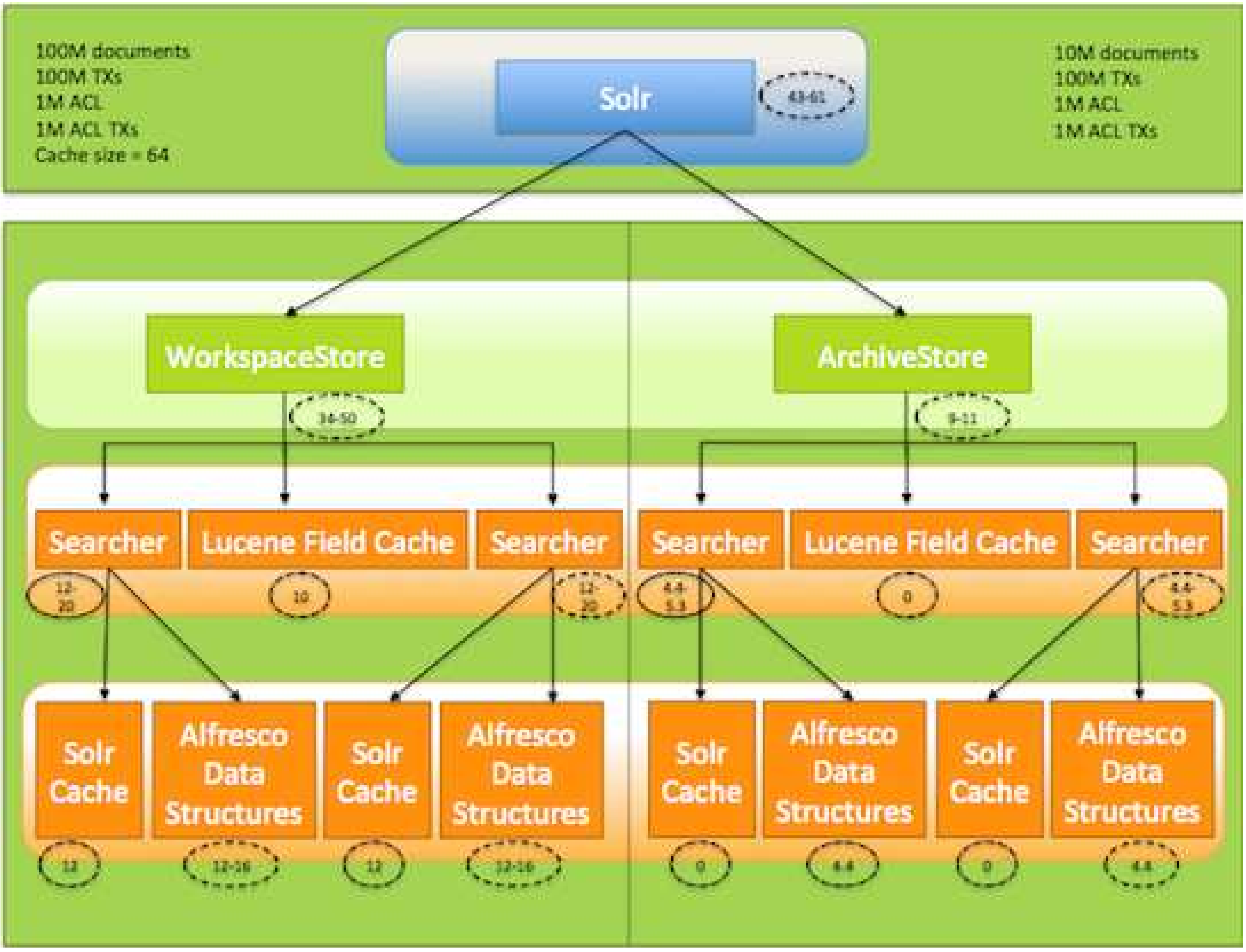
This example is based on the data above.

For WorkspaceStore: Assuming that there are 100M docs, 100M TXs, 1M ACLs and ACL TXs, cache size of 64 entries each for FilterCache, AuthorityCache and QCache, and 128 entries for PathCache, between 12 to 20G of memory is needed per searcher. Normally, there is one searcher live but around commit time there can be two searchers. So, approximately 34 to 50G of memory will be needed in total.

For Archivestore: Assuming that there are 100M transactions, 10M docs and all caches are tuned down, between 4.4G to 5.3G of memory is needed per searcher. Total memory needed for both the searchers will be between 9G to 11G.

So, the total memory requirement for both the cores is between 43G to 61G.

The following diagram shows the overall memory use for a Solr node as explained above:



Minimize the memory requirements for Solr nodes

- Reduce the cache sizes and check the cache hit rate.
- Disable ACL checks.
- Disable archive indexing, if you are not using it.
- Check the number of empty transactions. If there are many empty transactions, purge the transactions from Alfresco using the `action=FIX` action.
- Find the exact number of nodes in the store (N), exact number of transactions in the repository (T), number of ACLs (A) and related ACL transactions in the repository (X).
- Since everything scales to the number of documents in the index, add the Index control aspect to the documents you do not want in the index.

Backing up and restoring

This section describes the process for backing up the Alfresco content repository. It assumes that the various binaries (operating system, database, JDK, application server, and so on.) and configuration files (operating system, database, JDK, application server, Alfresco, and so on) are being backed up independently.

Your backup strategy must be tested end-to-end, including restoration of backups that were taken previously. Ensure that you have adequately tested your backup scripts prior to deploying Alfresco to production.

Backing up and restoring the repository

- The directory pointed to by the `dir.root` setting
- The database Alfresco is configured to use

If Solr is being used, only the following directories must be backed up from the `dir.root` directory:

- `contentstore` directory
- `solr/workspace/` directory
- `solr/archive/` directory
- `contentstore.deleted` directory (optional)

If Lucene is being used, only the following directories must be backed up from the `dir.root` directory:

- `contentstore` directory
- `lucene-indexes` directory
- `contentstore.deleted` directory (optional)

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 (only the above mentioned directories) 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.

Performing a cold backup

1. Stop Alfresco.
2. Back up the database Alfresco is configured to use, using your database vendor's backup tools.
3. In parallel, back up the `dir.root` directory in its entirety.
4. Store both the database and `dir.root` backups together as a single unit.

   For example, store the backups in the same directory or compressed file.
5. Start Alfresco.

Performing a hot backup

The high-level procedure for a hot backup is:

1. Back up Lucene.
2. Back up SQL.
3. Back up files.

Lucene indexes have to be backed up first and before SQL because if new rows are added in SQL after the Lucene backup is done, a Lucene reindex (AUTO) can regenerate the missing Lucene indexes from the SQL transaction data.

SQL has to be done before backing up the files because if you have a SQL node pointing to a missing file, then that node will be an orphan. Also, if you have a file without SQL node data, this means that the user has added the file too late to be included in a backup.

It is critical to perform hot backups in the following order of steps:

1. Ensure that you have a `backup-lucene-indexes` directory under `dir.root`.
2. Backup the database Alfresco is configured to use, using your database vendor's backup tools.
3. As soon as the database backup completes, backup specific subdirectories in the file system Alfresco `dir.root`.
4. Store both the database and Alfresco `dir.root` backups together as a single unit.

   For example, store the backups in the same directory or in a single compressed file. Do not store the database and `dir.root` backups independently, as that makes it difficult to reconstruct a valid backup set, if restoration becomes necessary.
5. Store both the database and `dir.root` backups together as a single unit.

   Ensure that the cron generated in the `backup-lucene-indexes` does not run while you do the SQL backup. The `backup-lucene-indexes` generation should be finished when you start the SQL backup.

   If you set `system.content.eagerOrphanCleanup=true` in the `alfresco-global.properties` file, performing a hot backup is not recommended. Actions taken between the database backup and the file system backup have the potential to cause orphaned nodes in the repository.

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.

Refreshing the backup Lucene indexes (optional)

1. Trigger a Lucene index backup using JMX, and it can be done several ways, including using:

   The backup can be done in several ways, including:

   - VisualVM
   - JConsole (MBeans tab `-gt Alfresco/Schedule/DEFAULT/MonitoredCronTrigger/indexBackupTrigger/Operations` - **executeNow** button)
   - Command line

   During the creation of the backup Lucene indexes, the system is placed in read-only mode, which could last several minutes depending on the size of the Lucene indexes.
2. After completing this operation, the `backup-lucene-indexes` directory contains an up-to-date cold copy of the Lucene indexes, ready to be backed up.

Backing up the database

Database hot backup requires a tool that can "snapshot" a consistent version of the Alfresco database (that is, it must capture a transactionally-consistent copy of all the tables in the Alfresco database). In addition, to avoid serious performance problems in the running Alfresco system while the backup is in progress, this "snapshot" operation should either operate without taking out locks in the Alfresco database or it should complete quickly (within seconds).

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.

Backing up the file system

Backup the following subdirectories of the Alfresco `dir.root` directory using whatever tools you are comfortable with (`rsync`, `xcopy`):

- `contentstore`
- `contentstore.deleted`
- `audit.contentstore`
- `backup-lucene-indexes`

Do not attempt to backup the `lucene-indexes` subdirectory while Alfresco is running. This will cause Lucene index corruption. Use `backup-lucene-indexes` instead.

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.

Changing the scheduled Lucene back up time

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. Rename the file to `custom-scheduled-jobs-context.xml`.

   As your override file ends with `-context.xml`, you do not need to point to your file.
4. 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>
```

   The default is to run the job at 3am every day.
5. Modify the `cronExpression` values, if required

   The value `0 0 3 * * ?` specifies the backup is triggered at 3am every day.

   After each backup scheduled in the `indexBackupTrigger` bean, perform your normal backup of this directory.

   Each backup will overwrite the existing backup.

Specifying the Lucene backup directory

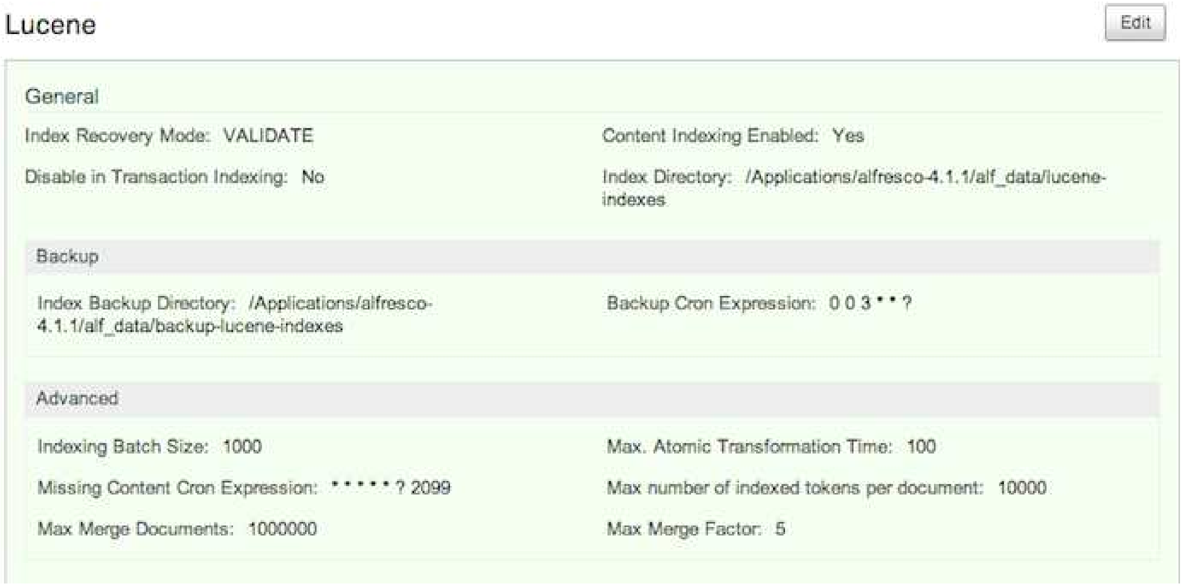
You can set the Lucene backup directory using the following three ways: by using the Admin Console in Share, by editing the `alfresco-global.properties` file or by using a JMX client, such as JConsole.

Set up Lucene backup directory using Share Admin Console

You can only see the Admin Console if you are an administrator user.

1. On the toolbar, expand the **More** menu, and then click **More** in the **Admin Tools...** list.
2. Under the **Tools** section on the left navigation bar, you see various tools available and the options that you can set. In the **Search** sub-section, click on **Lucene**.

The **Lucene** window is displayed.



Lucene

General

Index Recovery Mode: VALIDATE

Content Indexing Enabled: Yes

Disable in Transaction Indexing: No

Index Directory: /Applications/alfresco-4.1.1/alf_data/lucene-indexes

Backup

Index Backup Directory: /Applications/alfresco-4.1.1/alf_data/backup-lucene-indexes

Backup Cron Expression: 0 0 3 * * ?

Advanced

Indexing Batch Size: 1000

Max. Atomic Transformation Time: 100

Missing Content Cron Expression: * * * * * ? 2099

Max number of indexed tokens per document: 10000

Max Merge Documents: 1000000

Max Merge Factor: 5

3. Click **Edit**.
4. Edit the backup properties for the Lucene index by specifying when the backup occurs in the **Backup Cron Expression** text box.
5. Specify the full path on the Alfresco server file system to store the index backup in the **Index Backup Directory** text box.
6. Click **Save**.

Specifying Lucene backup directory via `alfresco-global.properties` file

This task shows how to specify the Lucene backup directory via `alfresco-global.properties` file.

- To set the Lucene backup directory and schedule, using the `alfresco-global.properties` file, set the value of the following properties to the relevant cron expression and the full path where the backups should be kept:

```
index.backup.cronExpression=0 0 3 * * ?
dir.indexes.backup=${dir.root}/backup-lucene-indexes
```

Specifying Lucene backup directory via JMX client

You can use the JMX client, JConsole to specify the backup directory for Lucene indexes.

- To use JMX client to setup Lucene backup directory, navigate to **MBeans tab > Alfresco > Configuration > Search > managed > lucene > Attributes** and change the values for `index.backup.cronExpression` and `dir.indexes.backup` properties.

Restoring the Lucene indexes

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-indexes` 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. As the size of your content set grows, the time savings from performing incremental reindexing rather than full reindexing will become greater and greater. Incremental reindexing is typically measured in minutes, whereas full reindexing can take hours for large content sets.

   For incremental reindexing to occur properly, set the `index.recovery.mode` property to `AUTO` to ensure that the restored Lucene indexes are incrementally built with any newer transactions in the database and contentstore. 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.

   When running Alfresco with Solr as the default search subsystem, you need to change the `index.recovery.mode` property to `AUTO` before switching to Lucene subsystem for creating Lucene indexes.

Performing a full hot reindex on a cluster

This process applies to Lucene indexes only.

1. Change your load balancer configuration to prevent any user traffic from being directed to node 1.
2. Shut down Alfresco on node 1.
3. Remove the Lucene indexes on node 1 (`alf_home_dir/alf_data/lucene-indexes`).
4. Make the following changes in the `alfresco-global.properties` file on node 1:

   ```
   system.cache.disableMutableSharedCaches=true
   alfresco.cluster.name=temporaryclustername
   ```

   You can set the value of `alfresco.cluster.name` parameter to anything, provided that cluster name is not in use by any other Alfresco nodes on the same network.
5. Start Alfresco on node 1. It takes some time for indexing to complete. The following message is displayed in the log files:

   ```
   18:14:44,806 INFO [node.index.FullIndexRecoveryComponent] 100 %
    complete.
   18:14:44,992 INFO [node.index.FullIndexRecoveryComponent] Index recovery
    completed.
   ```

   This message does not indicate that full reindex is finished, it shows that the full reindex of the metadata is finished. Node 1 will then index the content of the documents in the background. You need to wait till this process is finished. To confirm that the content is indexed, you could monitor the CPU activity - it will be consistently high until the indexing is finished, at which point it should drop sharply.
6. Shut down Alfresco on node 1.
7. Make a backup copy of the indexes on node 1.
8. Revert the configuration changes made in Step 4. To do so, remove the `system.cache.disableMutableSharedCaches=true` setting and change `alfresco.cluster.name` back to what it was previously.
9. Start up Alfresco on node 1.
10. Change your load balancer configuration to allow user traffic back to node 1 and stop traffic from going to node 2.
11. Shut down Alfresco on node 2.
12. Delete the Lucene indexes on node 2.

13. Copy the backup made in Step 7 into the lucene-indexes directory on node 2.
14. Start Alfresco on node 2.
15. There may be a small amount of additional top-up indexing required on node 2. Once this indexing has finished (as per Step 5), change your load balancer configuration to allow traffic to node 2.

Using the Bulk Import tool

The Bulk Import Tool provides a mechanism for bulk importing existing content into a repository from the Alfresco server's filesystem.

It will (optionally) replace existing content items if they already exist in the repository, but does not perform deletes (it is not designed to fully synchronize the repository with the local filesystem). The basic on-disk file/folder structure is preserved verbatim in the repository. It is possible to load metadata for the files and spaces being ingested, as well as a version history for files (each version may consist of content, metadata, or both).

There are two types of bulk import:

- Streaming import: This import streams the files into the repository content store by copying them in during the import.
- In-place import: Available in Enterprise Only, these files are assumed to already exist within the repository content store, so no copying is required. This can result in a significant improvement in performance .

There are a number of restrictions:

- No support for AVM.
- Only one bulk import can be running at a time. This is enforced by the JobLockService.
- Access to the Bulk Import tool is restricted to Alfresco administrators, by default.

In-Place bulk import

The In-Place import is available in Enterprise Only, and imports files that already exist within the repository content store. As no copying is required, this can result in a significant performance improvement.

Three assumptions are made when importing content "in place".

- The content is already at its initial repository location prior to import, as it will be *not* be moved during the import.
- The in-place content must be within the tree structure of a registered content store, as defined by either:
  - the default fileContentStore
  - a filesystem-based store defined by the content store selector
- Steps have already been taken prior to import to ensure the content structure is well distributed.
  - The default fileContentStore distributes content, based on the import date (year/month/day/hour/minute). This avoids having thousands of file under the same root, which is inefficient both for the file system and for computing parent associations in Alfresco (among other things).
  - It is recommended you keep immediate children to a few thousands maximum.

- In order to choose an efficient distribution scheme, you should know that, when m files are randomly distributed into n leaf folders, when m >> n log n the statistical maximum load of a leaf is m/n + O(sqrt((m log n)/n)).

In addition, the In-Place bulk import provides support for the Managing the content store. This allows you to select under which store the content to import is to be found.

Streaming Bulk Import

The Streaming bulk import copies the source content into the repository content store.

In all other respects, in-place and streaming bulk import are the same.

Preparing the filesystem

There are a number of things you must do to prepare the filesystem before you do the bulk import.

Metadata files

The Bulk Import tool has the ability to load metadata (types, aspects, and their properties) into the repository. This is accomplished using "shadow" Java property files in XML format as it has good support for Unicode characters. These shadow properties files must have exactly the same name and extension as the file for which it describes the metadata, but with the suffix ".metadata.properties.xml". For example, if there is a file called "IMG_1967.jpg", the "shadow" metadata file is called "IMG_1967.jpg.metadata.properties.xml".

These shadow files can also be used for directories. For example, if you have a directory called "MyDocuments", the shadow metadata file is called "MyDocuments.metadata.properties.xml".

The metadata file itself follows the usual syntax for Java XML properties files:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
   <entry key="key1">value1</entry>
    entry key="key2">value2</entry>
    ...
</properties>
```

There are two special keys:

- type - contains the qualified name of the content type to use for the file or folder
- aspects - contains a comma-delimited list of the qualified names of the aspect(s) to attach to the file or folder

The remaining entries in the file are treated as metadata properties, with the key being the qualified name of the property and the value being the value of that property. Multi-valued properties are comma-delimited. However, these values are not trimmed so it's recommended you do not place a space character either before or after the comma, unless you want that in the value of the property.

Here's an example using IMG_1967.jpg.metadata.properties.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
   <entry key="type">cm:content</entry>
   <entry key="aspects">cm:versionable,cm:dublincore</entry>
   <entry key="cm:title">A photo of a flower.</entry>
   <entry key="cm:description">A photo I took of a flower while walking around
Bantry Bay.</entry>
   <entry key="cm:created">1901-01-01T12:34:56.789+10:00</entry>
   <!-- cm:dublincore properties -->
   <entry key="cm:author">Peter Monks</entry>
   <entry key="cm:publisher">Peter Monks</entry>
   <entry key="cm:contributor">Peter Monks</entry>
   <entry key="cm:type">Photograph</entry>
   <entry key="cm:identifier">IMG_1967.jpg</entry>
   <entry key="cm:dcsource">Canon Powershot G2</entry>
   <entry key="cm:coverage">Worldwide</entry>
   <entry key="cm:rights">Copyright (c) Peter Monks 2002, All Rights
Reserved</entry>
   <entry key="cm:subject">A photo of a flower.</entry>
 </properties>
```

Additional notes on metadata loading:

- You cannot create a new node based on metadata only, you must have a content file (even if zero bytes) for the metadata to be loaded. That said, you can "replace" an existing node in the repository with nothing but metadata. Despite the confusing name, this won't replace the content; it will simply be decorated with the new metadata.
- The metadata must conform to the type and aspect definitions configured in Alfresco (including mandatory fields, constraints, and data types). Any violations will terminate the bulk import process.
- Associations between content items loaded by the tool are not yet nicely supported. Associations to objects that are already in the repository can be created using the NodeRef of the target object as the value of the property.
- Non-string data types (including numeric and date types) have not been exhaustively tested. Date values have been tested and do work when specified using ISO8601 format.
- Updating the aspects or metadata on existing content will not remove any existing aspects not listed in the new metadata file; this tool is not intended to provide a full filesystem synchronisation mechanism.
- The metadata loading facility can be used to decorate content that's already in the Alfresco repository, without having to upload that content again. To use this, create a "naked" metadata file in the same path as the target content file. The tool will match it up with the file in the repository and decorate that existing file with the new aspect(s) and/or metadata

Version History files

The import tool also supports loading a version history for each file. To do this, create a file with the same name as the main file, but append it with a "v#" extension. For example:

```
 IMG_1967.jpg.v1   <- version 1 content
 IMG_1967.jpg.v2   <- version 2 content
 IMG_1967.jpg      <- "head" (latest) revision of the content
```

This also applies to metadata files if you want to capture metadata history as well. For example:

```
 IMG_1967.jpg.metadata.properties.xml.v1   <- version 1 metadata
 IMG_1967.jpg.metadata.properties.xml.v2   <- version 2 metadata
 IMG_1967.jpg.metadata.properties.xml      <- "head" (latest) revision of the
metadata
```

Additional notes on version history loading:

- You cannot create a new node based on a version history only. You must have a head revision of the file.
- Version numbers don't have to be contiguous. You can number your version files however you want, provided you use whole numbers (integers).
- The version numbers in your version files will not be used in Alfresco. The version numbers in Alfresco will be contiguous, starting at 1.0 and increasing by 1.0 for every version (so 1.0, 2.0, 3.0, etc. etc.). Alfresco doesn't allow version labels to be set to arbitrary values, and currently the bulk import doesn't provide any way to specify whether a given version should have a major or minor increment.

- Each version can contain a content update, a metadata updat,e or both - you are not limited to updating everything for every version. If not included in a version, the prior version's content or metadata will remain in place for the next version.

The following example shows all possible combinations of content, metadata, and version files:

```
 IMG_1967.jpg.v1                           <- version 1 content
 IMG_1967.jpg.metadata.properties.xml.v1   <- version 1 metadata
 IMG_1967.jpg.v2                           <- version 2 content
 IMG_1967.jpg.metadata.properties.xml.v2   <- version 2 metadata
 IMG_1967.jpg.v3                           <- version 3 content (content only
version)
 IMG_1967.jpg.metadata.properties.xml.v4   <- version 4 metadata (metadata
only version)
 IMG_1967.jpg.metadata.properties.xml      <- "head" (latest) revision of the
metadata
 IMG_1967.jpg                              <- "head" (latest) revision of the
content
```

Importing via the user interface

Streaming

The Streaming bulk import is exposed in two web scripts:

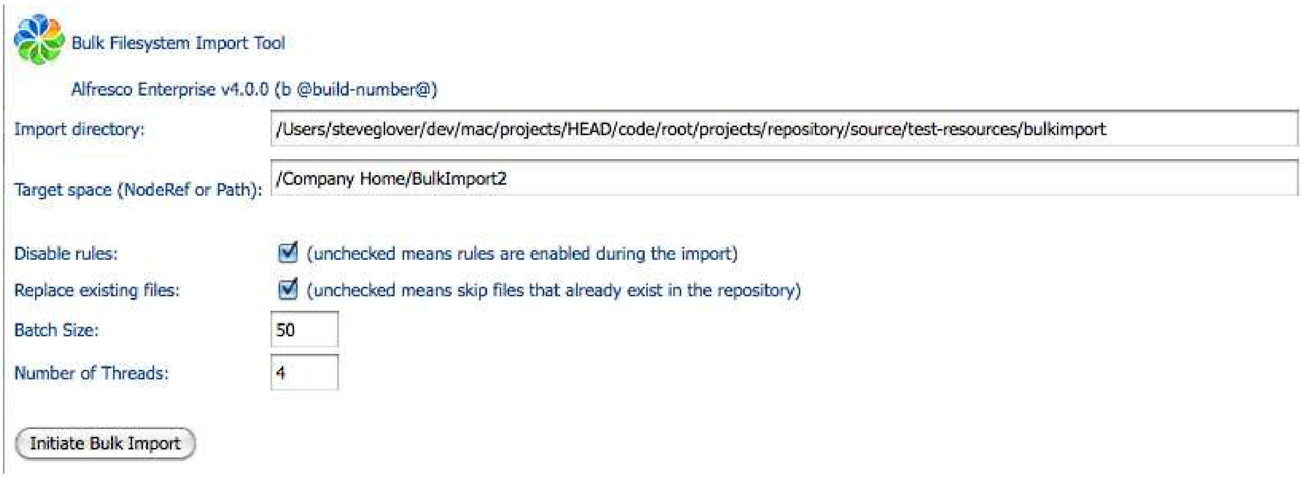
1. A simple UI web script that can be used to manually initiate an import. This is an HTTP GET web script with a path of:

   ```
   http://localhost:8080/alfresco/service/bulkfsimport
   ```

2. An Initiate web script, which kicks off an import using parameters that are passed to it (for the source directory, target space, and so on). If you want to script or invoke the tool programmatically, this is the web script you call. This is an HTTP GET web script with a path of:

   ```
   http://localhost:8080/alfresco/service/bulkfsimport/initiate
   ```

The UI web script presents the following simplified HTML form:



Bulk Filesystem Import Tool

Alfresco Enterprise v4.0.0 (b @build-number@)

Import directory: /Users/steveglover/dev/mac/projects/HEAD/code/root/projects/repository/source/test-resources/bulkimport

Target space (NodeRef or Path): /Company Home/BulkImport2

Disable rules: ☑ (unchecked means rules are enabled during the import)

Replace existing files: ☑ (unchecked means skip files that already exist in the repository)

Batch Size: 50

Number of Threads: 4

- The 'Import directory' field is required and indicates the absolute filesystem directory to load the content and spaces from, in an OS-specific format. Note that this directory must be locally accessible to the server the Alfresco instance is running on. It must either be a local filesystem or a locally mounted remote filesystem (mounted using NFS, GFS, CIFS, or similar).
- The 'Target space' field is also required and indicates the target space to load the content into, as a path starting with "/Company Home". The separator character is Unix-style ("/"),

regardless of the platform Alfresco is running on. This field includes an AJAX auto-suggest feature, so you may type any part of the target space name, and an AJAX search will be performed to find and display matching items.

- The 'Replace existing files' checkbox indicates whether to replace nodes that already exist in the repository (checked) or skip them (unchecked). Note that if versioning is enabled for a node, the node's existing content and metadata will be preserved as the prior version and the new content and/or metadata will be written into the head revision.
- The "Number of Threads" text field allows you to override the default number of threads (defined by the property "bulkImport.batch.numThreads") to use in the bulk import.
- The "Batch Size" text field allows you to override the default batch size (the number of directories and files to import at a time, per transaction; defined by the property "bulkImport.batch.batchSize") to use in the bulk import.
- The "Disable rules" checkbox allows you to turn off rule processing during the bulk import.

In-Place

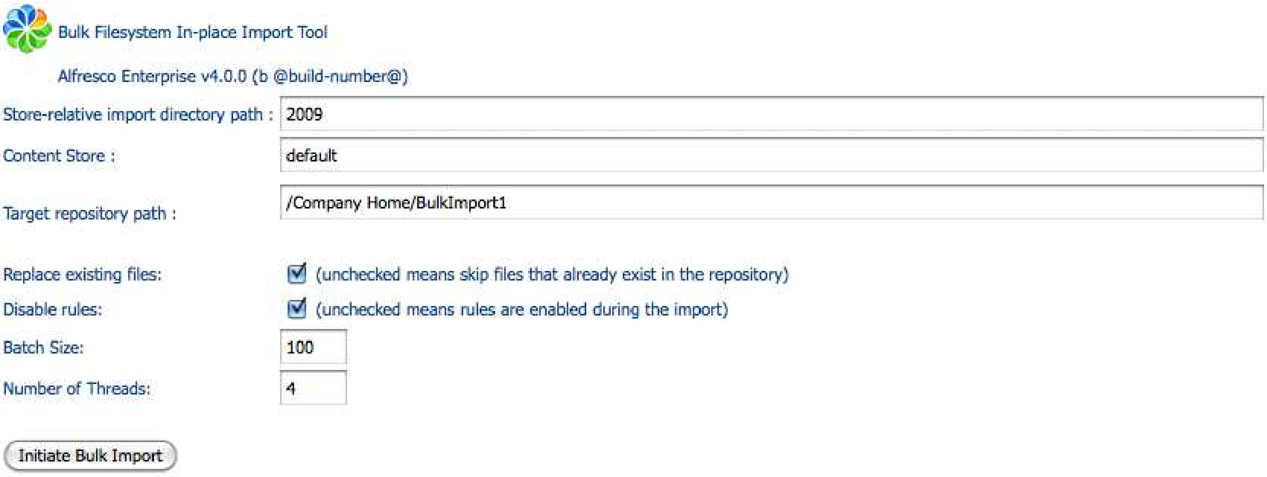
The In-Place bulk import is exposed as a series of two web scripts:

1. A simple UI web script that can be used to manually initiate an import. This is an HTTP GET web script with a path of:

   ```
   http://localhost:8080/alfresco/service/bulkfsimport/inplace
   ```
2. An Initiate web script that actually kicks off an import, using parameters that are passed to it (for the source directory, target space, etc.). If you want to script or programmatically invoke the tool, this is the web script you call. This is an HTTP GET web script with a path of:

   ```
   http://localhost:8080/alfresco/service/bulkfsimport/inplace/initiate
   ```

The In-Place UI web script presents the following simplified HTML form:



Bulk Filesystem In-place Import Tool

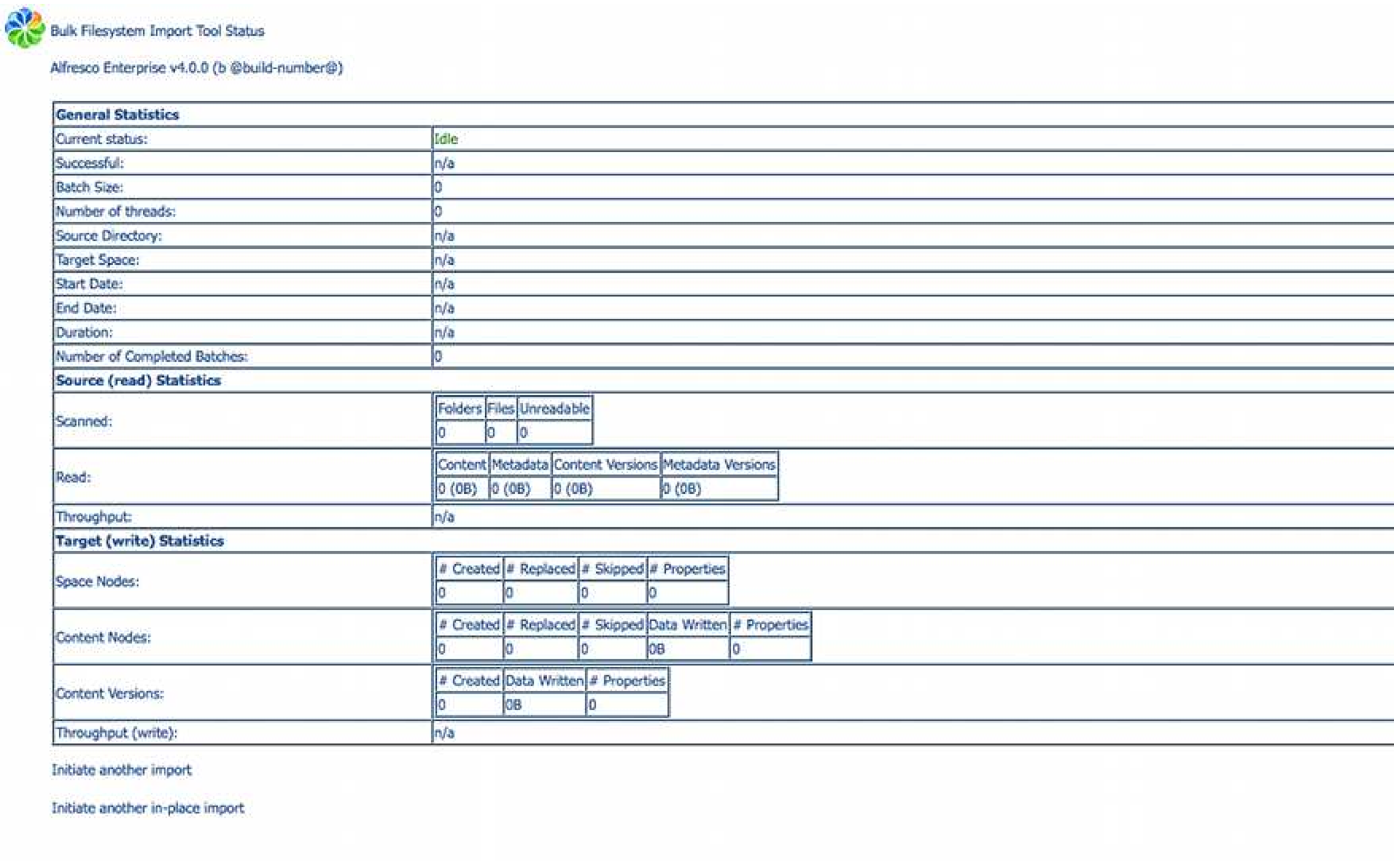
Alfresco Enterprise v4.0.0 (b @build-number@)

Store-relative import directory path :	2009
Content Store :	default
Target repository path :	/Company Home/BulkImport1
Replace existing files:	☑ (unchecked means skip files that already exist in the repository)
Disable rules:	☑ (unchecked means rules are enabled during the import)
Batch Size:	100
Number of Threads:	4

- The 'Import directory' field is required and indicates the absolute filesystem directory to load the content and spaces from, in an OS-specific format. Note that this directory must be locally accessible to the server the Alfresco instance is running on - it must either be a local filesystem or a locally mounted remote filesystem (mounted using NFS, GFS, CIFS or similar). This directory must already be inside an existing contentstore.

- The content store name that holds the content, as defined within the storage configuration (content store selector or direct fileContentStore). The default store is by default named "default". An autocomplete popup will assist in selecting the name as the first characters are entered. The 'Up' and 'Down' keyboards keys can be used to navigate the list, in addition to the mouse.
- The 'Target space' field is also required and indicates the target space to load the content into, as a path starting with "/Company Home". The separator character is Unix-style ("/"), regardless of the platform Alfresco is running on. This field includes an AJAX auto-suggest feature, so you may type any part of the target space name, and an AJAX search will be performed to find and display matching items.
- The 'Replace existing files' checkbox indicates whether to replace nodes that already exist in the repository (checked) or skip them (unchecked). Note that if versioning is enabled for a node, the node's existing content and metadata will be preserved as the prior version and the new content and/or metadata will be written into the head revision.
- The "Number of Threads" text field allows you to override the default number of threads (defined by the property "bulkImport.batch.numThreads") to use in the bulk import.
- The "Batch Size" text field allows you to override the default batch size (the number of directories and files to import at a time, per transaction; defined by the property "bulkImport.batch.batchSize") to use in the bulk import.
- The "Disable rules" checkbox allows you to turn off rule processing during the bulk import.

The status web page is the same for both streaming and In-Place import. It displays as follows:



Bulk Filesystem Import Tool Status

Alfresco Enterprise v4.0.0 (b @build-number@)

<table>
<tr><th colspan="2">General Statistics</th></tr>
<tr><td>Current status:</td><td>Idle</td></tr>
<tr><td>Successful:</td><td>n/a</td></tr>
<tr><td>Batch Size:</td><td>0</td></tr>
<tr><td>Number of threads:</td><td>0</td></tr>
<tr><td>Source Directory:</td><td>n/a</td></tr>
<tr><td>Target Space:</td><td>n/a</td></tr>
<tr><td>Start Date:</td><td>n/a</td></tr>
<tr><td>End Date:</td><td>n/a</td></tr>
<tr><td>Duration:</td><td>n/a</td></tr>
<tr><td>Number of Completed Batches:</td><td>0</td></tr>
<tr><th colspan="2">Source (read) Statistics</th></tr>
<tr><td>Scanned:</td><td><table><tr><td>Folders</td><td>Files</td><td>Unreadable</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table></td></tr>
<tr><td>Read:</td><td><table><tr><td>Content</td><td>Metadata</td><td>Content Versions</td><td>Metadata Versions</td></tr><tr><td>0 (0B)</td><td>0 (0B)</td><td>0 (0B)</td><td>0 (0B)</td></tr></table></td></tr>
<tr><td>Throughput:</td><td>n/a</td></tr>
<tr><th colspan="2">Target (write) Statistics</th></tr>
<tr><td>Space Nodes:</td><td><table><tr><td># Created</td><td># Replaced</td><td># Skipped</td><td># Properties</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr></table></td></tr>
<tr><td>Content Nodes:</td><td><table><tr><td># Created</td><td># Replaced</td><td># Skipped</td><td>Data Written</td><td># Properties</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0B</td><td>0</td></tr></table></td></tr>
<tr><td>Content Versions:</td><td><table><tr><td># Created</td><td>Data Written</td><td># Properties</td></tr><tr><td>0</td><td>0B</td><td>0</td></tr></table></td></tr>
<tr><td>Throughput (write):</td><td>n/a</td></tr>
</table>

Initiate another import

Initiate another in-place import

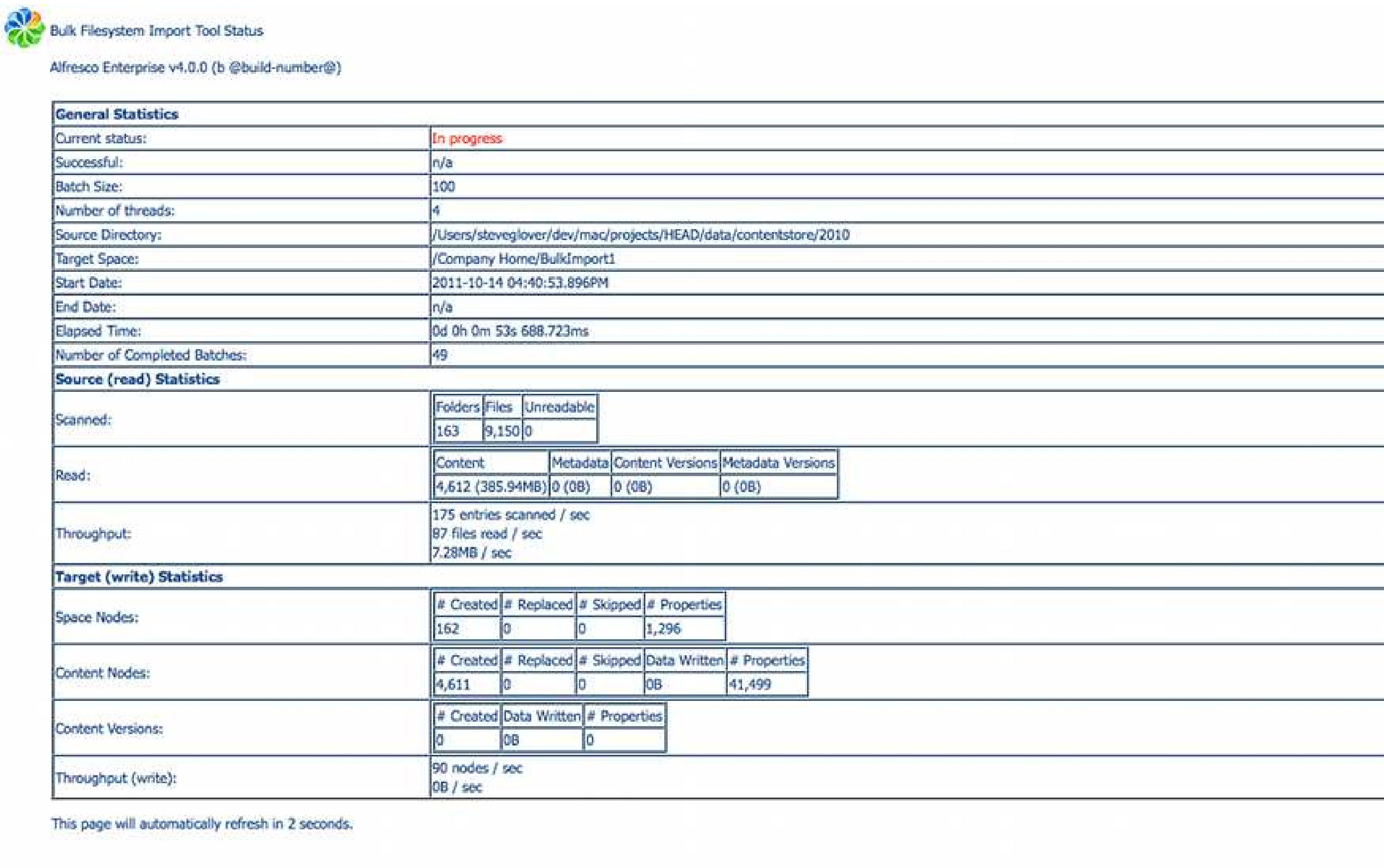
Bulk Import status

The Bulk Import Status web script returns status information on the current import (if one is in progress), or the status of the last import that was initiated. This web script has both HTML and

XML views, allowing external programs to programmatically monitor the status of imports. This is an HTTP GET web script with a path of:

```
http://localhost:8080/alfresco/service/bulkfsimport/status
```

The status web page is the same for both Streaming and In-Place import. The status is updated every five seconds when a bulk import has been initiated, as follows.



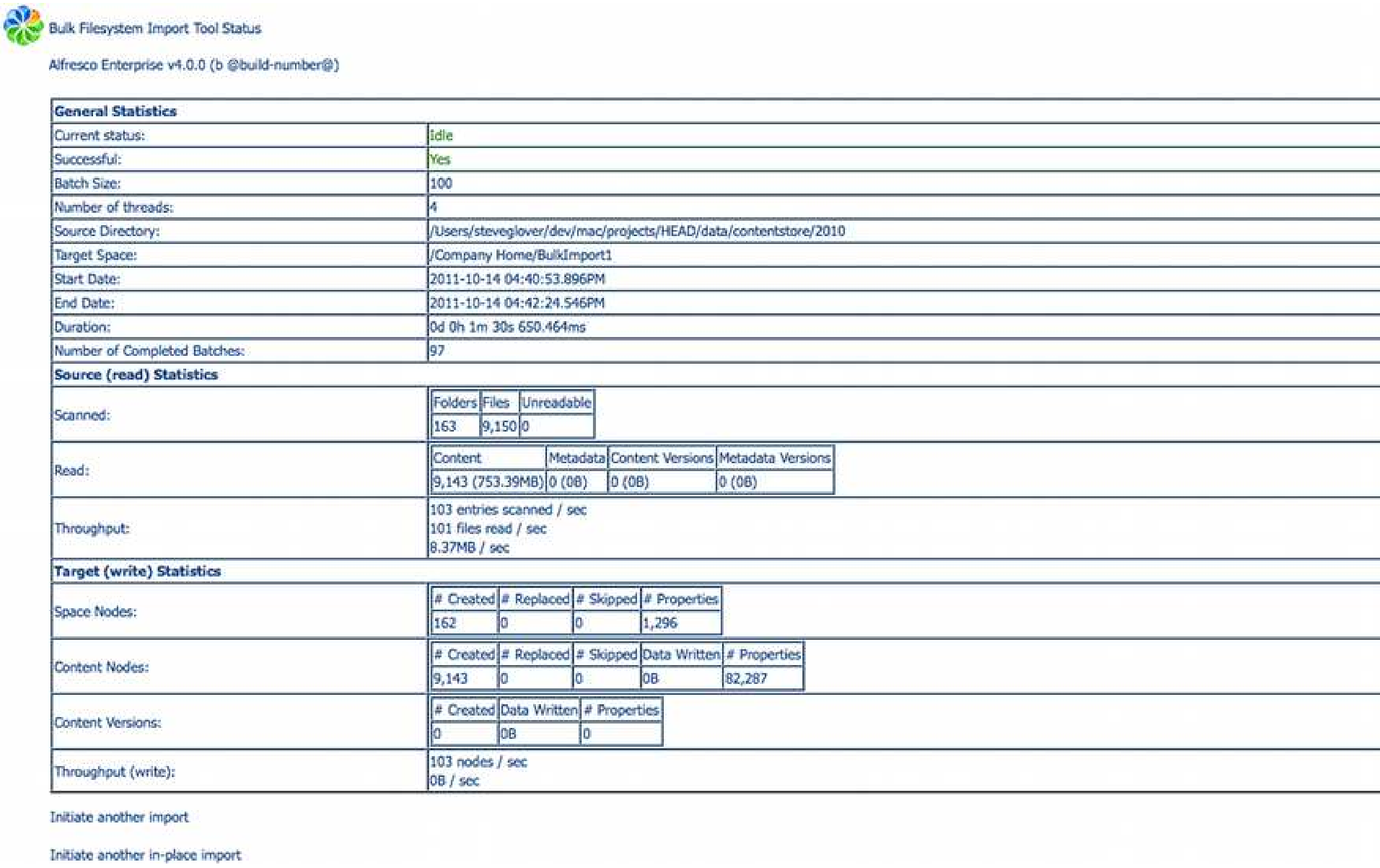
Bulk Filesystem Import Tool Status

Alfresco Enterprise v4.0.0 (b @build-number@)

<table>
<tr><th colspan="2">General Statistics</th></tr>
<tr><td>Current status:</td><td>In progress</td></tr>
<tr><td>Successful:</td><td>n/a</td></tr>
<tr><td>Batch Size:</td><td>100</td></tr>
<tr><td>Number of threads:</td><td>4</td></tr>
<tr><td>Source Directory:</td><td>/Users/steveglover/dev/mac/projects/HEAD/data/contentstore/2010</td></tr>
<tr><td>Target Space:</td><td>/Company Home/BulkImport1</td></tr>
<tr><td>Start Date:</td><td>2011-10-14 04:40:53.896PM</td></tr>
<tr><td>End Date:</td><td>n/a</td></tr>
<tr><td>Elapsed Time:</td><td>0d 0h 0m 53s 688.723ms</td></tr>
<tr><td>Number of Completed Batches:</td><td>49</td></tr>
<tr><th colspan="2">Source (read) Statistics</th></tr>
<tr><td>Scanned:</td><td><table><tr><td>Folders</td><td>Files</td><td>Unreadable</td></tr><tr><td>163</td><td>9,150</td><td>0</td></tr></table></td></tr>
<tr><td>Read:</td><td><table><tr><td>Content</td><td>Metadata</td><td>Content Versions</td><td>Metadata Versions</td></tr><tr><td>4,612 (385.94MB)</td><td>0 (0B)</td><td>0 (0B)</td><td>0 (0B)</td></tr></table></td></tr>
<tr><td>Throughput:</td><td>175 entries scanned / sec
87 files read / sec
7.28MB / sec</td></tr>
<tr><th colspan="2">Target (write) Statistics</th></tr>
<tr><td>Space Nodes:</td><td><table><tr><td># Created</td><td># Replaced</td><td># Skipped</td><td># Properties</td></tr><tr><td>162</td><td>0</td><td>0</td><td>1,296</td></tr></table></td></tr>
<tr><td>Content Nodes:</td><td><table><tr><td># Created</td><td># Replaced</td><td># Skipped</td><td>Data Written</td><td># Properties</td></tr><tr><td>4,611</td><td>0</td><td>0</td><td>0B</td><td>41,499</td></tr></table></td></tr>
<tr><td>Content Versions:</td><td><table><tr><td># Created</td><td>Data Written</td><td># Properties</td></tr><tr><td>0</td><td>0B</td><td>0</td></tr></table></td></tr>
<tr><td>Throughput (write):</td><td>90 nodes / sec
0B / sec</td></tr>
</table>

This page will automatically refresh in 2 seconds.

When the bulk import has completed, it displays as follows:



Bulk Filesystem Import Tool Status

Alfresco Enterprise v4.0.0 (b @build-number@)

<table>
<tr><th colspan="2">General Statistics</th></tr>
<tr><td>Current status:</td><td>idle</td></tr>
<tr><td>Successful:</td><td>Yes</td></tr>
<tr><td>Batch Size:</td><td>100</td></tr>
<tr><td>Number of threads:</td><td>4</td></tr>
<tr><td>Source Directory:</td><td>/Users/steveglover/dev/mac/projects/HEAD/data/contentstore/2010</td></tr>
<tr><td>Target Space:</td><td>/Company Home/BulkImport1</td></tr>
<tr><td>Start Date:</td><td>2011-10-14 04:40:53.896PM</td></tr>
<tr><td>End Date:</td><td>2011-10-14 04:42:24.546PM</td></tr>
<tr><td>Duration:</td><td>0d 0h 1m 30s 650.464ms</td></tr>
<tr><td>Number of Completed Batches:</td><td>97</td></tr>
<tr><th colspan="2">Source (read) Statistics</th></tr>
<tr><td>Scanned:</td><td><table><tr><td>Folders</td><td>Files</td><td>Unreadable</td></tr><tr><td>163</td><td>9,150</td><td>0</td></tr></table></td></tr>
<tr><td>Read:</td><td><table><tr><td>Content</td><td>Metadata</td><td>Content Versions</td><td>Metadata Versions</td></tr><tr><td>9,143 (753.39MB)</td><td>0 (0B)</td><td>0 (0B)</td><td>0 (0B)</td></tr></table></td></tr>
<tr><td>Throughput:</td><td>103 entries scanned / sec
101 files read / sec
8.37MB / sec</td></tr>
<tr><th colspan="2">Target (write) Statistics</th></tr>
<tr><td>Space Nodes:</td><td><table><tr><td># Created</td><td># Replaced</td><td># Skipped</td><td># Properties</td></tr><tr><td>162</td><td>0</td><td>0</td><td>1,296</td></tr></table></td></tr>
<tr><td>Content Nodes:</td><td><table><tr><td># Created</td><td># Replaced</td><td># Skipped</td><td>Data Written</td><td># Properties</td></tr><tr><td>9,143</td><td>0</td><td>0</td><td>0B</td><td>82,287</td></tr></table></td></tr>
<tr><td>Content Versions:</td><td><table><tr><td># Created</td><td>Data Written</td><td># Properties</td></tr><tr><td>0</td><td>0B</td><td>0</td></tr></table></td></tr>
<tr><td>Throughput (write):</td><td>103 nodes / sec
0B / sec</td></tr>
</table>

Initiate another import

Initiate another in-place import

Importing programmatically

The following code snippets show you how to complete a bulk import programmatically.

Streaming

```
   UserTransaction txn = transactionService.getUserTransaction();
   txn.begin();

   AuthenticationUtil.setRunAsUser("admin");

   StreamingNodeImporterFactory streamingNodeImporterFactory =
 (StreamingNodeImporterFactory)ctx.getBean("streamingNodeImporterFactory");
   NodeImporter nodeImporter = streamingNodeImporterFactory.getNodeImporter(new
 File("importdirectory"));
   BulkImportParameters bulkImportParameters = new BulkImportParameters();
   bulkImportParameters.setTarget(folderNode);
   bulkImportParameters.setReplaceExisting(true);
   bulkImportParameters.setBatchSize(40);
   bulkImportParameters.setNumThreads(4);
   bulkImporter.bulkImport(bulkImportParameters, nodeImporter);

   txn.commit();
```

In-Place

```
   txn = transactionService.getUserTransaction();
   txn.begin();

   AuthenticationUtil.setRunAsUser("admin");

   InPlaceNodeImporterFactory inPlaceNodeImporterFactory =
 (InPlaceNodeImporterFactory)ctx.getBean("inPlaceNodeImporterFactory");
```

```
  NodeImporter nodeImporter =
inPlaceNodeImporterFactory.getNodeImporter("default", "2011");
  BulkImportParameters bulkImportParameters = new BulkImportParameters();
  bulkImportParameters.setTarget(folderNode);
  bulkImportParameters.setReplaceExisting(true);
  bulkImportParameters.setBatchSize(150);
  bulkImportParameters.setNumThreads(4);
  bulkImporter.bulkImport(bulkImportParameters, nodeImporter);

  txn.commit();
```

Bulk Import diagnostics

To troubleshoot or diagnose any issues with the bulk importing, you can enable logging.

Enable debug statements with `log4j.logger.org.alfresco.repo.batch.BatchProcessor=info`.

You can also enable logging for retrying the transaction handler to pinpoint any transactional issues during the import. Enable this with `log4j.logger.org.alfresco.repo.transaction.RetryingTransactionHelper=info`.

Creating and managing workflows

What is a workflow?

For example, you might have a document that you needed reviewing and approving by a number of people. The sequence of connected tasks would be:

- Send an email to each reviewer asking the to review the document within a certain time
- Each reviewer reviews the document
- Each reviewer approves or rejects the document
- If enough reviewers approve, the task is completed successfully

Alfresco workflows automate the process for you. Users can choose from five workflow definitions provided in Alfresco. You can also create your own workflow definitions for more complex workflows. The five supplied workflow definitions are:

Adhoc
Enables you to assign a task to a single user

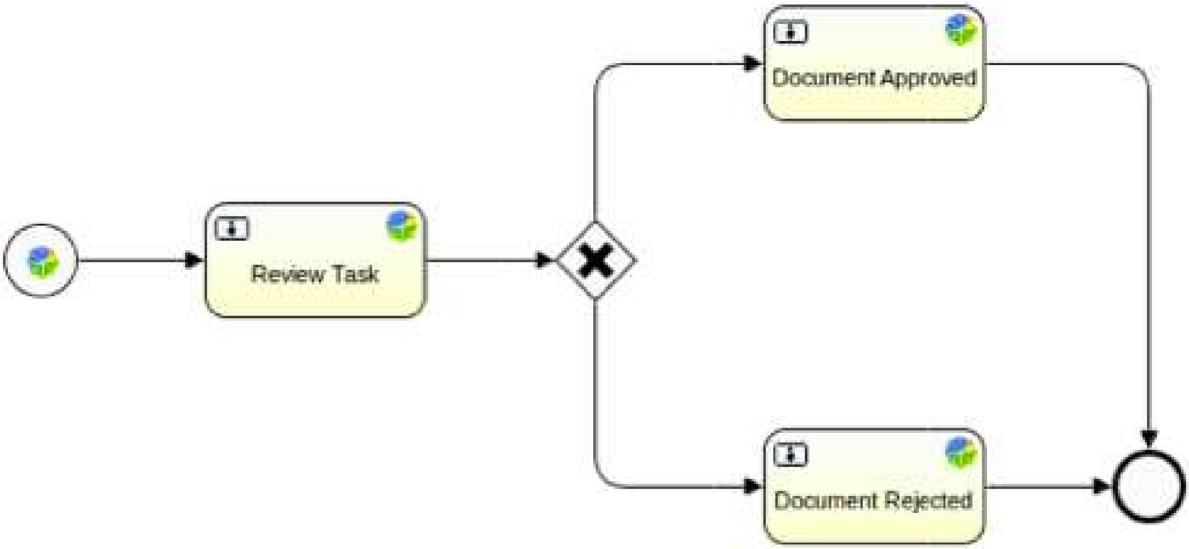
Group Review & Approve
Enables you to set up review and approval of content, assigning the workflow task to a single group

Parallel Review & Approve
Enables you to set up review and approval of content, assigning the workflow task to multiple users.

Pooled Review & Approve
Enables you to set up review and approval of content, assigning the workflow task to multiple users. One user can take ownership of the task at a time, completing it or returning it to the pool to be claimed by another user associated with the task.

Review & Approve
Enables you to set up review and approval of content, assigning the workflow task to a single user

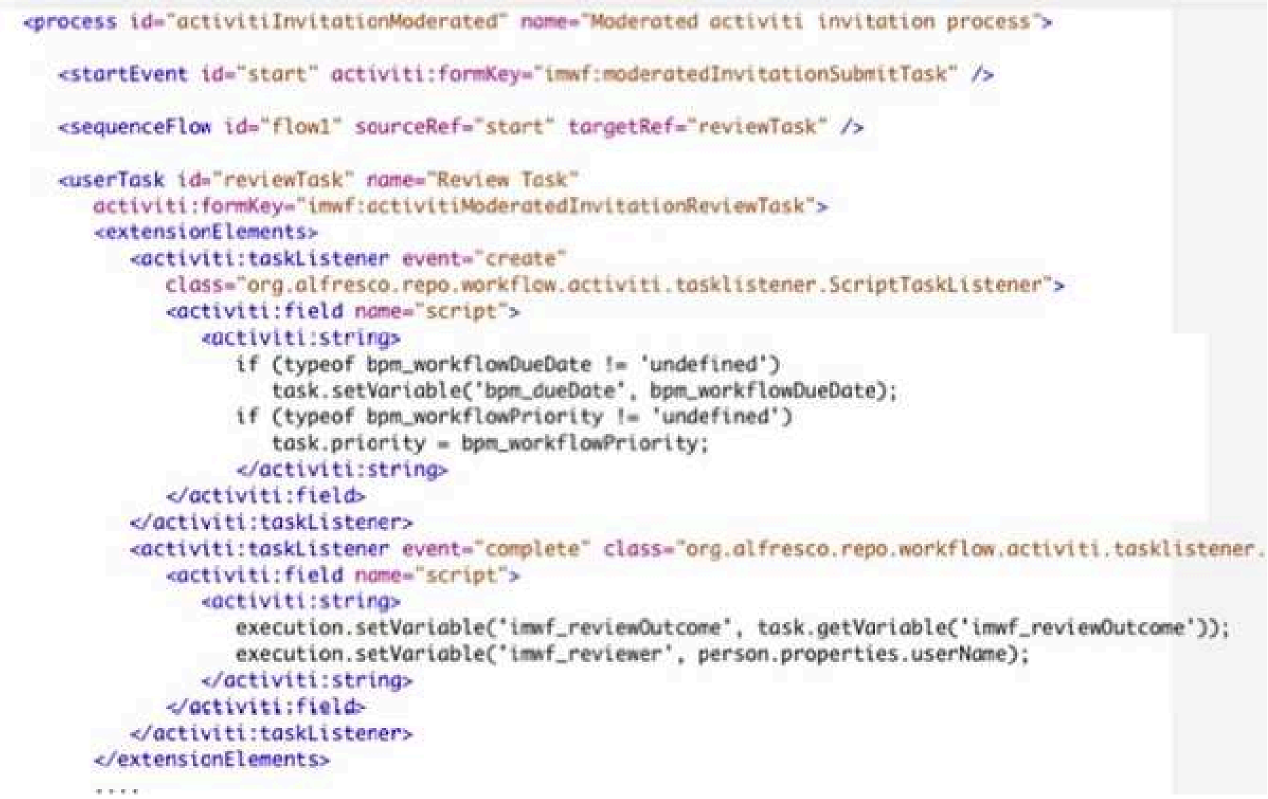
A graphical workflow modeler is often used to create a workflow. The following diagram shows a sample workflow taken from the workflow modeler running in Eclipse. The workflow consists of three tasks, a gate, and two events; start and end.



The Alfresco Activiti workflow engine executes BPMN 2.0 process definitions. BPNM 2.0 (Business Process Model and Notation) is an open standard developed by the Object Management Group (OMG) to provide a notation that is easily understandable by all business users: business analysts designing processes, developers implementing technology to perform those processes, and, business people managing and monitoring those processes. BPMN creates a standardized bridge for the gap between the business process design and process.

Standard BPMN 2.0 process definition models can be exchanged between graphical editors, and executed on any BPMN 2.0 compliant engine. Be aware that if you use technology specific features in your definition, you will not be able to use that workflow on a different technology. For example, if you define an Activiti workflow to work with Alfresco, you will not be able to run it on a TIBCO server.

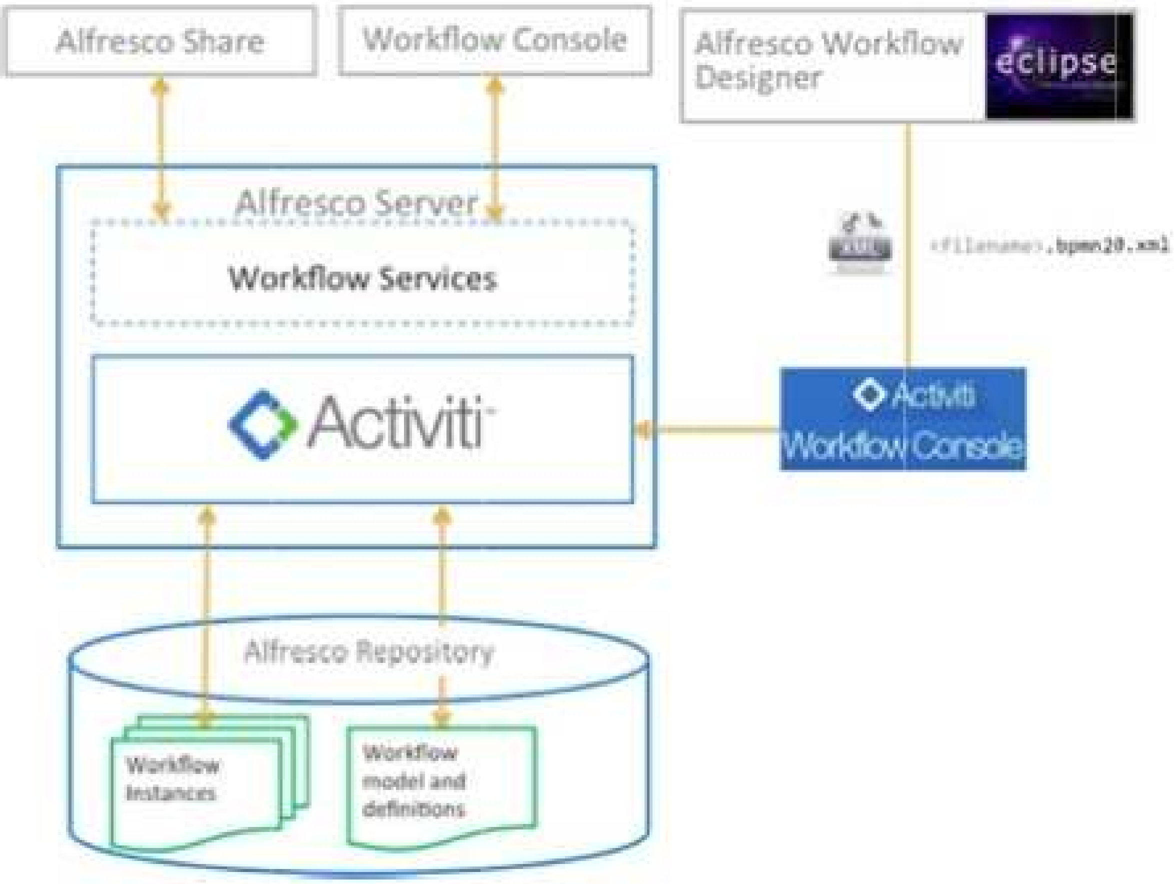
The following image shows part of a BPMN 2.0 process definition:



```
<process id="activitiInvitationModerated" name="Moderated activiti invitation process">

  <startEvent id="start" activiti:formKey="imwf:moderatedInvitationSubmitTask" />

  <sequenceFlow id="flow1" sourceRef="start" targetRef="reviewTask" />

  <userTask id="reviewTask" name="Review Task"
     activiti:formKey="imwf:activitiModeratedInvitationReviewTask">
     <extensionElements>
        <activiti:taskListener event="create"
           class="org.alfresco.repo.workflow.activiti.tasklistener.ScriptTaskListener">
           <activiti:field name="script">
              <activiti:string>
                 if (typeof bpm_workflowDueDate != 'undefined')
                    task.setVariable('bpm_dueDate', bpm_workflowDueDate);
                 if (typeof bpm_workflowPriority != 'undefined')
                    task.priority = bpm_workflowPriority;
                 </activiti:string>
           </activiti:field>
        </activiti:taskListener>
        <activiti:taskListener event="complete" class="org.alfresco.repo.workflow.activiti.tasklistener.
           <activiti:field name="script">
              <activiti:string>
                 execution.setVariable('imwf_reviewOutcome', task.getVariable('imwf_reviewOutcome'));
                 execution.setVariable('imwf_reviewer', person.properties.userName);
              </activiti:string>
           </activiti:field>
        </activiti:taskListener>
     </extensionElements>
     ....
```

Workflow Architecture

The following figure shows the high#level architecture for Alfresco workflow.

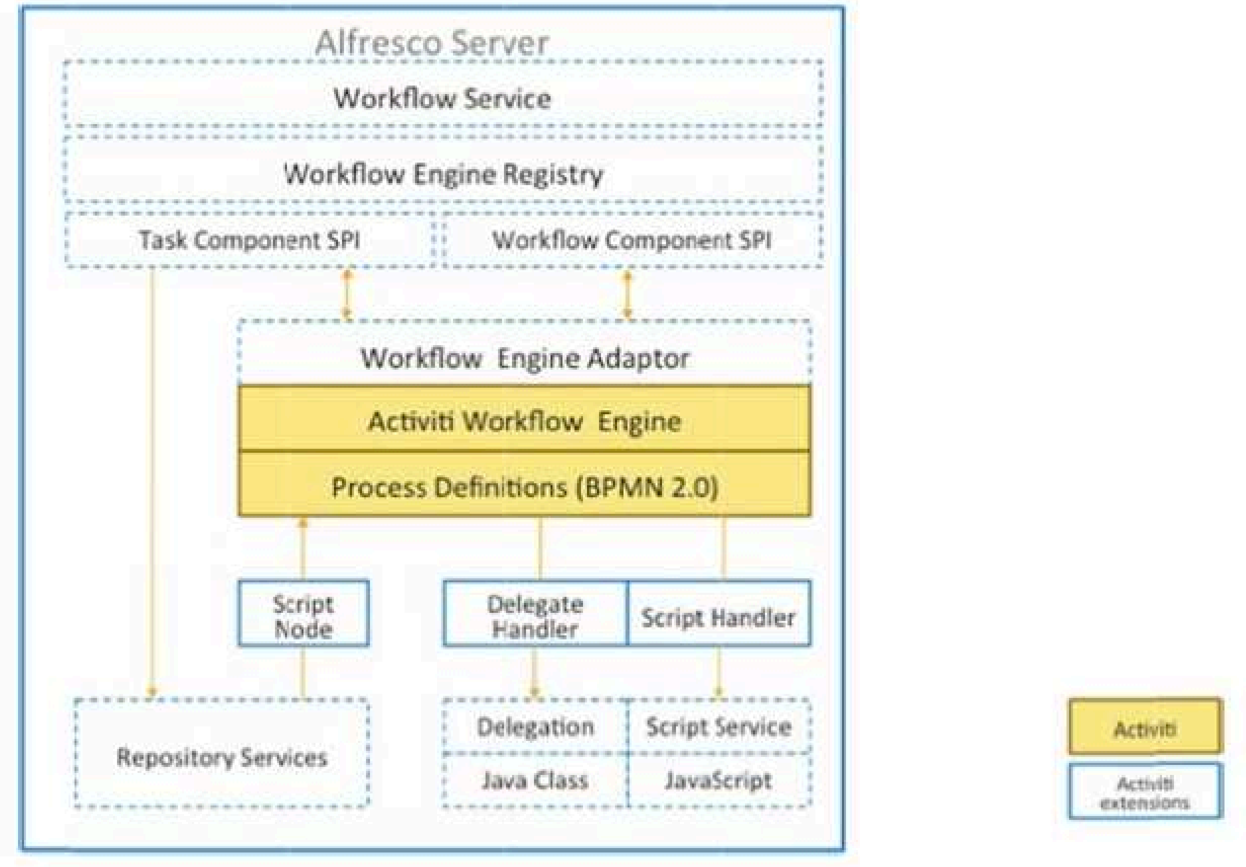


You can design workflow definitions using a graphical workflow designer that supports BPMN 2.0 or write the XML BPMN 2.0 process definition directly using an XML editor. Many workflow editors support BPMN 2.0 but may not understand some of the features of Alfresco workflow. We recommend the use of the Activiti eclipse designer plug#in for Eclipse that is Alfresco-aware.

You can deploy a workflow to Alfresco using the Activiti Workflow Console, or by using a Spring Bean.

Alfresco Activiti process definitions can include Alfresco JavaScript, and this in turn can access Alfresco content models in the repository so that you can provide your own specialized tasks for a workflow and access their properties. Process definitions have script node access which allows you to access objects in the Alfresco repository, such as documents and folders. Your workflow can access and modify document objects, for example marking documents as approved, or signed off.

Alfresco allows you to access your own Java Classes through the delegate handler, so you can integrate with other external systems. The following diagram show these features :-



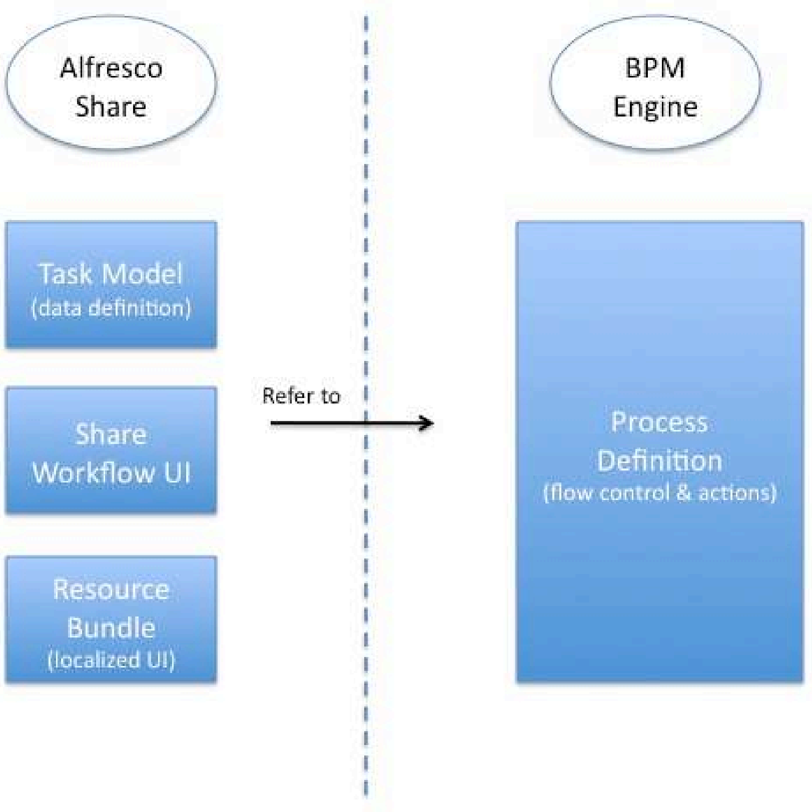
Workflow instances

Once a workflow instance has been started, it can not be changed. If you change the underlying process definition, it will be versioned. Any new workflow instance will reflect any changes to the workflow definition. Any old instances currently running will reference the old definition.

Workflow instances survive Alfresco server restarts, so all user tasks will still be running if you stop and restart the server. Process and task execution variables also survive Alfresco server restarts.

Workflow artifacts

The diagram below shows the artifacts and the relationship between them:-


Alfresco Share
BPM Engine
Task Model (data definition)
Share Workflow UI
Resource Bundle (localized UI)
Refer to
Process Definition (flow control & actions)

Process Definition

Activiti process definitions describe the events, activities (tasks) and gateways (choices) of a workflow. Tasks may be user tasks or script (system) tasks. User tasks are assigned to human performers (users). System tasks perform some kind of operation against the Alfresco repository. Both are described and implemented in the Process Definition.

Task Model

The Task Model provides a description for each of the user tasks in the workflow. Each task description consists of:

- Name and Title.
- Properties and Associations. For example, the information attached to the task.

The description is used to drive the user interface dialog for viewing and managing the Task. Alfresco provides a Data Dictionary for describing types of object to store, view and edit. This mechanism is also used to describe Workflow Tasks.

Share Workflow UI

You can customize the presentation of Tasks to the user in Alfresco Share. Customizing allows:

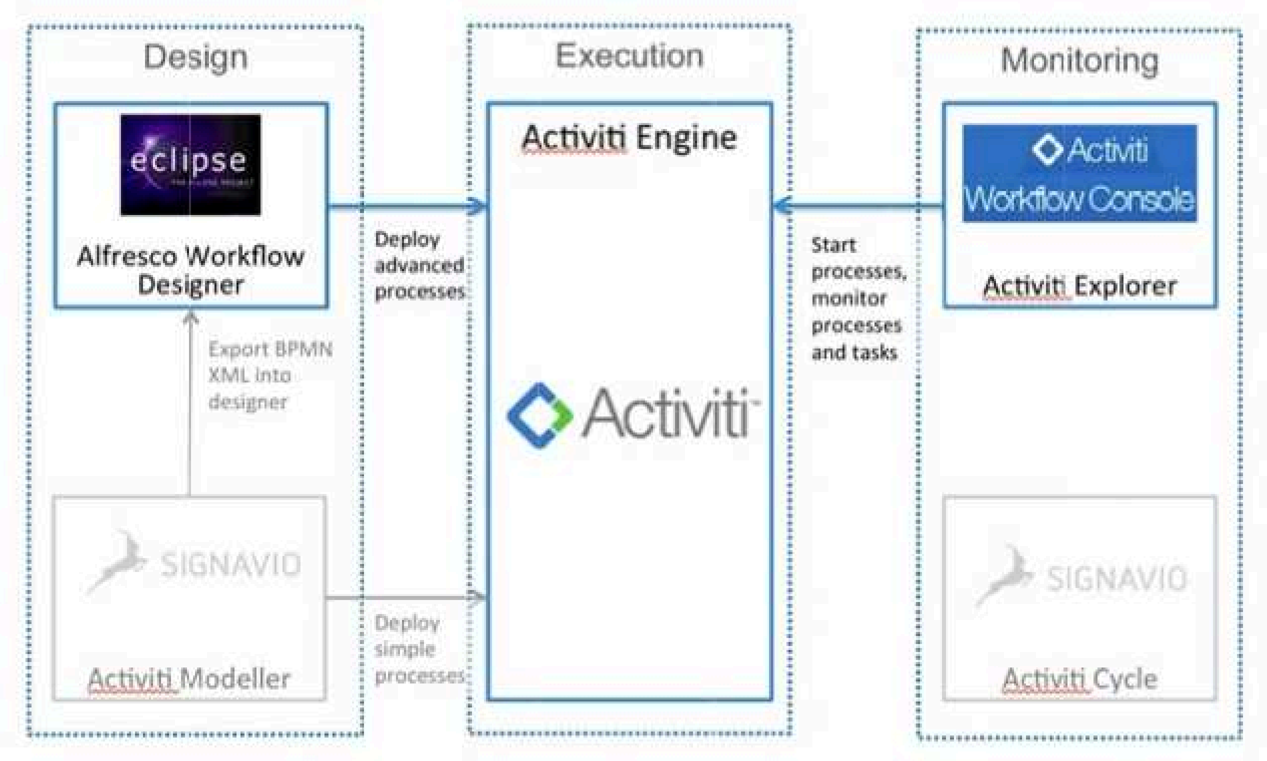
- Control over which Task properties are displayed
- Control over which Task properties are read-only and required
- Control over how each Task property is rendered in the forms

Resource Bundle (optional)

A workflow resource bundle provides all the human-readable messages displayed in the user interface for managing the workflow. Messages include Task titles, task property names, task choices etc. Alfresco supports full localization of Alfresco Share, including workflow. Therefore, the same Alfresco Share resource bundle configuration extends to workflow too.

Workflow tools

The following diagram shows the tools used in designing, executing, and monitoring an Alfresco workflow:


Design
eclipse
Alfresco Workflow Designer
Export BPMN XML into designer
SIGNAVIO
Activiti Modeller
Deploy advanced processes
Deploy simple processes
Execution
Activiti Engine
Activiti
Start processes, monitor processes and tasks
Monitoring
Activiti Workflow Console
Activiti Explorer
SIGNAVIO
Activiti Cycle

Activiti modeler

allows business and information analysts to model a BPMN 2.0 compliant business process in a web browser. This allows business processes to be shared, and no client software is needed before you can start modeling.

Activiti designer

is an Eclipse plugin, which enables a developer to enhance the model of the business process into a BPMN 2.0 process that can be executed on the Activiti process engine. You can also run unit tests, add Java logic, and create deployment artifacts with the Activiti Designer.

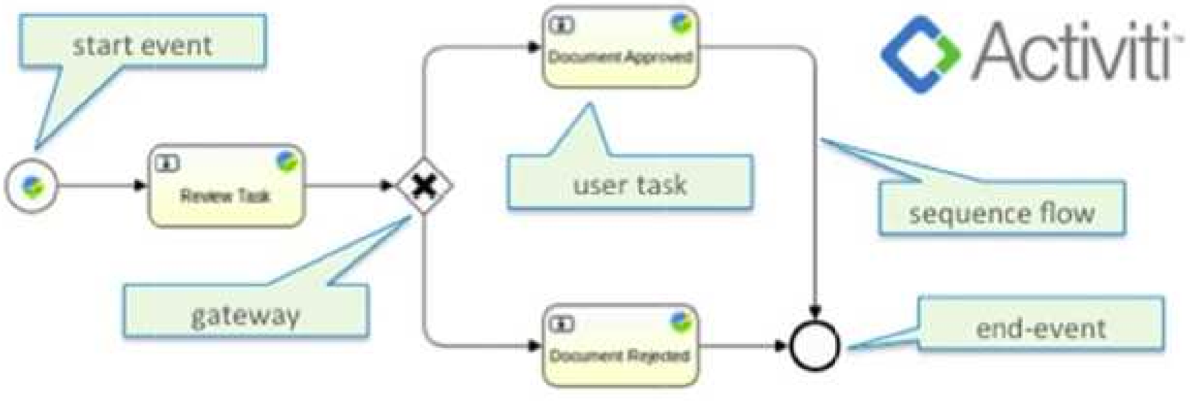
The Activiti workflow console

With the **Activiti Workflow Console** you can:

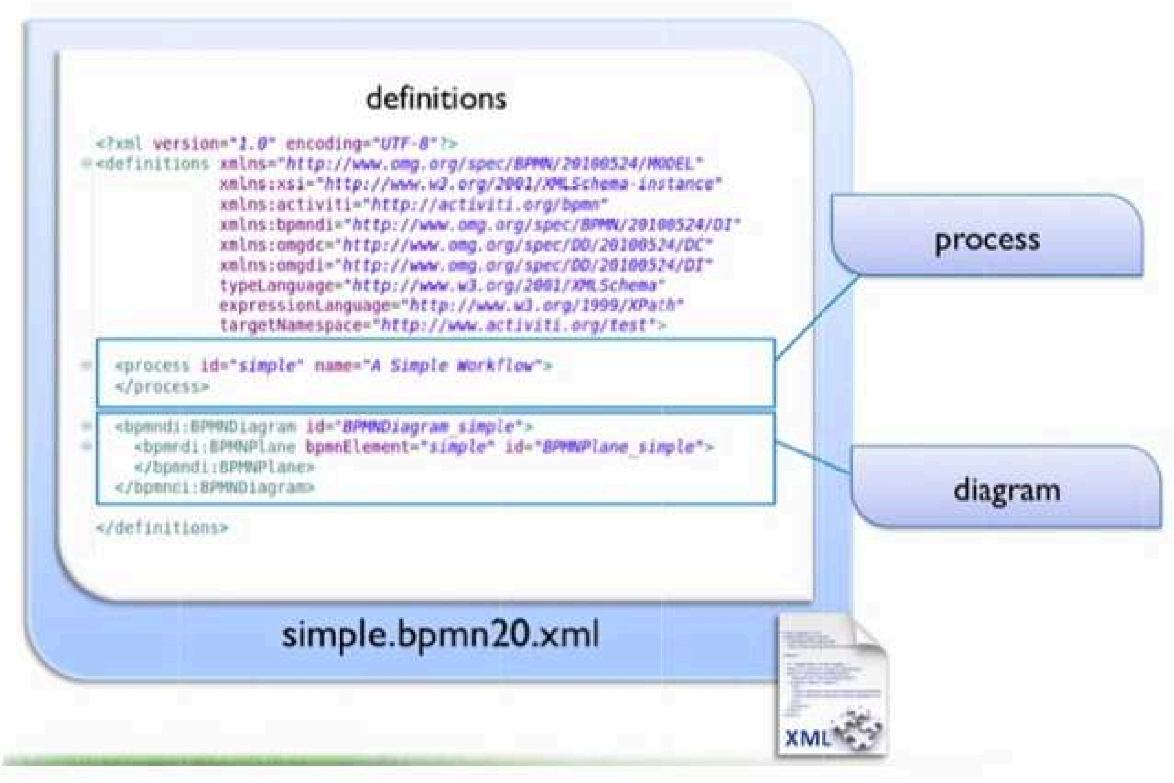
- View process definitions
- Manage deployments; deploy, view versions, and delete versions.
- Manage process instances
- View task variables
- Examine the process database

Process definitions

The following diagram shows a simple process definition and highlights the terminology used in BPMN 2.0.



The underlying definition is an xml file. The root element of the BPMN 2.0 schema is the `definitions` element, which can contain multiple process definitions. The following image show an empty process definition:



```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xmlns:activiti="http://activiti.org/bpmn"
             xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
             xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC"
             xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI"
             typeLanguage="http://www.w3.org/2001/XMLSchema"
             expressionLanguage="http://www.w3.org/1999/XPath"
             targetNamespace="http://www.activiti.org/test">

  <process id="simple" name="A Simple Workflow">
  </process>

  <bpmndi:BPMNDiagram id="BPMNDiagram_simple">
    <bpmndi:BPMNPlane bpmnElement="simple" id="BPMNPlane_simple">
    </bpmndi:BPMNPlane>
  </bpmndi:BPMNDiagram>

</definitions>
```

# A `definitions` element contains at least `xmlns` and `targetNamespace` declarations. The targetNamespace is an arbitrary string specified by you, and is useful for categorizing process definitions. The `process` element has two attributes:

id

this is required and maps to the key property of an Activiti ProcessDefinition object. The id is used to uniquely identify this process definition, for example when configuring the user interface, or in the Activiti workflow console.

name
this is optional and maps to the name property of a ProcessDefinition. The Activiti workflow engine itself does not use this property, but it is used in Alfresco Share for displaying the name in a user interface, so you should specify a name.

The `BPMNDiagram` element specifies the diagram interchange information for this process. The graphical design tool you use generates this information. This element will not appear when you are creating BPMN 2.0 process definition manually. The interchange information is used to re#create the diagram both in another graphical designer and in the run#time environment. Only one diagram is allowed per file, even though there may be more than one process definition.

Events

There are several types of events defined by BPMN 2.0, of which two always exist in a definition:

startEvent
Indicates where a process starts. A start event is triggered by the arrival of a message or similar trigger such as the expiration of a timer.

endEvent
An end event models the end process or subprocess. When process execution arrives in an end event, a result is thrown.

Events are described in detail in the Activiti user guide.

Sequence flows

After an element is visited during process execution, all outgoing sequence flows will be followed. So by default two outgoing sequence flows will create two separate, parallel paths of execution. This behavior can be modified. Sequence flows are described in detail in the Activiti user guide.

Tasks

userTask
describes work to be done by a human actor. When process execution arrives at a user task, a new task is created in the task list of the user or group assigned to that task.

scriptTask
describes an automatic activity. When a process execution arrives at the script task, the corresponding script is executed.

mailTask
is similar to a script task, but is specifically set up to send an email.

Tasks are described in detail in the Activiti user guide.

Gateways

A gateway is capable of consuming or generating tokens. It is graphically visualized as a diamond shape, with an icon inside. The icon describes the type of gateway. Gateways are described in detail in the Activiti user guide.

Parallel gateways

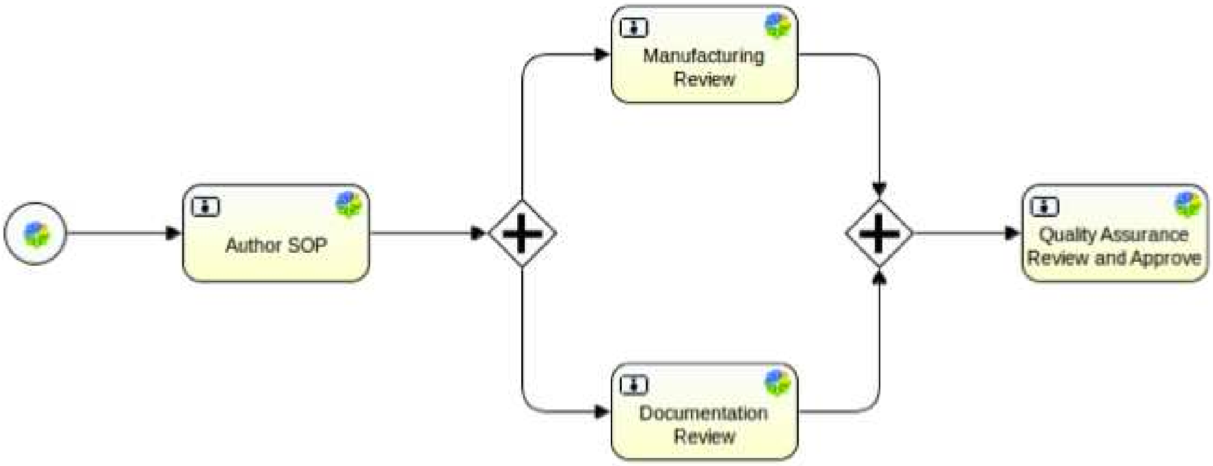
fork
all outgoing sequence flows are followed in parallel, creating one concurrent execution for each sequence flow.

join
all concurrent executions arriving at the parallel gateway wait at the gateway until execution has completed for each of the incoming sequence flows. The process then continues.

A parallel gateway can have both fork and join behavior, if there are multiple incoming and outgoing sequence flows for the same parallel gateway. In this case, the gateway will first join all the incoming sequence flows, before splitting into multiple concurrent paths of execution.

A parallel gateway does not evaluate conditions. If conditions are defined on the sequence flow connected with the parallel gateway, they are ignored.

The following diagram shows a definition with two parallel gateways.



The first gateway forks the flow of execution, generating two tokens for two review tasks. When these two tasks are completed, the second parallel gateway joins the two execution. Ssince there is only one outgoing sequence flow, no concurrent paths of execution will be created, and only the quality assurance task will be active.

Note that a parallel gateway does not need to be 'balanced'. You do not need to specify a matching number of incoming/outgoing sequence flows for corresponding parallel gateways.

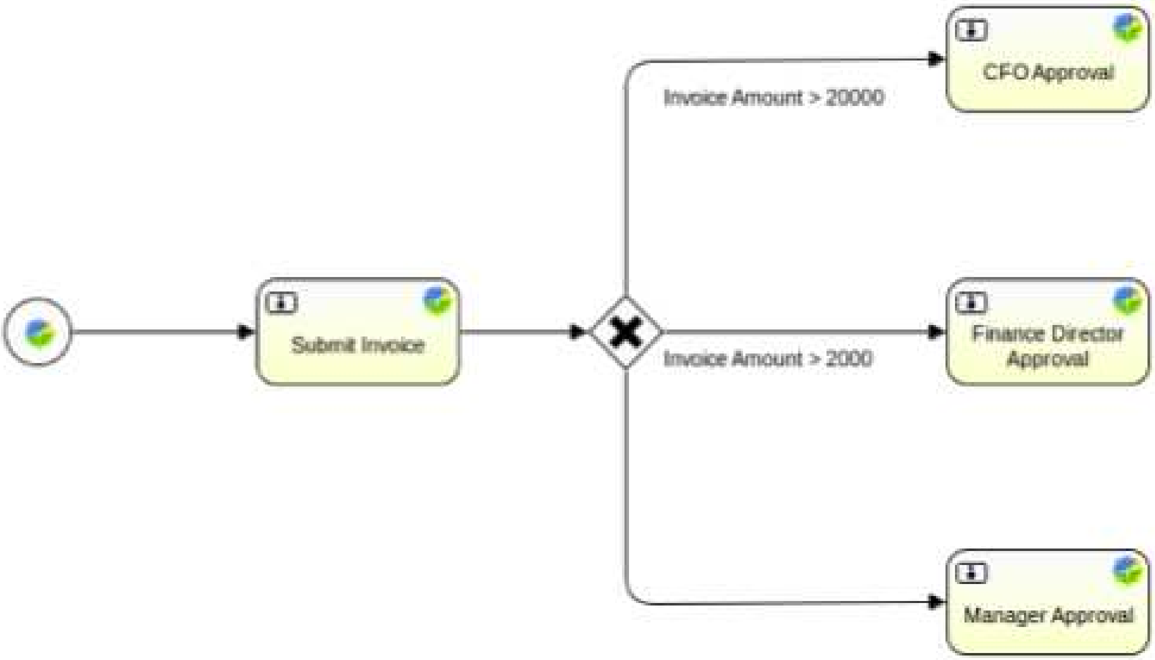
Exclusive gateways

When the execution of a workflow arrives at this gateway, all outgoing sequence flows are evaluated in the order in which they are defined. The sequence flow whose condition evaluates to true, is selected for propagating the token flow.

Note that the semantics of an outgoing sequence flow:

- In general in BPMN 2.0, all sequence flows whose conditions evaluate to true are selected to continue in a parallel way. When using an exclusive gateway, only one sequence flow is selected.
- When multiple sequence flows have conditions which evaluate to true, only the first one defined is selected to continue the process.
- If no sequence flow can be selected, an exception will be thrown. To ensure a sequence flow will always be selected, have no condition on one of your flows. No condition will always evaluate to true.

The following diagram shows an exclusive gateway that will choose one sequence flow based on the value of a property, in this example, the invoice amount. Only two flows have conditions on them going to CFO Approval and Finance Director Approval. The last sequence flow has no condition, and will be selected by default if the other conditional flows evaluate to false.



Variables

For example the Alfresco supplied BPM task model defines the property **bpm:assignee**. To reference this property in your process definition you would specify the string **bpm_assignee**. Note that the colon character is replaced by an underscore.

Variables in workflows exist at two levels; the process execution level and the task level. If you set the value of a variable in a task, the new value is not available at the process level. If you want to use a variable across tasks, or between a task and conditional flow, you need to copy the variable to the process execution level. Process level variables are available to tasks and sequence flows.

Node objects

The following variables are set by the start task in your process definition, and are accessible after the start task completes:

bpm_workflowDescription
The description for this in#flight workflow.

bpm_workflowDueDate
the due date for the workflow.

bpm_workflowPriority
The priority for the workflow.

bpm_package
A Repository Node with aspect **bpm:workflowPackage** representing the Workflow package containing content being routed through the workflow.

bpm_context
A Repository Node of type **cm:folder** representing the Alfresco folder in which the workflow was started.

The are some special node objects available in the process definition, that are not part of the task model:

initiator
A Repository Node of type **cm:person** representing the person who initiated the workflow.

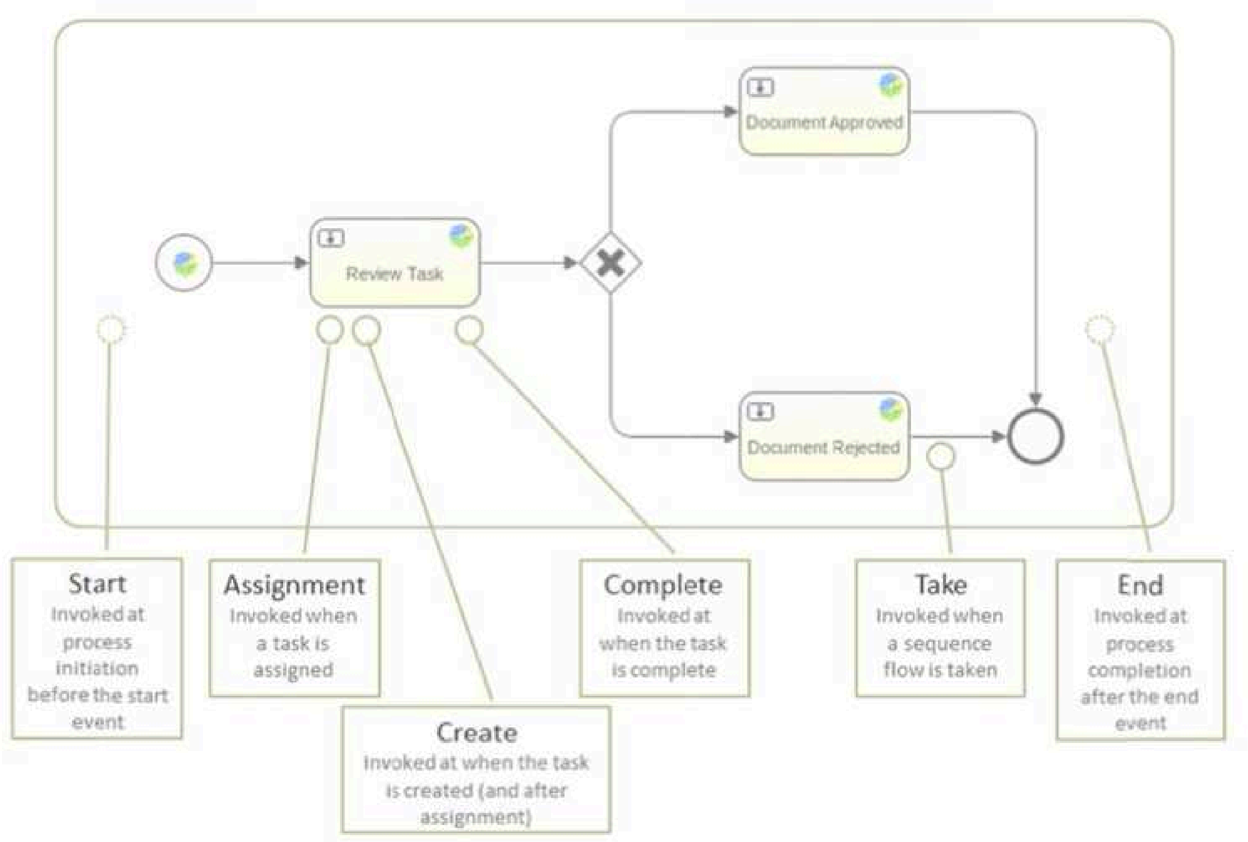
initiatorhome
A Repository Node of type **cm:space** representing the home folder of the person who initiated the workflow.

companyhome
A Repository Node of type **cm:space** representing the company home root folder.

Listeners

Execution listeners can be configured on the process itself, as well as activities and transitions. Task listeners can only be configured on user tasks.

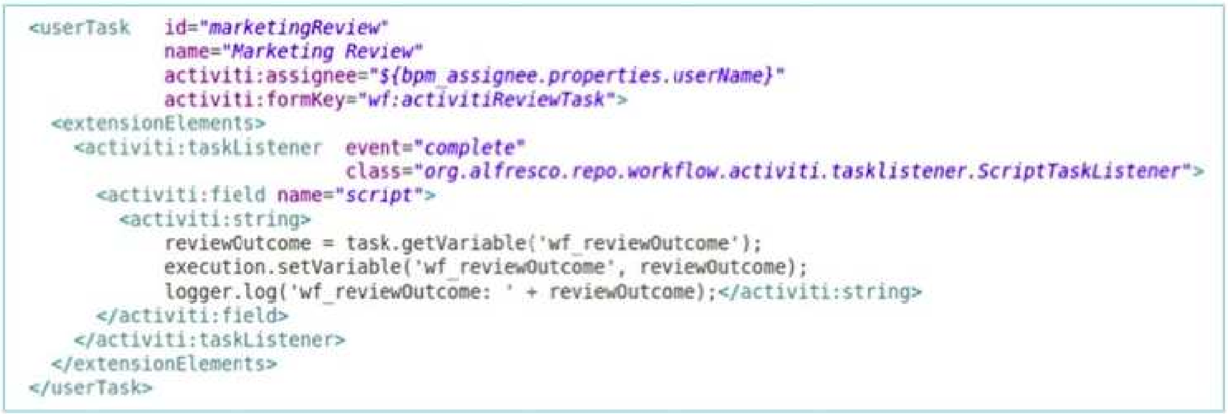
Listeners enable you to run your own code in the workflow. This can be Alfresco Javascript or a call to a Java class. The following diagram shows the events in a process definition where you can configure a listener.


Document Approved
Review Task
Document Rejected
Start
Invoked at process initiation before the start event
Assignment
Invoked when a task is assigned
Create
Invoked at when the task is created (and after assignment)
Complete
Invoked at when the task is complete
Take
Invoked when a sequence flow is taken
End
Invoked at process completion after the end event

Listeners are described in detail in the Activiti user guide.

Task listeners

The following diagram shows an XML fragment from a process definition that contains Alfresco-specific task listener.



```
<userTask   id="marketingReview"
            name="Marketing Review"
            activiti:assignee="${bpm_assignee.properties.userName}"
            activiti:formKey="wf:activitiReviewTask">
  <extensionElements>
    <activiti:taskListener  event="complete"
                            class="org.alfresco.repo.workflow.activiti.tasklistener.ScriptTaskListener">
      <activiti:field name="script">
        <activiti:string>
            reviewOutcome = task.getVariable('wf_reviewOutcome');
            execution.setVariable('wf_reviewOutcome', reviewOutcome);
            logger.log('wf_reviewOutcome: ' + reviewOutcome);</activiti:string>
      </activiti:field>
    </activiti:taskListener>
  </extensionElements>
</userTask>
```

Listeners are described in detail in the Activiti user guide.

Execution listeners

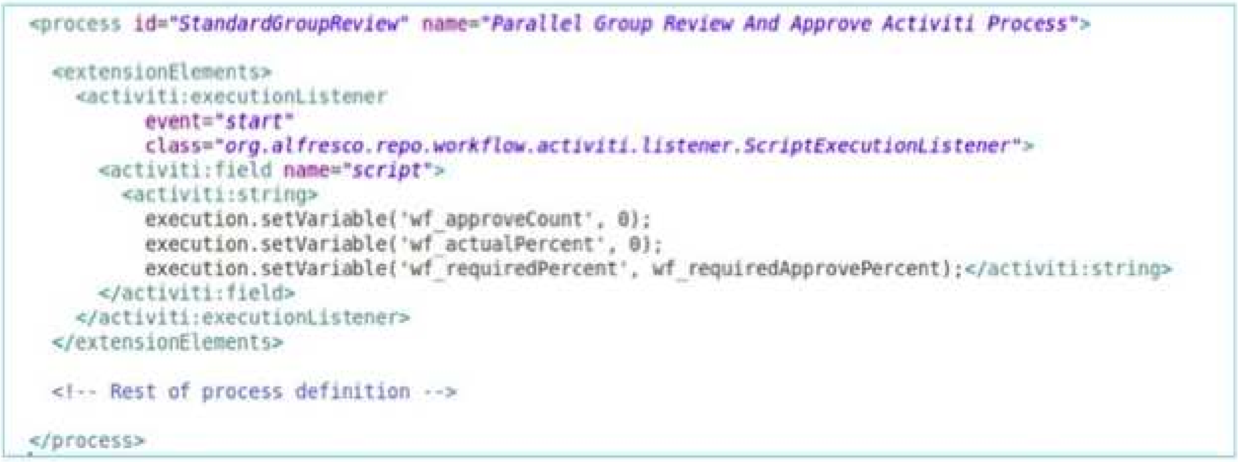
There are three events available:

start
invoked at the beginning of process execution, before the start event.

end
invoked at the end of the process execution, after the end event.

take
invoked when a sequence flow is invoked.

The code below shows an example of an execution listener to be invoked at the beginning of the process execution.

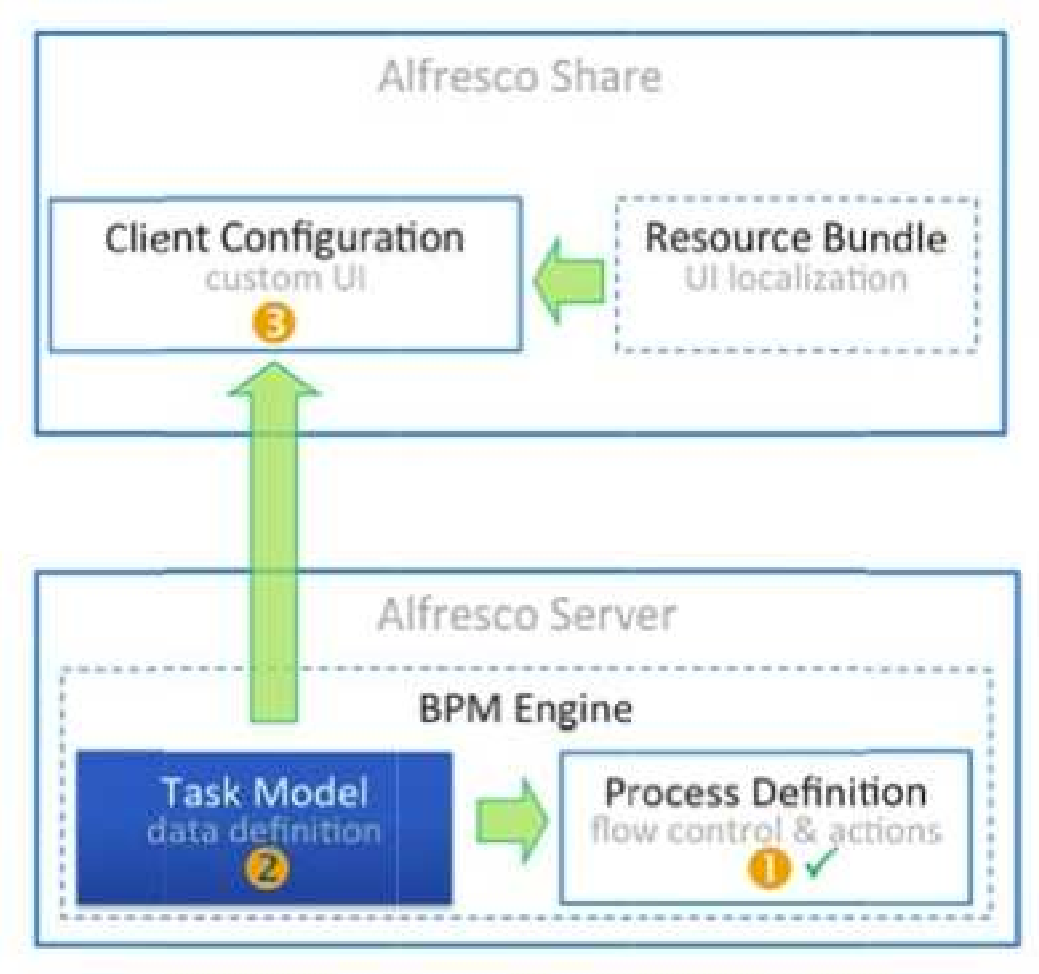


```
<process id="StandardGroupReview" name="Parallel Group Review And Approve Activiti Process">

  <extensionElements>
    <activiti:executionListener
          event="start"
          class="org.alfresco.repo.workflow.activiti.listener.ScriptExecutionListener">
      <activiti:field name="script">
        <activiti:string>
          execution.setVariable('wf_approveCount', 0);
          execution.setVariable('wf_actualPercent', 0);
          execution.setVariable('wf_requiredPercent', wf_requiredApprovePercent);</activiti:string>
      </activiti:field>
    </activiti:executionListener>
  </extensionElements>

  <!-- Rest of process definition -->

</process>
```

Listeners are described in detail in the Activiti user guide.

Task model

The client configuration allows for customization of the UI component that is used for presenting workflow#related information to the user and taking inputs from the user. Alfresco uses resource bundles to select the text that displays. Resource bundles allow language-specific strings to be used to display information about a workflow or task. The following diagram shows the relationship between the process definition and the task model on the server, and the client configurations and resource bundle in the client.

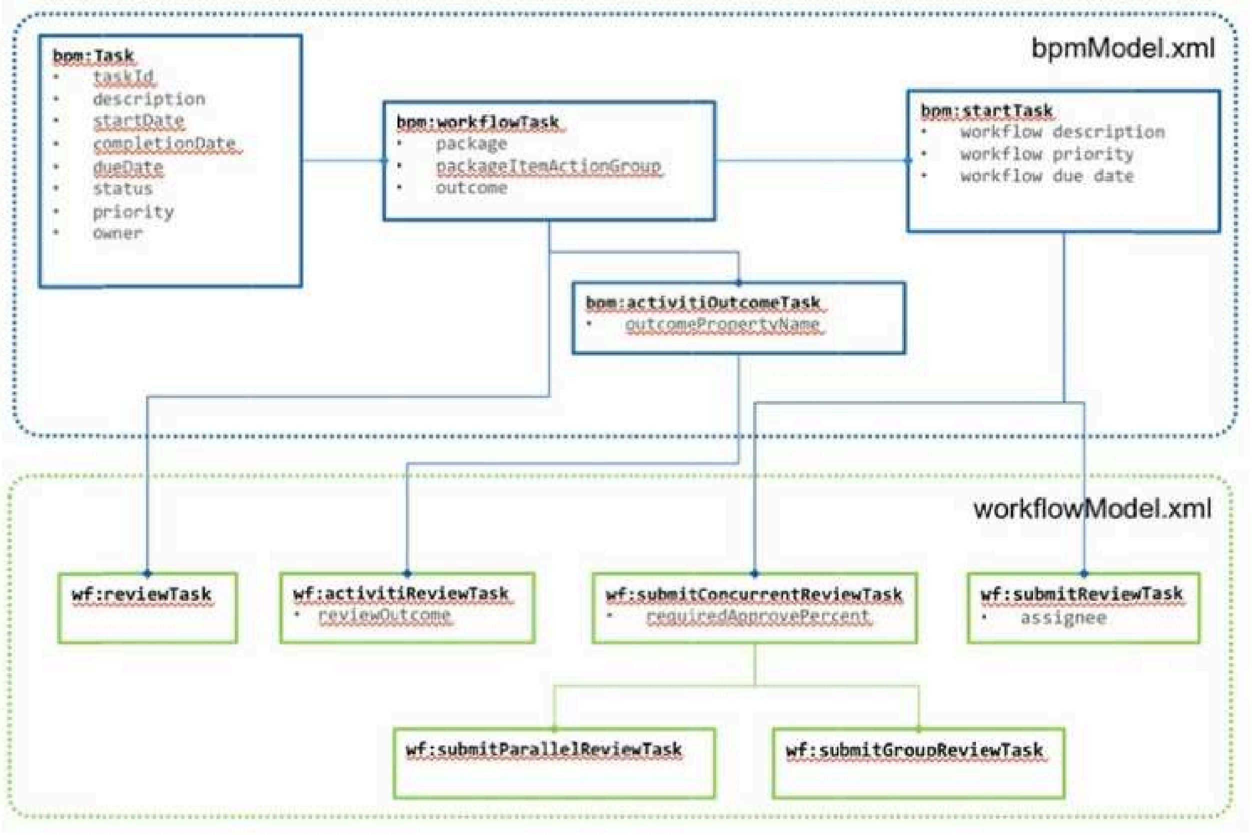


When creating workflows you will need to create the process definition using the graphical designer, create a task model to define your specific metadata items required on a task, and optionally look at customizing the user interface to support the custom task model that you have defined. Using a resource bundle is optional.

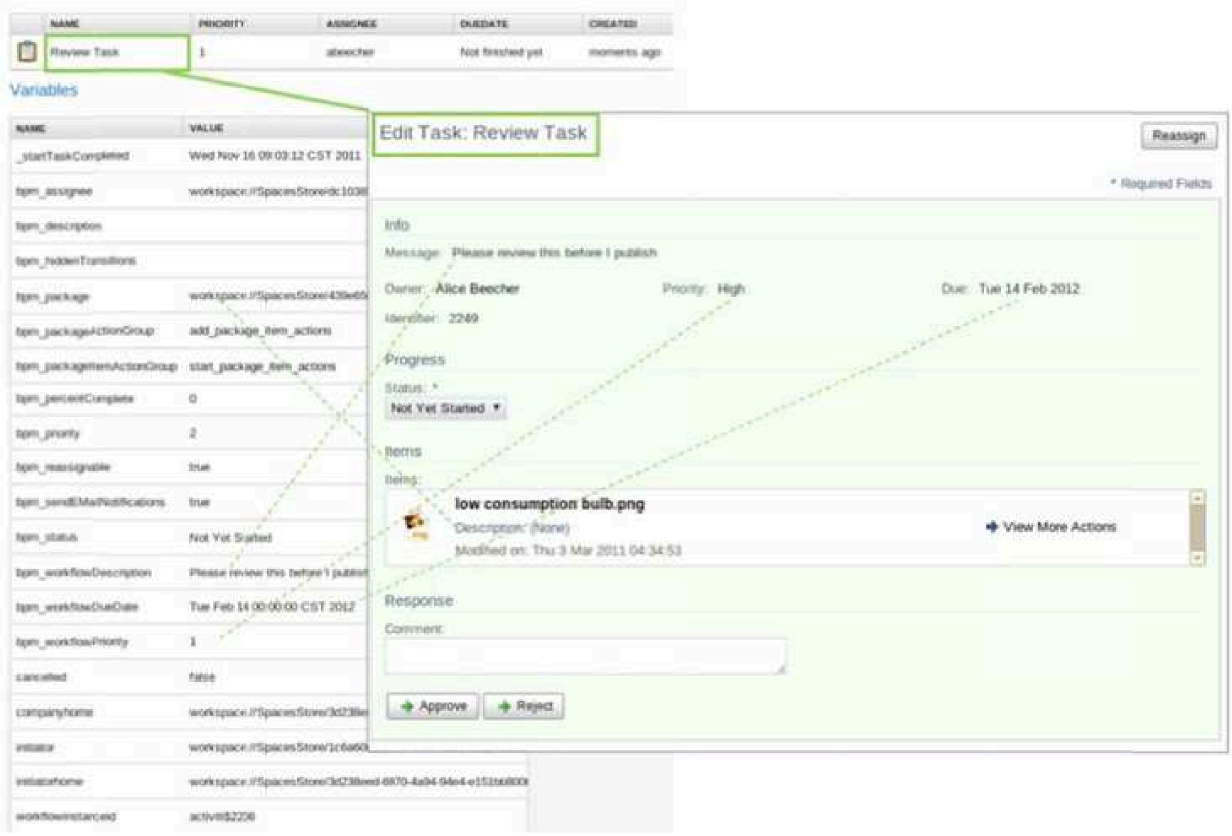
Alfresco ships with two default workflow models that support the default set of process definitions.

bpmModel.xml
the basic workflow content model

workflowModel.xml
contains more detailed tasks and specializes the basic tasks from the bpm model.



The task model is important when considering user interfaces, as the properties from task types are the only properties which can be shown to the user. The following diagram shows how a review task, which is of type **wf:activitiReviewTask** maps to the user interface. The property list in the background is taken from the Activiti workflow explorer.



	NAME	PRIORITY	ASSIGNEE	DUEDATE	CREATED
	Review Task	1	abeecher	Not finished yet	moments ago

Variables

NAME	VALUE
_startTaskCompleted	Wed Nov 16 09:03:12 CST 2011
bpm_assignee	workspace://SpacesStore/dc1038
bpm_description	
bpm_hiddenTransitions	
bpm_package	workspace://SpacesStore/438e65
bpm_packageActionGroup	add_package_item_actions
bpm_packageItemActionGroup	start_package_item_actions
bpm_percentComplete	0
bpm_priority	2
bpm_reassignable	true
bpm_sendEMailNotifications	true
bpm_status	Not Yet Started
bpm_workflowDescription	Please review this before I publish
bpm_workflowDueDate	Tue Feb 14 00:00:00 CST 2012
bpm_workflowPriority	1
cancelled	false
companyhome	workspace://SpacesStore/3d238e
initiator	workspace://SpacesStore/1c6a60
initiatorhome	workspace://SpacesStore/3d238eed-6870-4a94-94e4-e151bb800
workflowinstanceid	activiti$2208

Edit Task: Review Task

* Required Fields

Info

Message: Please review this before I publish

Owner: Alice Beecher Priority: High Due: Tue 14 Feb 2012

Identifier: 2249

Progress

Status: *
Not Yet Started ▾

Items

Items:

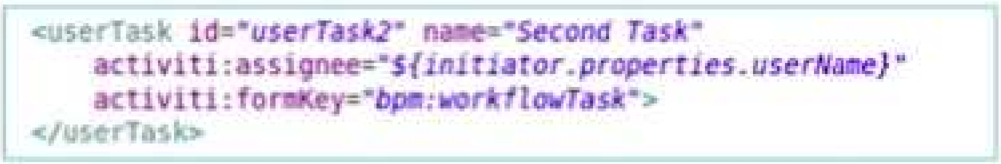
low consumption bulb.png
Description: (None)
Modified on: Thu 3 Mar 2011 04:34:53

Response

Comment:

Specifying the task type

You specify the task type using the **formKey** attribute on a userTask element. If you are developing your BPMN from scratch you can specify this in your XML. If you are using the Activiti designer you can specify it under the main configuration for a task.



```
<userTask id="userTask2" name="Second Task"
    activiti:assignee="${initiator.properties.userName}"
    activiti:formKey="bpm:workflowTask">
</userTask>
```

Setting up Activiti Designer

Installing Eclipse

1. Download the latest version of Eclipse for your platform from http://www.eclipse.org/downloads.
2. Follow the installation instructions on linked to on the download page.
3. To run Eclipse, follow the advice in the release notes, `readme_eclipse.html` .

You now have a running eclipse instance in which you can install the Activiti designer plugin.

Installing Activiti designer

Follow these steps to install the plugin.

1. In the eclipse menu bar, click **Help** > **Install New Software**

2. Click **Add**

   The **Add Repository** dialog is displayed
3. Start Eclipse.
4. Fill in the name field with `Activiti BPMN 2.0 designer`, and fill in the location field with `http://activiti.org/designer/update/`.
5. Click **OK**
6. Click **Finish**

   Eclipse will install the latest version of the Activiti designer eclipse plugin.

Deploying the task model

In the following example configuration we are deploying a process definition `adHocModel.bpmn2.0.xml`) and a workflow content model `adHocModel.bpmn2.0.xml`. In both properties, the “location” is the classpath location of the XML file.

```
   <bean id="myworkflows.workflowBootstrap" parent="workflowDeployer">
   <property name="models">
      <list>
         <-- Task Model associated with above process definition -->
         <value>alfresco/workflow/adhocModel.xml</value>
      </list>
   </property>
   <property name="workflowDefinitions">
      <props>
         <prop key="engineId">activiti</prop>
         <prop key="location">alfresco/extension/adHocModel.bpmn2.0.xml</prop>

         <prop key="mimetype">text/xml</prop>
         <prop key="redeploy">false</prop>
      </props>
   </property>
</bean>
```

Deploying a process definition

If you use manual deployment, the Alfresco server must be shut down. Process definitions will be deployed when Alfresco starts.

Configuring templated nodes

Templated nodes allows for the storing of content templates in Alfresco repositories that users can then use to create content.

Templated nodes provide a convenient way for users to quickly create content based on a pre-determined style, such as documents already formatted to company guidelines. Once a template has been stored in the Alfresco repository users can create new content items based upon it from the **Create Content** menu in the **Document Library**.

You can have an unlimited number of templated nodes.

Setting files as templates

In the Node Templates folder you can store documents that users can then use as document templates.

1. In the Alfresco **Repository** open the **Data Dictionary** then **Node Templates**.
2. Either drag and drop a content item that you want to use as a template onto the **Node Templates** drag and drop area, or click **Upload** and browse to and select the required file.

The standard Alfresco **Create Content** options are also available, so that you can create templates directly from Alfresco, in just the same way as a user would create new content.

If you already have templated nodes set up, you can select to create content **By Templated Node** and create additional templates based on your existing templates.

3. Click **OK** when the upload is complete.

The file is now available to your users as a template when they select to create content **By Templated Node**.

Managing content stores

This section gives an overview on the Content Store Selector and Caching Content Store, their properties, and configuration details with examples.

The Content Store Selector

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 as an 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>
   ...
```

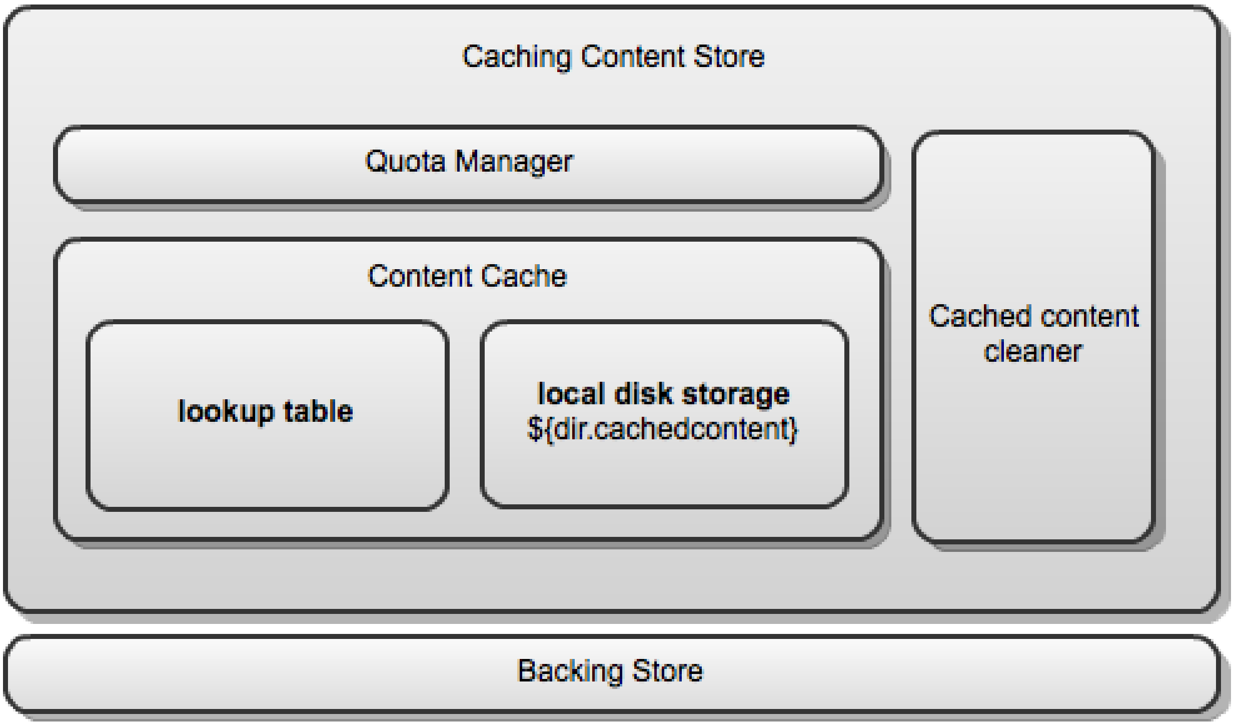
Caching Content Store (CCS)

This section provides an overview on Caching Content Store (CCS) and describes how to configure it.

`CachingContentStore` class overview

The `CachingContentStore` class adds transparent caching to any ContentStore implementation. Wrapping a slow ContentStore in a `CachingContentStore` improves access speed in many use cases. Example use cases include document storage using a XAM appliance or cloud-based storage, such as Amazon's S3.

The diagram below shows the architecture of CCS.



The major classes and interfaces that form the Caching Content Store are:

- **CachingContentStore:** This is the main class that implements the ContentStore interface, and can therefore, be used anywhere that a ContentStore could be used. The `CachingContentStore` handles all the high level logic of interaction between the cache and the backing store, while the caching itself is provided by a collaborating `ContentCache` object.
- **ContentCache:** This class is responsible for putting items into and getting items from the cache. The single supplied implementation (`ContentCacheImpl`) for this class uses a lookup table to keep track of the files that are being managed by the cache, and a directory on the local file system to store the cached content files. The lookup table itself is an `EhcacheAdapter` instance.
- **QuotaManagerStrategy:** The quota managers implement this interface and control how the disk usage is consumed for cached content storage. Alfresco provides two implementations for this: `UnlimitedQuotaStrategy` (does not restrict disk usage, thereby effectively disabling the quota function) and `StandardQuotaStrategy` (attempts to keep usage below the maximum specified in bytes or MB).

The `CachingContentStore` class is highly configurable and many of its components could be exchanged for other implementations. For example, the lookup table could be easily replaced with a different implementation from the Ehcache-based class supplied, or the `ContentCacheImpl` could be replaced with an implementation that uses some alternative to the local file system for storage of cached content.

The cached content cleaner (`CachedContentCleaner`) periodically traverses the directory structure containing the cached content files and deletes the content files that are not in use by the cache. Files are considered not in use by the cache if they have no entry in the lookup table managed by `ContentCacheImpl`. The content cache cleaner is not a part of the architecture but is a helper object for `ContentCacheImpl` and allows it to operate more efficiently.

`CachingContentStore` properties

This topic describes the properties that you can configure for the `CachingContentStore` class.

The following properties are used in the sample context file, `caching-content-store-context.xml.sample` and can be set in the `alfresco-global.properties` file. Their default values are provided in the `repository.properties` file.

system.content.caching.cacheOnInbound=true

Enables write-through caching. If true, an attempt to write the content to the backing store results in the item being cached. Therefore, the first time an item is read (provided the item has not been removed from the cache in the mean time), the file is already cached locally for faster access times. It is recommended that this property is set to `true` for most usage scenarios.

system.content.caching.maxDeleteWatchCount=1

Defines the number of times the file must have been observed as being available for deletion by previous cleanup runs before it is actually deleted. The default value is always set to 1, but can be increased if readers obtained from the cache could not be used due to the underlying file being deleted.

system.content.caching.contentCleanup.cronExpression=0 0 3 * * ?

Specifies how often the cached content cleanup job will run. The supplied value is a quartz expression and is similar to a Unix cron expression. In this case, the cleaner will run at 3 am every morning.

system.content.caching.timeToLiveSeconds=0

Specifies the maximum time in seconds that an item can exist in the cache. After this time elapses, the item will no longer be cached and a request for the content URL will result in the item being fetched from the backing store and cached afresh. A value of 0 means that items will not have a TTL parameter applied to them.

system.content.caching.timeToIdleSeconds=60

Specifies the maximum time an item in the cache can exist without being requested. Each time the item is accessed, the Time To Idle parameter is refreshed and the item will remain in the cache.

system.content.caching.maxElementsInMemory=5000

Applies to the lookup table in the ContentCache. Each content URL requires two entries in the lookup table, so a value of 5000 can allow 2500 content items to be held in memory for the lookup table.

system.content.caching.maxElementsOnDisk=10000

Applies to the lookup table in the ContentCache. Each content URL requires two entries in the lookup table, so a value of 10000 can allow 5000 items to be held on disk.

system.content.caching.minFileAgeInMillis=2000

Specifies that files must be at least this age before they are marked for deletion. This also stops unnecessary checks, such as loading and examining the associated properties file.

system.content.caching.maxUsageMB=4096

Specifies the maximum disk usage in MB that cached content should consume. In other words, this property defines the disk space quota allocated to the `${dir.cachedcontent}` directory. It is used by the `StandardQuotaStrategy` class as configured in the `caching-content-store-context.xml.sample` file.

system.content.caching.maxFileSizeMB=0

Specifies the maximum size in MB of any individual file of cached content. Content larger than this size can still be retrieved using the `CachingContentStore` class but the content will not be cached. If this property is set to zero, then no size limit will apply to the individual files. This property is used by the `StandardQuotaStrategy` class as configured in the `caching-content-store-context.xml.sample` file.

Configuring `CachingContentStore`

This topic describes how to configure the `CachingContentStore` class.

To demonstrate step-by-step configuration of the `CachingContentStore` class, the spring context file, `caching-content-store-context.xml.sample` is used as a starting point for adding caching to a content store. Once configured, you can activate the sample file by removing the `.sample` file extension and placing it in your Alfresco installation `extension` directory at `<ALFRESCO_HOME>/tomcat/shared/classes/alfresco/extension`.

1. Define an instance of the `CachingContentStore` class. This is the top level bean that ties together the CCS as a whole.

   ```
   <bean id="fileContentStore"
    class="org.alfresco.repo.content.caching.CachingContentStore" init-
   method="init">
     <property name="backingStore" ref="backingStore"/>
     <property name="cache" ref="contentCache"/>
     <property name="cacheOnInbound"
    value="${system.content.caching.cacheOnInbound}"/>
     <property name="quota" ref="standardQuotaManager"/>
   </bean>
   ```

   In this case, the `fileContentStore` bean is overridden. The `ContentService` bean uses `fileContentStore` bean, so CCS is used automatically. You can also specify a different name and an overridden `contentService` bean. The main collaborators of backingStore, cache and quota refer to the beans for Backing Store, Content Cache and Quota Manager as shown in the diagram in the CachingContentStore overview topic. Each CachingContentStore class should have its own dedicated instances of these collaborators and they should not be shared across other CachingContentStore beans, should you have any defined.

2. Define a backing store. This CCS uses this ContentStore to provide caching for TenantRoutingS3ContentStore.

```
<bean id="tenantRoutingContentStore"

 class="org.alfresco.module.org_alfresco_module_cloud.repo.content.s3store.TenantR
         parent="baseTenantRoutingContentStore">

  <property name="defaultRootDir" value="${dir.contentstore}" />
  <property name="s3AccessKey" value="${s3.accessKey}" />
  <property name="s3SecretKey" value="${s3.secretKey}" />
  <property name="s3BucketName" value="${s3.bucketName}" />
  <property name="s3BucketLocation" value="${s3.bucketLocation}" />
  <property name="s3FlatRoot" value="${s3.flatRoot}" />
  <property name="globalProperties">
    <ref bean="global-properties" />
  </property>

</bean>
```

Remember to change this bean's ID to backingStore for use with the preceding XML snippet, or change the ref attribute in the fileContentStore bean definition to refer to the correct ID (tenantRoutingContentStore).

3. Define a ContentCache. This object is responsible for placing content into (and retrieving content from) the cache.

```
<bean id="contentCache"
 class="org.alfresco.repo.content.caching.ContentCacheImpl">
  <property name="memoryStore" ref="cachingContentStoreCache"/>
  <property name="cacheRoot" value="${dir.cachedcontent}"/>
</bean>
```

The ContentCacheImpl uses a fast lookup table provided by Ehcache for determining whether an item is currently cached by the CCS, for controlling the maximum number of items in the cache and their Time To Live (TTL). The lookup table is specified here by the memoryStore property. The ContentCacheImpl also uses a directory on the local filesystem for storing binary content data (the actual content being cached). This directory is specified by the cacheRoot property. The following code illustrates the bean referencing the memoryStore reference above:

```
<bean id="cachingContentStoreCache"
 class="org.alfresco.repo.cache.EhCacheAdapter">
       <property name="cache">
           <bean
 class="org.springframework.cache.ehcache.EhCacheFactoryBean">
               <property name="cacheManager">
                   <ref bean="internalEHCacheManager" />
               </property>
               <property name="cacheName">
                   <value>org.alfresco.cache.cachingContentStoreCache</
value>
               </property>
               <property name="eternal" value="false"/>
               <property name="timeToLive"
 value="${system.content.caching.timeToLiveSeconds}"/>
               <property name="timeToIdle"
 value="${system.content.caching.timeToIdleSeconds}"/>
               <property name="maxElementsInMemory"
 value="${system.content.caching.maxElementsInMemory}"/>
```

```
               <property name="maxElementsOnDisk"
 value="${system.content.caching.maxElementsOnDisk}"/>
               <property name="overflowToDisk" value="true"/>
               <property name="diskPersistent" value="true"/>
           </bean>
       </property>
   </bean>
```

4. Now that you have configured the key components of the `CachingContentStore` class, backing store (ContentStore) and ContentCache, you can optionally specify a quota manager. If you do not wish to specify the quota manager, then the `UnlimitedQuotaStrategy` will be used. The example CCS bean above expects this bean to be defined:

```
<bean id="standardQuotaManager"

 class="org.alfresco.repo.content.caching.quota.StandardQuotaStrategy"
       init-method="init"
       destroy-method="shutdown">
     <property name="maxUsageMB" value="4096"/>
     <property name="maxFileSizeMB" value="0"/>
     <property name="cache" ref="contentCache"/>
     <property name="cleaner" ref="cachedContentCleaner"/>
</bean>
```

5. Finally, to ensure that the disk space is used in a controlled manner, a `CachedContentCleaner` should be configured to clean up cached content files that are no longer being used by the cache.

```
bean id="cachingContentStoreCleanerJobDetail"
 class="org.springframework.scheduling.quartz.JobDetailBean">
       <property name="jobClass">

 <value>org.alfresco.repo.content.caching.cleanup.CachedContentCleanupJob</
value>
       </property>
       <property name="jobDataAsMap">
           <map>
               <entry key="cachedContentCleaner">
                   <ref bean="cachedContentCleaner" />
               </entry>
           </map>
       </property>
   </bean>

   <bean id="cachedContentCleaner"

 class="org.alfresco.repo.content.caching.cleanup.CachedContentCleaner"
       init-method="init">
       <property name="minFileAgeMillis"
 value="${system.content.caching.minFileAgeMillis}"/>
       <property name="maxDeleteWatchCount"
 value="${system.content.caching.maxDeleteWatchCount}"/>
       <property name="cache" ref="contentCache"/>
       <property name="usageTracker" ref="standardQuotaManager"/>
   </bean>

   <bean id="cachingContentStoreCleanerTrigger"
 class="org.alfresco.util.CronTriggerBean">
       <property name="jobDetail">
           <ref bean="cachingContentStoreCleanerJobDetail" />
       </property>
       <property name="scheduler">
           <ref bean="schedulerFactory" />
       </property>
       <property name="cronExpression">
           <value>
${system.content.caching.contentCleanup.cronExpression}</value>
```

```
        </property>
    </bean>
```

Note that both the cleaner and the quota manager limit the usage of disk space but hey do not perform the same function. In addition to removing the orphaned content, the cleaner's job is to remove files that are out of use from the cache due to parameters, such as TTL, which sets the maximum time an item should be used by the CCS. The quota manager exists to set specific requirements in terms of allowed disk space.

A number of property placeholders are used in the above definitions. You can replace them directly in your configuration with the required values, or you can use the placeholders as they are and set the values in the `repository.properties` file. An advantage of using the property placeholders is that the sample file can be used with very few changes and the appropriate properties can be modified to get the CCS running with little effort.

Migrating

Migrating servers

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

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

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.

Monitoring Alfresco

This section describes the various methods for monitoring Alfresco.

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.

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, 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

When using Tomcat and a Sun JVM for a richest 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, hot spot 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 without manual configuration of connection details.

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, 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.

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.
`ehCacheTracerJob`	Collects detailed cache usage statistics and outputs them to the console, depending on how logging has been configured for the server. By default, this job is not activated. To activate this job, uncomment the `scheduler` property.

Setting up Alfresco multi-tenancy

Alfresco also supports multi-tenancy (MT) features that enable Alfresco to be configured as a true single-instance, multi-tenant environment. Multi-tenancy allows 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

In Alfresco, the multi-tenancy feature is pre-configured out-of-the-box, although it is not enabled by default.

When you install Alfresco, multi-tenancy is disabled. The multi-tenancy feature is automatically enabled when the first tenant is created.

Only an Administrator user can create tenants.

However, if you wish to disable multi-tenancy, you need to delete all the tenants. Refer to Managing tenants.

Managing tenants

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

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

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

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

This topic describes the features and components that are not supported in a multi-tenant production environment.

Using multi-tenancy, you can configure multiple, independent tenants on a single Alfresco instance. However, multi-tenancy is not supported in the following functionalities:

- Record Management module
- CIFS
- WCM
- Portlets
- LDAP, NTLM and authentication methods other than `alfresco`
- Inbound email
- Content replication
- IMAP
- SharePoint Protocol
- Alfresco Share (for versions prior to Alfresco 3.2)

Setting up replication jobs

You manage replication jobs using the Admin Console tool in Alfresco Share, which provides the user interface for the replication service.

The replication service is responsible for persisting replication jobs that specify what is to be replicated, to where, and when. In addition, it monitors the status of currently executing replication jobs and allows you to cancel replications. The replication service finds the nodes that need to be transferred, and then it delegates to the transfer service.

Before you can use the replication jobs feature, you must set up transfer target definitions to specify where the transfer should be sent.

Creating a new transfer target for replication jobs

The files that control and monitor the transfer service are stored using a folder in Alfresco Share, in the **Transfers** space under **Data Dictionary**. The **Transfer Target Groups** space contains the transfer target definitions that specify the destination of the transfer. The group level below this space, called **Default Group**, is used to classify different sets of transfer targets.

1. On the source Alfresco server, browse to **Company Home > Data Dictionary > Transfers > Transfer Target Groups > Default Group**.
2. Create a folder for the new transfer target.

   For example, create a folder called **Transfer1**.
3. Edit the properties for the new folder and modify the following fields:

Field	What is it?
Endpoint Host	Type the host name of the target server.

Field	What is it?
Endpoint Port	Type the port number of the target server. For example, type `8080`.
User Name	Type a user name of an account on the target server.
Password	Type a password for the user account.
Enabled	Click this checkbox to enable the transfer target.

A rule defined on the **Default Group** folder specializes the type of any folder created within it. The type of the folder is set to `trx:transferTarget`, which you can then complete through the user interface.

4. Click **Save**.
5. On the source server, configure Share to open locked content.

   Refer to Configuring Share to open locked content in the source repository.
6. Verify that the target server is set up correctly.
   a. Log in to the source server.
   b. Create a folder on the source server.

      For example, in Company Home, create **Folder1**.
   c. Create a job with the source as **Folder1** and a transfer target as **Transfer1**. Refer to Creating a new replication job.
   d. Run the new job.
   e. On the target server, open the folder that contains the source for the job (**Folder1**).
   f. On the target server, click **View in Source Repository**.

      Folder1 opens in the source repository.

Configuring Share to open locked content in the source repository

Configure Share by mapping the remote repository identifier (`repositoryId`) and the remote Share URL. This then gives access to the remote repository.

1. Locate the `repositoryId` by browsing to the remote server's CMIS landing page using the following URL:

   ```
   http://{server}:{port}/alfresco/service/cmis/index.html
   ```

   The `repositoryId` field is displayed in the **CMIS Repository Information** panel.
2. Open the `<web-extension>\share-config-custom.xml.sample` file.
3. Locate the following example configuration:

   ```
    <config evaluator="string-compare" condition="Replication">
         <share-urls>
            <!--
               To discover a Repository Id, browse to the remote server's
    CMIS landing page at:
                 http://{server}:{port}/alfresco/service/cmis/index.html
               The Repository Id field is found under the "CMIS Repository
    Information" expandable panel.

               Example config entry:
                 <share-url repositoryId="622f9533-2a1e-48fe-af4e-
   ee9e41667ea4">http://new-york-office:8080/share/</share-url>
            -->
         </share-urls>
   ```

```
   </config>
```

4. Modify the `share-url repositoryId` tag with the remote server's `repositoryId` value, plus the remote server's URL.

   For example:

   ```
   <share-url repositoryId="622f9533-2a1e-48fe-af4e-ee9e41667ea4">http://
   new-york-office:8080/share/</share-url>
   ```
5. Save the sample file without the `.sample` extension, or alternatively, you can copy this configuration setting directly to your `share-config-custom.xml` file.
6. Restart the Alfresco server.

Configuring the File System Transfer Receiver

The Transfer Service is accessible as a bean named `TransferService`, and it can be defined, along with other related beans, in the `transfer-service-context.xml` spring context file.

A file system transfer target is marked by specializing a normal transfer target to the type `trx:fileTransferTarget`. It allows you to specify which folder node corresponds to the root folder of the file system receiver by associating the transfer target with a folder (the `trx:fileTransferRootFolder` association).

It supports sync mode transfer, so it can also be used by the replication service. It includes an embedded Derby database to keep track of data (NodeRef to file path mappings, for example), and it runs as a web application in an embedded Tomcat 7 instance using the Web Script Framework and MyBatis.

Setting up the File System Transfer Receiver

The File System Transfer Receiver is delivered as a compressed zip file.

1. Download the following file from the Alfresco download area:

   `alfresco-enterprise-file-transfer-receiver-4.1.5.zip`
2. Extract the ZIP file into a relevant directory.

   The File System Transfer Receiver ZIP file extracts into the following directory structure:

   ```
   classes
   lib
   webapps
   file-transfer-receiver.jar
   ```

   The following files are contained within the subdirectories.

   `/classes`

   ```
   ftr-custom-context.xml
   ftr-custom.properties
   ftr-launcher-context.xml
   ftr-launcher.properties
   log4j.properties
   ```

   `/lib`

   ```
   various library files
   ```

   `/webapps`

   ```
   file-transfer-receiver.war
   ```

Start File System Transfer Receiver

This section describes how to start the File System Transfer Receiver.

1. Ensure that you have expanded the File System Transfer Receiver ZIP file:

   `alfresco-enterprise-file-transfer-receiver-4.1.5.zip`
2. To run the File System Transfer Receiver, enter the following command:

   ```
   java –jar file-transfer-receiver.jar
   ```

   You see information messages to indicate that the Tomcat web application server is starting.

File System Transfer Receiver launcher properties

This section describes the properties that are available in the `ftr-launcher.properties` file.

Property	Description
`ftr.tomcat.baseDir=`	Specifies the base directory in which the embedded Tomcat web application server is installed. This can either be an absolute path or a path relative to where the server is being started from. The default value of `${user.dir}` means that the Tomcat base directory is taken to be the user's current working directory.
`ftr.tomcat.portNum=`	Specifies the port number on which the FSTR Tomcat web application server is to listen. The default is 9090.

File System Transfer Receiver custom properties

This section describes the properties that are available in the `ftr-custom.properties` file.

Property	Description
`fileTransferReceiver.stagingDirectory=`	The staging directory is where the FSTR will temporarily store the files that it receives from the source repository during a transfer. These files include the manifest file that describes the metadata of the nodes being transferred as well as the actual content files associated with those nodes. All of these files are staged in the directory referenced by this property prior to being moved to their correct location below the root directory. The default is `./ftr-staging`
`fileTransferReceiver.rootDirectory=`	Specifies the location of the directory on the local file system that is the top level of the transferred tree of nodes. A node that is a child of the nominated root node of the transfer in the source repository will be placed in the directory referenced by this property when it's transferred. The default it `./ftr-root`
`fileTransferReceiver.jdbcUrl=jdbc:derby:./derbyDB;create=true;user=alfresco;password=alfresco`	The FSTR contains an embedded Apache Derby database that it uses to keep track of which nodes it receives and which file on the file system corresponds to which node. This property specifies the connection URL for this embedded database. It is unlikely that it will need to be changed. Alfresco recommends that you do not store FSTR database on a network file system location, such as an NFS volume. The database must be on a local disk to ensure data integrity.

Property	Description
`fileTransferReceiver.username=`	The user name that the source repository will have to declare when initiating a transfer to this FSTR. This property must correspond with the user name property stored on the transfer target in the source repository. The default is set to `admin`.
`fileTransferReceiver.password=`	The password that the source repository will have to declare when initiating a transfer to this FSTR. This property must correspond with the password property stored on the transfer target in the source repository. The default is set to `admin`.

File System Transfer Receiver log file properties

You can debug the File System Transfer Receiver issues using log4jproperties. This section describes the log4j properties that you can set.

For example:

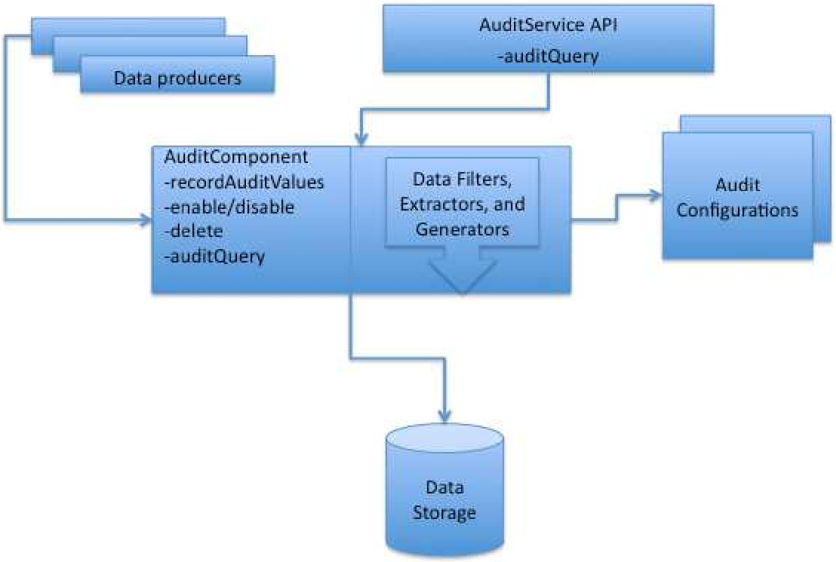
```
log4j.logger.org.alfresco.repo.transfer.fsr=warn
log4j.logger.org.alfresco.repo.web.scripts.transfer=warn
```

Auditing Alfresco

Alfresco provides the ability to audit activity. This section describes how Alfresco generates, stores, and retrieves auditing information.

The auditing mechanism prior to Version 3.4.0 has been removed but the old tables remain in the system. You can access the previous audit data but any new audit data will be directed to the new tables. Any customizations of the auditing feature must be rewritten using the new configuration files. All SQL-based queries used previously must be replaced by calls to the supplied APIs. The use of low-level SQL statements to retrieve data is not supported.

The architecture of the auditing features comprises the following components:



Data Producers defines the components that produce data that might be audited. Data producers do not need to know anything about how the data is stored. Data is generated and sent to the `AuditComponent.recordAuditValues` component. The only requirement is that each packet of data is a *Map* of data keyed by logical path names, which are specific to the producers.

The **AuditService** search should be used for data retrieval; however, for completeness, the following tables are used:

- Tables exclusive to the new audit (*AlfrescoPostCreate-3.2-AuditTables.sql*)
    - `alf_audit_model`: Contains the record of the audit configuration files.
    - `alf_audit_application`: Contains an entry for each logical application. There may be several audit applications defined in a single audit model.
    - `alf_audit_entry`: Contains an entry for each call to `AuditComponent.recordAuditValues`. There is a reference to a property.
- Shared tables (*AlfrescoPostCreate-3.2-PropertyValueTables.sql*)
    - `alf_prop_root`: Entry point for properties: shared values are fully indexed; arbitrarily-deep collections; quick data query and reconstruction.

Audit configuration and environment

Configuration and environment	Details
Tomcat environment	• Set the configuration properties in the `alfresco-global.properties` file. • Log4J settings can be added in a file `<tomcat>/shared/classes/alfresco/extension/audit-log.properties`.
View the available web scripts and details	Use the following scripts: • Script index: http://localhost:8080/alfresco/service/ • Audit scripts: http://localhost:8080/alfresco/service/index/package/org/alfresco/repository/audit
HTTP client	• `curl` will be used as the HTTP client
Sample files	• Audit sample files are distributed in the `<extension>/audit` directory. Activate the sample files by removing the `.sample` extension.

Check the state of auditing on the server:

```
% curl -u admin:admin "http://localhost:8080/alfresco/service/api/audit/
control"
{
   "enabled" : false,
   "applications":
   [
   ]
}
```

Audit filters

This section describes how to use Alfresco global properties to **filter audit data** generated by any audit data producer.

Audit data producers call *AuditComponent.recordAuditValues(rootPath, auditMap)* once for each event to be audited. Filters are applied to reject events so that their values are never used by **audit configurations**. The *rootPath* identifies the data producer and the *auditMap* is the event data. The *rootPath* value and keys in the map represent a tree structure.

Example rootPath and auditMap

The last component in the *rootPath* is considered by the *AuditFilter* to be the event **action**. The keys in an audit map identify each audit value. Global properties may be defined to accept or reject each value. If any value in an audit map is rejected, the whole map is rejected. So that one does not have to define too many properties, a *default* event action property may be defined. This will be inherited by all actions unless a property is defined for a particular event action.

```
rootPath:
    /alfresco-access/transaction

auditMap:
    "action"         => "MOVE"
    "node"           => "workspace://SpacesStore/90a398d1-8e0d-462a-8c3b-
f0b17a2d1143"
    "move/from/node" => "workspace://SpacesStore/
a82446e9-4dca-49d2-9ce0-4526687fb310"
    "move/from/path" => "/app:company_home/st:sites/cm:fred/cm:documentLibrary/
cm:folder1"
    "move/from/type" => "cm:folder"
    "move/to/node"   => "workspace://
SpacesStore/517bd4d0-99bc-47ad-8cd7-5d425f94c7db"
    "move/to/path"   => "/app:company_home/st:sites/cm:fred/cm:documentLibrary"
    "move/to/type"   => "cm:folder"
    "path"           => "/app:company_home/st:sites/cm:fred/cm:documentLibrary/
cm:Word 123.docx"
    "sub-actions"    => "moveNode readContent"
    "type"           => "cm:content"
    "user"           => "admin"
```

Example filter

Each property value defines a list of regular expressions that will be used to match the actual audit map values.

```
audit.filter.alfresco-access.default.enabled=true
audit.filter.alfresco-access.default.user=~System;.*
audit.filter.alfresco-access.default.type=cm:folder;cm:content
audit.filter.alfresco-access.default.path=/app:company_home/.*
audit.filter.alfresco-access.transaction.user=
audit.filter.alfresco-access.login.user=jblogs
...
```

In the above example, events created by any user except for the internal user *"System"* will be recorded by default for all event actions. However the property for the *transaction* event action overrides this to record even *"System"* events.

For any filters to be applied to an event action, that action's filters must be enabled with an *"enabled"* property set to *"true"*. However this may also be done by using the *default* event action, as shown above. Property names have a *"audit.filter."* prefix and use '.' as a separator where as components of rootPath and keys in the audit map use '/'.

Lists are evaluated from left to right allowing one flexibility to accept or reject different combinations of values. If no match is made by the end of the list the value is rejected. If there is not a property for a given value or an empty list is defined (as above for the *"user"* value on a *"transaction"* action) any value is accepted. Each regular expression in the list is separated by a semicolon (';'). Expressions that include a semicolon may be escaped using a '\'. An expression that starts with a '~' indicates that any matching value should be rejected. If the first character of an expression needs to be a '~', it too may be escaped with a '\'.

A property value may be a reference to another property, which saves having multiple copies of the same regular expression. This is indicated by a '$' as the first character of the property value. If the first character of an expression needs to be a '$' it too may be escaped with a '\'.

Redirected properties

```
audit.filter.alfresco-access.transaction.type=$transaction.content.types

transaction.content.types=$general.content.types
general.content.types=cm:folder;cm:content
```

Debug information

The PropertyAuditFilter provides log4j debug information (in the alfresco.log file) when it rejects values. Turning on this debug can generate large volumes of output. **Enable debug**

```
# Change file appender to include debug from any source
log4j.appender.File.Threshold=debug

# Enable debug from the PropertyAuditFilter
log4j.logger.org.alfresco.repo.audit.PropertyAuditFilter=debug
```

Audit filter customizations

You can define additional filter properties and override predefined filter values.

If you are using the Tomcat web application server, add the additional properties to the `<tomcat>/shared/classes/alfresco-global.properties` file.

Content auditing

This section describes how to use Alfresco to audit actions performed on your content and folders, including a technical overview, and also examples of how to customize the standard configuration.

Content auditing technical overview

The data producer `org.alfresco.repo.audit.access.AccessAuditor` gathers together lower events into user recognizable events. For example, the download or preview of content are recorded as a single read. Similarly the upload of a new version of a document is recorded as a single create version. By contrast the `AuditMethodInterceptor` data producer typically would record multiple events.

A default audit configuration file located at `<alfresco.war>/WEB-INF/classes/alfresco/audit/alfresco-audit-access.xml` is provided that persists audit data for general use. This may be enhanced to extract additional data of interest to specific installations. For ease of use, login success, login failure and logout events are also persisted by the default configuration.

Default audit filter settings are also provided for the `AccessAuditor` data producer, so that internal events are not reported. These settings may be customized (by setting global properties) to include or exclude auditing of specific areas of the repository, users or some other value included in the audit data created by `AccessAuditor`.

No additional functionality is provided for the retrieval of persisted audit data, as all data is stored in the standard way, and so is accessible using the `AuditService` search, audit web scripts, database queries, and Alfresco Explorer `show_audit.ftl` preview.

Example audit trail

The user actions from the Share interface were:

1. Create a new folder called **My Documents**.
2. Upload a document (`The fox.odt`).
3. Preview of the document.
4. Update the meta data.
5. Upload a new version.
6. Copy the document to a folder called **MyPictures**.
7. Delete the copy of the document.

   In the example, the property values show "..." to indication that they are truncated.

```
1. /alfresco-access/transaction/action=CREATE
   /alfresco-access/transaction/aspects/add=[cm:titled]
   /alfresco-access/transaction/path=/app:company_home/st:sites/
cm:mysite/cm:documentLibrary/cm:My Documents
   /alfresco-access/transaction/properties/add=...
   /alfresco-access/transaction/sub-actions=createNode
 updateNodeProperties addNodeAspect
   /alfresco-access/transaction/type=cm:folder
   /alfresco-access/transaction/user=admin

2. /alfresco-access/transaction/action=CREATE
   /alfresco-access/transaction/aspects/add=[cm:titled, cm:author]
   /alfresco-access/transaction/path=/app:company_home/st:sites/
cm:mysite/cm:documentLibrary/cm:My Documents/cm:The fox.odt
```

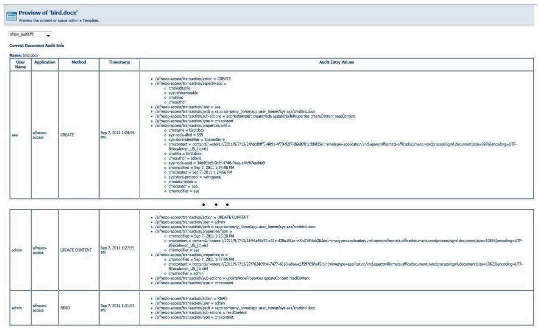
```
   /alfresco-access/transaction/properties/add=...
   /alfresco-access/transaction/sub-actions=createNode
 updateNodeProperties readContent createContent updateContent
 addNodeAspect
   /alfresco-access/transaction/type=cm:content
   /alfresco-access/transaction/user=admin

3. /alfresco-access/transaction/action=READ
   /alfresco-access/transaction/path=/app:company_home/st:sites/
cm:mysite/cm:documentLibrary/cm:My Documents/cm:The fox.odt
   /alfresco-access/transaction/sub-actions=readContent
   /alfresco-access/transaction/type=cm:content
   /alfresco-access/transaction/user=admin

4. /alfresco-access/transaction/action=updateNodeProperties
   /alfresco-access/transaction/aspects/add=[cm:taggable]
   /alfresco-access/transaction/path=/app:company_home/st:sites/
cm:mysite/cm:documentLibrary/cm:My Documents/cm:The fox.odt
   /alfresco-access/transaction/properties/add=...
   /alfresco-access/transaction/properties/from={cm:modified=Mon Jun 13
 15:34:05 BST 2011}
   /alfresco-access/transaction/properties/to={cm:modified=Mon Jun 13
 15:39:35 BST 2011}
   /alfresco-access/transaction/sub-actions=updateNodeProperties
 addNodeAspect readContent
   /alfresco-access/transaction/type=cm:content
   /alfresco-access/transaction/user=admin

5. /alfresco-access/transaction/action=CHECK IN
   /alfresco-access/transaction/aspects/add=[cm:versionable]
   /alfresco-access/transaction/copy/from/path=/app:company_home/
st:sites/cm:mysite/cm:documentLibrary/cm:My  Documents/cm:The fox
 (Working Copy).odt
   /alfresco-access/transaction/path=/app:company_home/st:sites/
cm:mysite/cm:documentLibrary/cm:My Documents/cm:The fox.odt
   /alfresco-access/transaction/properties/add=...
   /alfresco-access/transaction/properties/from=...
   /alfresco-access/transaction/properties/to=...
   /alfresco-access/transaction/sub-actions=updateNodeProperties
 addNodeAspect createVersion readContent deleteNodeAspect updateContent
 copyNode checkIn
   /alfresco-access/transaction/type=cm:content
   /alfresco-access/transaction/user=admin
   /alfresco-access/transaction/version=2.0

6. /alfresco-access/transaction/action=COPY
   /alfresco-access/transaction/aspects/add=[cm:titled, cm:copiedfrom,
 cm:author, cm:taggable]
   /alfresco-access/transaction/copy/from/path=/app:company_home/
st:sites/cm:mysite/cm:documentLibrary/cm:My  Documents/cm:The fox.odt
   /alfresco-access/transaction/path=/app:company_home/st:sites/
cm:mysite/cm:documentLibrary/cm:My Pictures/cm:The fox.odt
   /alfresco-access/transaction/properties/add=...
   /alfresco-access/transaction/sub-actions=createNode readContent
 createContent updateNodeProperties addNodeAspect copyNode
   /alfresco-access/transaction/type=cm:content
   /alfresco-access/transaction/user=admin

7. /alfresco-access/transaction/action=DELETE
   /alfresco-access/transaction/path=/app:company_home/st:sites/
cm:mysite/cm:documentLibrary/cm:My Pictures/cm:The fox.odt
   /alfresco-access/transaction/sub-actions=deleteNode
   /alfresco-access/transaction/type=cm:content
   /alfresco-access/transaction/user=admin
```

Using Alfresco Explorer to view the example audit trail

1. Locate the content.

2. Select **Preview in Template**, using the `show_audit.ftl` preview.

   The following is an example of some high level events.



Preview of 'bird.docx'

Preview the content or space within a Template.

show_audit.ftl

Current Document Audit Info

Name: bird.docx

User Name	Application	Method	Timestamp	Audit Entry Values
aaa	alfresco-access	CREATE	Sep 7, 2011 1:24:56 PM	• /alfresco-access/transaction/action = CREATE • /alfresco-access/transaction/aspects/add = ◦ cm:auditable ◦ sys:referenceable ◦ cm:titled ◦ cm:author • /alfresco-access/transaction/user = aaa • /alfresco-access/transaction/path = /app:company_home/app:user_homes/sys:aaa/cm:bird.docx • /alfresco-access/transaction/sub-actions = addNodeAspect createNode updateNodeProperties createContent readContent • /alfresco-access/transaction/type = cm:content • /alfresco-access/transaction/properties/add = ◦ cm:name = bird.docx ◦ sys:node-dbid = 559 ◦ sys:store-identifier = SpacesStore ◦ cm:content = contentUrl=store://2011/9/7/13/24/dcdbfff3-4091-4f79-92f7-d6e0781cdd4f.bin|mimetype=application/vnd.openxmlformats-officedocument.wordprocessingml.document|size=9876|encoding=UTF-8|locale=en_US_|id=61 ◦ cm:title = bird.docx ◦ cm:author = adavis ◦ sys:node-uuid = 34d465d9-0c9f-4748-9eea-c44fb7ead4a9 ◦ cm:modified = Sep 7, 2011 1:24:56 PM ◦ cm:created = Sep 7, 2011 1:24:56 PM ◦ sys:store-protocol = workspace ◦ cm:description = ◦ cm:creator = aaa ◦ cm:modifier = aaa

• • •

User Name	Application	Method	Timestamp	Audit Entry Values
admin	alfresco-access	UPDATE CONTENT	Sep 7, 2011 1:27:55 PM	• /alfresco-access/transaction/action = UPDATE CONTENT • /alfresco-access/transaction/user = admin • /alfresco-access/transaction/path = /app:company_home/app:user_homes/sys:aaa/cm:bird.docx • /alfresco-access/transaction/properties/from = ◦ cm:modified = Sep 7, 2011 1:25:30 PM ◦ cm:content = contentUrl=store://2011/9/7/13/25/f4a49a92-c62a-439a-88ac-b00d7454bb26.bin|mimetype=application/vnd.openxmlformats-officedocument.wordprocessingml.document|size=10854|encoding=UTF-8|locale=en_US_|id=63 ◦ cm:modifier = aaa • /alfresco-access/transaction/properties/to = ◦ cm:modified = Sep 7, 2011 1:27:55 PM ◦ cm:content = contentUrl=store://2011/9/7/13/27/702409b4-7477-4818-a8ae-c1f597098a45.bin|mimetype=application/vnd.openxmlformats-officedocument.wordprocessingml.document|size=10623|encoding=UTF-8|locale=en_US_|id=64 ◦ cm:modifier = admin • /alfresco-access/transaction/sub-actions = updateNodeProperties updateContent readContent • /alfresco-access/transaction/type = cm:content
admin	alfresco-access	READ	Sep 7, 2011 1:31:03 PM	• /alfresco-access/transaction/action = READ • /alfresco-access/transaction/user = admin • /alfresco-access/transaction/path = /app:company_home/app:user_homes/sys:aaa/cm:bird.docx • /alfresco-access/transaction/sub-actions = readContent • /alfresco-access/transaction/type = cm:content

Enabling auditing of content

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

```
# Enable audit in general
audit.enabled=true

# Enable the alfresco-access audit application
audit.alfresco-access.enabled=true

# Enable the auditing of sub-actions. Normally disabled as these values
 are
# not normally needed by audit configurations, but may be useful to
# developers
#audit.alfresco-access.sub-actions.enabled=true
```

Default audit filter settings

These values result in events only being recorded for common actions initiated by users of the system. These values may be overridden if required.

```
audit.filter.alfresco-access.default.enabled=true
audit.filter.alfresco-access.transaction.user=~System;~null;.*
audit.filter.alfresco-access.transaction.type=cm:folder;cm:content;st:site
audit.filter.alfresco-access.transaction.path=~/sys:archivedItem;~/ver:;.*
```

Audit data generated by AccessAuditor

The **/sub-action/<sequence>** structure holds cut down details of each sub-action, but are only included if the global property *audit.alfresco-access.sub-actions.enabled=true*.

The structure of the audit data is shown as follows:

```
/alfresco-access
      /transaction
        /action=<actionNamegt
        /sub-actions=<sub action listgt
        /path=<prefixPathgt
        /type=<prefixTypegt
        /node=<nodeRefgt
        /user=<usergt
        /copy
          /from
            /node=<nodeRefgt
            /path=<prefixPathgt
            /type=<prefixTypegt
        /move
          /from
            /node=<nodeRefgt
            /path=<prefixPathgt
            /type=<prefixTypegt
        /properties
           /from=<mapOfValuesgt
             /<propertyNamegt=<propertyValuegt
             ...
           /to=<mapOfValuesgt
             /<propertyNamegt=<propertyValuegt
             ...
           /add=<mapOfValuesgt
             /<propertyNamegt=<propertyValuegt
             ...
           /delete=<mapOfValuesgt
             /<propertyNamegt=<propertyValuegt
             ...
         /aspects
           /add=<mapOfNamesgt
             /<aspectNamegt=null
             ...
           /delete=<mapOfNamesgt
             /<aspectNamegt=null
             ...
         /version-properties=<mapOfValuesgt
         /sub-action/<sequencegt
           /action=<actionNamegt
           /copy
             ...
           /move
             ...
           /properties
             ...
           /aspects
             ...
```

An example of audit data is shown as follows:

```
Inbound audit values:
    /alfresco-access/transaction/action=MOVE
     /alfresco-access/transaction/node=workspace://
SpacesStore/74a5985a-45dd-4698-82db-8eaeff9df8d7
     /alfresco-access/transaction/move/from/node=workspace://SpacesStore/
d8a0dfd8-fe45-47da-acc2-fd8df9ea2b2e
     /alfresco-access/transaction/move/from/path=/app:company_home/st:sites/
cm:abc/cm:documentLibrary/cm:folder1/cm:Word 123.docx
     /alfresco-access/transaction/move/from/type=cm:folder
     /alfresco-access/transaction/path=/app:company_home/st:sites/cm:abc/
cm:documentLibrary/cm:folder2/cm:Word 123.docx
     /alfresco-access/transaction/sub-actions=moveNode readContent
     /alfresco-access/transaction/type=cm:content
     /alfresco-access/transaction/user=admin
     /alfresco-access/transaction/sub-action/00/action=moveNode
     /alfresco-access/transaction/sub-action/00/move/from/node=workspace://
SpacesStore/d8a0dfd8-fe45-47da-acc2-fd8df9ea2b2e
     /alfresco-access/transaction/sub-action/00/move/from/path=/
app:company_home/st:sites/cm:abc/cm:documentLibrary/cm:folder1/cm:folder1/
cm:Word 123.docx
     /alfresco-access/transaction/sub-action/00/move/from/type=cm:folder
     /alfresco-access/transaction/sub-action/01/action=readContent
```

Persisted audit data

The default structure of the persisted audit data is shown as follows:

```
/alfresco-access
  /login/user=<usergt
  /loginFailure/user=<usergt
  /logout/user=<usergt
  /transaction/
   /action=<actionNamegt
   /sub-actions=<sub action listgt
   /path=<prefixPathgt
   /type=<prefixTypegt
   /user=<usergt
   /version=<versiongt
   /copy/from/path=<prefixPathgt
   /move
     /from/path=<prefixPathgt
   /properties
      /from=<mapOfValuesgt
      /to=<mapOfValuesgt
      /add=<mapOfValuesgt
      /delete=<mapOfValuesgt
      /fromName=<oldNamegt
      /toName=<newNamegt
    /aspects
      /add=<mapOfNamesgt
      /delete=<mapOfNamesgt
```

The `version` value is sourced from either the `add/cm:versionLabel` or `to/cm:versionLabel` values.

The exception is the property `name`, individual property and aspect changes are not included, as it is not possible to know all possible names. The map values of all changes is however included. The individual property `name` value is included as it is a well known property, which changes if content or a folder is renamed within the same parent folder.

Content auditing customizations

- Creating a custom audit filter
- Creating a custom configuration

Custom audit filter

These filter values are used to include or exclude selected events. Global property names identifies elements in the generated audit data. Each property value is a list of regular expressions that either accept or reject the generated data value. If any value is rejected in a set of data the whole set is rejected. For example, to audit the users `"jblogs"` and any user that starts with `"temp"` other than `"tempmanager"`, override the following global property value. If using tomcat, add a value to the `<tomcat>/shared/classes/alfresco-global.properties` file.

The following is an example custom filter:

```
audit.filter.alfresco-access.transaction.user=~tempManager;test.*;jblogs
```

The list is semicolon separated. Any regular expression that starts with a '~' indicates that a matching value should be rejected. The list is evaluated from left to right until there is a match. If no match is made the value is rejected. If the list is empty (zero length) all values are accepted. It is possible to filter on any of the generated data values. Refer to the audit filtering section for a more detailed description of filter properties.

Custom audit configuration

For example, a security clearance level has been added to content and it is important to include that clearly in the persisted audit data, rather than having to find it deep within a map of all properties. The default configuration includes an example. It extracts the `name` property. It is generally a good idea to create a new audit configuration file that includes a mapped path to avoid confusion with the default. If running under Tomcat place the audit configuration file in the `<tomcat>/shared/classes/alfresco/extension/audit` directory. The following example is simply a cut down version of the default with the path mapped to a new value.

The following is an example of the `myApp.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<Audit xmlns="http://www.alfresco.org/repo/audit/model/3.2"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.alfresco.org/repo/audit/model/3.2
    alfresco-audit-3.2.xsd">

  <DataExtractors>
    <DataExtractor name="simpleValue"
        registeredName="auditModel.extractor.simpleValue"/>
  </DataExtractors>

  <PathMappings>
    <PathMap source="/alfresco-access" target="/my-app" />
  </PathMappings>

  <Application name="my-app" key="my-app">
    <RecordValue
        key="action" dataExtractor="simpleValue"
        dataSource="/my-app/transaction/action"
        dataTrigger="/my-app/transaction/action" />
    <RecordValue
        key="user" dataExtractor="simpleValue"
        dataSource="/my-app/transaction/user"
        dataTrigger="/my-app/transaction/user" />
    <RecordValue
        key="path" dataExtractor="simpleValue"
        dataSource="/my-app/transaction/path"
        dataTrigger="/my-app/transaction/path" />
  </Application>

</Audit>
```

The following shows the AccessAuditor debug for a move action.

```
Audit data:
    /my-app/action=MOVE
    /my-app/path=/app:company_home/st:sites/cm:fred/cm:documentLibrary/cm:Word
 123.docx
    /my-app/user=admin

Inbound audit values:
    /alfresco-access/transaction/action=MOVE
    /alfresco-access/transaction/node=workspace://
SpacesStore/90a398d1-8e0d-462a-8c3b-f0b17a2d1143
    /alfresco-access/transaction/move/from/node=workspace://SpacesStore/
a82446e9-4dca-49d2-9ce0-4526687fb310
    /alfresco-access/transaction/move/from/path=/app:company_home/st:sites/
cm:fred/cm:documentLibrary/cm:folder1/cm:Word 123.docx
    /alfresco-access/transaction/move/from/type=cm:folder
    /alfresco-access/transaction/path=/app:company_home/st:sites/cm:fred/
cm:documentLibrary/cm:Word 123.docx
    /alfresco-access/transaction/sub-action/00/action=moveNode
    /alfresco-access/transaction/sub-action/00/move/from/node=workspace://
SpacesStore/a82446e9-4dca-49d2-9ce0-4526687fb310
    /alfresco-access/transaction/sub-action/00/move/from/path=/
app:company_home/st:sites/cm:fred/cm:documentLibrary/cm:folder1/cm:Word
 123.docx
    /alfresco-access/transaction/sub-action/00/move/from/type=cm:folder
    /alfresco-access/transaction/sub-action/01/action=readContent
    /alfresco-access/transaction/sub-actions=moveNode readContent
    /alfresco-access/transaction/type=cm:content
    /alfresco-access/transaction/user=admin
```

Sample files

Samples can also be downloaded directly from the following location in svn:

```
http://svn.alfresco.com/repos/alfresco-open-mirror/alfresco/HEAD/root/projects/
repository/config/alfresco/extension/audit/
```

When using a sample file, remove the `.sample` extension.

Enabling auditing

This topic describes the auditing properties and how to enable auditing using these properties.

The different auditing properties are:

- `audit.enabled=true`
- `audit.tagging.enabled=true`
- `audit.alfresco-access.enabled=false`
- `audit.alfresco-access.sub-actions.enabled=false`
- `audit.cmischangelog.enabled=false`
- `audit.dod5015.enabled=false`

It is recommended that you do not change the `audit.enabled=true`, `audit.tagging.enabled=true` and `audit.dod5015.enabled=false` auditing properties. However, you can modify the `audit.alfresco-access.enabled=false` and `audit.cmischangelog.enabled=false` propeties to enable/disable a particular audit application.

Generation of audit data is disabled by default. To enable auditing permanently, settings must be added to the Alfresco global properties file as shown in the following text. To enable auditing permanently, add the following setting to the `alfresco-global.properties` file:

```
audit.alfresco-access.enabled=true
```

Auditing is enabled by default. To enable generation of audit data that you can view in Explorer or Share, you will need to enable the `audit.alfresco-access.enabled` property.

Once changes to the global properties file have been saved, you will need to restart the Alfresco server, for auditing to be fully enabled.

You can check the status of auditing conveniently from the command line by using a tool such as `curl` to access the Audit Control web script.

To check the global status of auditing issue a command, such as:

```
curl -u admin:password "http://localhost:8080/alfresco/service/api/audit/
control"
```

This invokes the web script with a GET request. This will result in a JSON response such as the following if auditing is currently enabled:

```
{
   "enabled" : true,
   "applications":
   [
      {
         "name": "Alfresco Tagging Service",
         "path" : "/tagging",
         "enabled" : true
      }
         ,
      {
         "name": "alfresco-access",
         "path" : "/alfresco-access",
         "enabled" : true
      }

   ]
}
```

While this does return the global status of the auditing framework, audit data will only be generated if `audit.alfresco-access.enabled` is enabled.

Auditing can also be globally enabled or disabled using the control web script. To do this a POST request is sent to the web script. For example, using `curl`, auditing can be enabled using the following command:

```
curl -u admin:password -d "" "http://localhost:8080/alfresco/service/api/audit/
control?enable=true"
```

This results in the following response:

```
{
   "enabled" : true
}
```

To disable auditing issue the following command:

```
curl -u admin:password -d "" "http://localhost:8080/alfresco/service/api/audit/
control?enable=false"
```

This results in the following response:

```
{
   "enabled" : false
}
```

While the global status of the auditing framework can be switched on and off in this manner, audit data will only be generated if `audit.alfresco-access.enabled` is enabled in the global properties file.

Enabling or disabling auditing using the Audit Control web script only remains valid in force while the server is running; the setting will not be retained following a server restart, but will subsequently be set according to the values in `alfresco-global.properties`.

Using JMX to control auditing

A JMX client can be used to access global properties. The properties can be modified using the JMX client. A server restart is not required for changes to properties to take effect.

Auditing examples

Audit data passed to recordAuditValues():

```
Root path:
   /alfresco-api/post/NodeService/createStore
Map:
   args/protocol = "workspace"
   args/identifier = "SpacesStore"
   result = StoreRef[workspace://SpacesStore]
```

If the root path passes the initial filtration phase - there is at least one component interested in auditing the information - then the map is expanded.

Expanded audit data:

```
Map:
   /alfresco-api/post/NodeService/createStore/args/protocol = "workspace"
   /alfresco-api/post/NodeService/createStore/args/identifier =
 "SpacesStore"
   /alfresco-api/post/NodeService/createStore/result = StoreRef[workspace://
SpacesStore]
```

The filtered data is then passed through the path mappings, generating a new "Map" of data for each application.

Path-mapped audit data:

```
Map:
   /MyApp/createStore = StoreRef[workspace://SpacesStore]
```

This data is then passed to any extractors and generators to produce a final "Map" of data that will be persisted.

Persisted audit data:

```
Map:
   /MyApp/createStore/value = StoreRef[workspace://SpacesStore]
   /MyApp/createStore/rootNode = NodeRef[workspace://SpacesStore/fd123...]
```

Audit configuration files

Audit configuration files are picked up automatically using the following search paths.

- `classpath*:alfresco/audit/*.xml`
- `classpath*:alfresco/enterprise/audit/*.xml`
- `classpath*:alfresco/module/*/audit/*.xml`
- `classpath*:alfresco/extension/audit/*.xml`

The XML schema is located at `<configRoot>/classes/alfresco/audit/alfresco-audit-3.2.xsd`.

The configuration file structure is divided into four basic sections:

<DataExtractors>

In this section, DataExtractors are declared for use in the `<Application>` sections of the configuration files. A DataExtractor is a component that uses input data to produce some output, either transforming the data or outputting the data verbatim. The simplest extractor is the `SimpleValueDataExtractor`, which returns whatever data is passed in. A more complex extractor is the `NodeNameDataExtractor`, which is able to produce the `cm:name` value of a node, assuming the data passed in is a NodeRef. For the complete set of built-in generators, see the `org.alfresco.repo.audit.extractor` package, or the `auditModel.extractor.*` beans, which are declared in `alfresco/audit-services-context.xml`.

The extractors can be declared in-line, for example:

```
    <DataExtractors>
       <DataExtractor name="simpleValue"
 class="org.alfresco.repo.audit.extractor.SimpleValueDataExtractor"/>
       ...
    </DataExtractors>
```

Or they can be declared in Spring configuration and referenced in the audit configuration (see the `alfresco/audit-services-context.xml` file), for example:

```
    <DataExtractors>
       <DataExtractor name="simpleValue"
 registeredName="auditModel.extractor.simpleValue"/>
       ...
    </DataExtractors>
```

<DataGenerators>

In this section, DataGenerators are declared for use in the `<Application>` sections of the configuration files. A DataGenerator is a component that produces data without any input, that is, data is produced when a data path is active, but is independent of the values at that path. Examples of generators are the `AuthenticatedUserDataGenerator` component, which produces the name of the currently-authenticated user (user in context) and the `AuthenticatedPersonDataGenerator` component, which produces the full name of the currently-authenticated user (person in context). For the complete set of built-in generators, see the `org.alfresco.repo.audit.generator` package or the `auditModel.generator.*` beans, which are declared in the `alfresco/audit-services-context.xml` file.

The generators can be declared in-line, for example:

```
    <DataGenerators>
       <DataGenerator name="currentUser"
 class="org.alfresco.repo.audit.generator.AuthenticatedUserDataGenerator"/>
       <DataGenerator name="personFullName"
 class="org.alfresco.repo.audit.generator.AuthenticatedPersonDataGenerator"/
>
    </DataGenerators>
```

Or they can be declared in Spring configuration and referenced in the audit configuration (see the `alfresco/audit-services-context.xml` file), for example:

```
    <DataGenerators>
       <DataGenerator name="currentUser"
 registeredName="auditModel.generator.user"/>
       <DataGenerator name="personFullName"
 registeredName="auditModel.generator.personFullName"/>
    </DataGenerators>
```

<PathMappings>

The expanded map coming from the Data Producers is passed through the path mappings. This is a raw remapping of the input data based on the path names in the data map.

```
    <PathMappings>
        <PathMap source="/DOD5015" target="/DOD5015"/>
        <!-- Force the fullName generator to trigger -->
        <PathMap source="/DOD5015/event/node" target="/DOD5015/event/
person"/>
        <PathMap source="/alfresco-api/post/AuthenticationService/
authenticate" target="/DOD5015/login"/>
    </PathMappings>
```

In this example, all paths starting with `/DOD5015` are mapped verbatim, but without the declaration, the data paths starting with `/DOD5015` are discarded. A small subset of the Alfresco API data is used (only the `AuthenticationService.authenticate` call) by mapping all values starting with that path to `/DOD5015/login`.

<Application>

This section defines how the mapped data is to be used by DataGenerators or by DataExtractors.

```
    <Application name="DOD5015" key="DOD5015">
        <AuditPath key="login">
            <AuditPath key="args">
                <AuditPath key="userName">
                    <RecordValue key="value" dataExtractor="simpleValue"/>
                </AuditPath>
            </AuditPath>
            <AuditPath key="no-error">
                <GenerateValue key="fullName"
 dataGenerator="personFullName"/>
            </AuditPath>
            <AuditPath key="error">
                <RecordValue key="value" dataExtractor="nullValue"/>
            </AuditPath>
        </AuditPath>
    </Application>
```

Built-in data producers

- `org.alfresco.repo.audit.AuditMethodInterceptor`: Generates audit data for all public service API calls. Refer to the javadocs for the data structure.
- `org.alfresco.repo.node.NodeAuditor`: Generates audit data for `beforeDeleteNode`

It is possible for any server-side component to pass data to the `auditComponent` bean.

To see what information is available to audit, enable the following logging:

```
log4j.logger.org.alfresco.repo.audit.inbound=DEBUG
```

The following is an example of output generated when a node is deleted from Alfresco Explorer:

```
15:55:26,590 User:admin DEBUG [repo.audit.inbound]
Inbound audit values:
 /alfresco-node/beforeDeleteNode/node=workspace://SpacesStore/
c4728f24-4a11-40f7-9062-315edf959d79
15:55:26,748 User:admin DEBUG [repo.audit.inbound]
Inbound audit values:
 /alfresco-api/post/NodeService/deleteNode/no-error=null
 /alfresco-api/post/NodeService/deleteNode/args/nodeRef=workspace://
SpacesStore/c4728f24-4a11-40f7-9062-315edf959d79
```

DataExtractors and DataGenerators

It is possible for any server-side component to pass data to the `auditComponent` bean.

DataExtractor

Uses an inbound mapped value as the source of the data. *AuditExampleLogin1* records values quite literally using the *simpleValue* data extractor.

DataGenerator

Activates when an inbound mapped path is present, but is not dependent on the value on that path. *AuditExampleLogin2* triggers the *personFullName* generator when the *authenticate/no-error* path is present; this records the full name of the currently-authenticated user even though the inbound data for *authenticate/no-error* is *null*.

Look at the data recorded for the two sample applications:

```
% curl -u admin:admin "http://localhost:8080/alfresco/service/api/audit/query/
AuditExampleLogin1?verbose=true&forward=false&limit=1"
{
   "count":1,
   "entries":
   [
      {
         "id":137,
         "application":AuditExampleLogin1,
         "user":admin,
         "time":"2010-09-20T17:37:14.699+01:00",
         "values":
         {
                     "\/auditexamplelogin1\/login\/no-error\/user":"admin"
         }

      }
   ]
}
% curl -u admin:admin "http://localhost:8080/alfresco/service/api/audit/query/
AuditExampleLogin2?verbose=true&forward=false&limit=1"
{
   "count":1,
   "entries":
   [
      {
         "id":138,
         "application":AuditExampleLogin2,
         "user":admin,
         "time":"2010-09-20T17:37:23.101+01:00",
         "values":
         {
                     "\/auditexamplelogin2\/login\/user":"Administrator"
         }

      }
   ]
}
```

Locating the audit code

This section describes the location of audit code.

1. For DEBUG logging, to see which data is being produced, rejected, or recorded, enable DEBUG for:

   ```
   log4j.logger.org.alfresco.repo.audit.AuditComponentImpl=DEBUG
   ```

2. For JUnit code, the unit test code demonstrates use of the Audit APIs and configuration:

   ```
   org.alfresco.repo.audit.AuditComponentTest
   ```

   - `alfresco-audit-test-authenticationservice.xml`: This is used by the test to capture both successful and unsuccessful login attempts in the audit data.
   - `testAuditAuthenticationService`: This demonstrates the use of the `auditSearch` method.

3. For Records Management (DOD5015) and auditing, the module pulls in audit data from the '`AuthenticationService`' but adds more data around the individual actions that take place during Records Management processes.

   ```
   org.alfresco.module.org_alfresco_module_dod5015.audit.*
   ```

   - `RecordsManagementAuditServiceImpl$RMAuditTxnListener`: This transaction listener generates Records Management-specific data for events (it is a `Data Producer`). It generates node property deltas.
   - `config/alfresco/module/org_alfresco_module_dod5015/audit/rm-audit.xml`: This defines how the data produced by the `AuthenticationService` and the Records Management module is persisted. There are some custom `DataGenerators` and `DataRecorders`.

- `RecordsManagementAuditServiceImpl.getAuditTrailImpl`: This method demonstrates how the Records Management use-case searches the audit data. Further query extensions are required to extend the search filters available using the `auditQuery` API.

Defining the audit application

This section describes the audit applications.

Data producers have no knowledge of how or whether data will be stored. Different use cases need to store or modify inbound data independently, therefore the use cases are separated into audit applications. Each application defines how data is mapped, extracted, and recorded without affecting data required by other applications.

For example, the Records Management module records before and after values when specific nodes are modified, whereas the CMIS standard requires a slightly different set of data to be recorded. Additionally, each of the audit logs can be enabled and disabled independently within the same server. Usually, each audit application is defined in its own configuration file, but for demonstration purposes, multiple application definitions can be defined in one configuration file.

1. Enable the sample file by removing the `.sample` extension.

   `alfresco/extensions/audit/alfresco-audit-example-login.xml.sample`
2. Restart the Alfresco server.
3. Ensure that the applications have been registered properly and are enabled:

```
% curl -u admin:admin "http://localhost:8080/alfresco/service/api/audit/
control"
{
   "enabled" : true,
   "applications":
   [
      {
         "name": "AuditExampleLogin1",
         "path" : "/auditexamplelogin1",
         "enabled" : true
      }
         ,
      {
         "name": "AuditExampleLogin2",
         "path" : "/auditexamplelogin2",
         "enabled" : true
      }
         ,
      {
         "name": "CMISChangeLog",
         "path" : "/CMISChangeLog",
         "enabled" : true
      }

   ]
}
```

4. At an application level, auditing is enabled or disabled for specific paths; changes made to an application's audit state are persisted. To disable all auditing for an application, disable the root path; in this case, disable the root path for the `CMISChangeLog` application. If you restart the server you will see that the application remains disabled.

```
% curl -u admin:admin -d "" "http://localhost:8080/alfresco/service/api/
audit/control/CMISChangeLog/CMISChangeLog?enable=false"
{
   "enabled" : false
}
```

Simple audit query

This section describes a simple audit query example.

1. Generate some auditing data for the sample applications.
2. Connect to the Alfresco Explorer client.
3. Login as the `admin` user.
4. Logout of Alfresco.
5. Login as the `admin` user but use an illegal password.

   The following examples are two queries to return results: without and with full-audited values respectively. Some entries have been replaced with a (**...**) for brevity.

```
% curl -u admin:admin "http://localhost:8080/alfresco/service/api/audit/
query/AuditExampleLogin1"
{
   "count":4,
   "entries":
   [
      {
         "id":69,
         "application":AuditExampleLogin1,
         "user":admin,
         "time":"2010-09-20T14:45:28.998+01:00",
         "values":
null
      },
      ...
   ]
}
% curl -u admin:admin "http://localhost:8080/alfresco/service/api/audit/
query/AuditExampleLogin1?verbose=true"
{
   "count":5,
   "entries":
   [
      ...
      {
         "id":72,
         "application":AuditExampleLogin1,
         "user":null,
         "time":"2010-09-20T14:45:43.884+01:00",
         "values":
         {
                     "\/auditexamplelogin1\/login\/error\/user":"admin"
         }

      },
      ...
      {
         "id":76,
         "application":AuditExampleLogin1,
         "user":admin,
         "time":"2010-09-20T14:46:23.319+01:00",
         "values":
         {
                     "\/auditexamplelogin1\/login\/no-error\/
user":"admin"
         }

      }
   ]
}
```

There is no count function in the search API. This is by design; use the `limit` parameter instead.

6. Assume that a client wants to see the details of the latest two results but knows of the existence of the next eight results. In this case, it would be pointless pulling back full (`verbose=true`) results for the latest 10 entries. Instead, pull back the last two results with values and then pull back the next eight results without values.

   Notice that the response contains a count of the number of entries returned; the individual entries are provided so that the entry IDs can be used for further result retrieval.

```
% curl -u admin:admin "http://localhost:8080/alfresco/service/api/audit/
query/AuditExampleLogin1?verbose=true&limit=2&forward=false"
{
   "count":2,
   "entries":
   [
      {
         "id":98,
         "application":AuditExampleLogin1,
         "user":admin,
         "time":"2010-09-20T15:10:04.043+01:00",
         "values":
         {
                     "\/auditexamplelogin1\/login\/no-error\/
user":"admin"
         }

      },
      {
         "id":96,
         "application":AuditExampleLogin1,
         "user":admin,
         "time":"2010-09-20T15:09:50.117+01:00",
         "values":
         {
                     "\/auditexamplelogin1\/login\/no-error\/
user":"admin"
         }

      }
   ]
}
% curl -u admin:admin "http://localhost:8080/alfresco/service/api/audit/
query/AuditExampleLogin1?verbose=false&limit=8&forward=false&toId=96"
{
   "count":8,
   "entries":
   [
      {
         "id":94,
         "application":AuditExampleLogin1,
         "user":admin,
         "time":"2010-09-20T15:09:47.606+01:00",
         "values":
null
      },
      ...
      {
         "id":80,
         "application":AuditExampleLogin1,
         "user":admin,
         "time":"2010-09-20T14:58:34.305+01:00",
         "values":
null
      }
   ]
}
```

The `fromTime` and `toTime` parameters are based on the generic millisecond timestamp format. For example the timestamp 1305899677764 can be converted to a standard time-date format: Fri, 20 May 2011 13:54:37 GMT.

For more information, see the following topic Alfresco Audit Service Query.

Advanced audit query

This section describes an advanced audit query example.

This type of query URL makes use of a data path within the audit application. This allows entries to be found that match specific audited values. By default, query values are treated as case-sensitive string types, but it is possible to specify the type to query against.

1. Generate some audit data.
2. Connect to the Alfresco Explorer client.
3. Attempt a failed login as `joe`.

```
% curl -u admin:admin "http://localhost:8080/alfresco/service/api/
audit/query/AuditExampleLogin1/auditexamplelogin1/login/error/user?
verbose=true&value=joe"
{
   "count":1,
   "entries":
   [
      {
         "id":101,
         "application":AuditExampleLogin1,
         "user":null,
         "time":"2010-09-20T15:13:57.947+01:00",
         "values":
         {
                     "\/auditexamplelogin1\/login\/error\/user":"joe"
         }

      }
   ]
}
% curl -u admin:admin "http://localhost:8080/alfresco/service/api/
audit/query/AuditExampleLogin1/auditexamplelogin1/login/error/user?
verbose=true&value=JOE"
{
   "count":0,
   "entries":
   [
   ]
}
```

Understanding PathMappings

To create an audit configuration file, it is necessary to know which data can be audited and how the data is mapped onto your application.

1. Turn on debugging for the inbound data. For a better understanding, you can turn on debug logging for the mapping components as well, although this is more verbose.

```
% cat <tomcatgt/shared/classes/alfresco/extension/audit-log4j.properties
log4j.logger.org.alfresco.repo.audit.AuditComponentImpl=DEBUG
log4j.logger.org.alfresco.repo.audit.inbound=DEBUG
```

2. Tail the log file and watch the output.

   a. Login as admin.

```
16:47:37,434  DEBUG [repo.audit.inbound]
```

```
Inbound audit values:
 /alfresco-api/pre/AuthenticationService/authenticate/args/
userName=admin
16:47:37,443 User:admin DEBUG [repo.audit.inbound]
Inbound audit values:
 /alfresco-api/post/AuthenticationService/authenticate/no-error=null
 /alfresco-api/post/AuthenticationService/authenticate/args/
userName=admin
```

b. From the inbound values (and if you have the `AuditComponentImpl` debugging on):

```
16:47:37,445 User:System DEBUG [repo.audit.AuditComponentImpl]
 Extracted audit data:
   Application: AuditApplication[ name=AuditExampleLogin2, id=7,
 disabledPathsId=7]
   Raw values:  {/auditexamplelogin2/login=null}
   Extracted:   {}
16:47:37,447 User:admin DEBUG [repo.audit.AuditComponentImpl] New
 audit entry:
   Application ID: 7
   Entry ID:       130
   Values:         {/auditexamplelogin2/login=null}
   Audit Data:     {/auditexamplelogin2/login/user=Administrator}
16:47:37,447 User:System DEBUG [repo.audit.AuditComponentImpl]
 Extracted audit data:
   Application: AuditApplication[ name=AuditExampleLogin1, id=6,
 disabledPathsId=6]
   Raw values:  {/auditexamplelogin1/login/no-error=null, /
auditexamplelogin1/login/args/userName=admin}
   Extracted:   {/auditexamplelogin1/login/no-error/user=admin}
16:47:37,449 User:admin DEBUG [repo.audit.AuditComponentImpl] New
 audit entry:
   Application ID: 6
   Entry ID:       131
   Values:         {/auditexamplelogin1/login/no-error=null, /
auditexamplelogin1/login/args/userName=admin}
   Audit Data:     {/auditexamplelogin1/login/no-error/user=admin}
```

You can see that the `AuthenticationService.authenticate` method generate two sets of "inbound" data: the `/alfresco-api/pre/AuthenticationService/authenticate` data is passed through before the service call is processed; the `/alfresco-api/post/AuthenticationService/authenticate` data is passed through after the service call has been processed. When logging in successfully, the post-call data is generated with a `no-error` path.

c. Perform a failed login with user joe.

```
17:02:09,697  DEBUG [repo.audit.inbound]
Inbound audit values:
 /alfresco-api/pre/AuthenticationService/authenticate/args/
userName=joe
17:02:09,704  DEBUG [repo.audit.inbound]
Inbound audit values:
 /alfresco-api/post/AuthenticationService/authenticate/error=08200014
 Failed to authenticate
   Started at:

 org.alfresco.repo.security.authentication.AbstractChainingAuthenticationServi
      ...
```

This is translated and recorded:

```
17:02:09,704 User:System DEBUG [repo.audit.AuditComponentImpl]
 Extracted audit data:
   Application: AuditApplication[ name=AuditExampleLogin1, id=6,
 disabledPathsId=6]
   Raw values:  {/auditexamplelogin1/login/error=08200014 Failed to
 authenticate
   Started at:
```

```
 org.alfresco.repo.security.authentication.AbstractChainingAuthenticationServi
      ...
17:02:09,704  DEBUG [repo.audit.AuditComponentImpl] New audit entry:
   Application ID: 6
   E6try ID:       135
   Values:         {/auditexamplelogin1/login/error=08200016 Failed to
 authenticate
   Started at:

 org.alfresco.repo.security.authentication.AbstractChainingAuthenticationServi
      ...
   Audit Data:     {/auditexamplelogin1/login/error/user=joe}
```

d. Notice that the failed login did not generate any data for audit application `AuditExampleLogin2`. To understand this, look at the `PathMappings` section of the example:

```
<PathMappings>
        <PathMap source="/alfresco-api/post/AuthenticationService/
authenticate" target="/auditexamplelogin1/login"/>
        <PathMap source="/alfresco-api/post/AuthenticationService/
authenticate/no-error" target="/auditexamplelogin2/login"/>
    </PathMappings>
```

Before any data is considered for persistence, the inbound data paths are remapped using the `PathMappings` configuration. The `/auditexamplelogin2/login` path is mapped onto `.../no-error` only, so failed logins were not recorded for the `AuditExampleLogin2` audit application, while the `AuditExampleLogin1` application recorded both successful and failed logins.

Audit recording values

The `RecordValue` element makes use of the `DataExtractor` definitions, but specifies when to be activated (`dataTrigger`) and where to get the data from (`dataSource`). Both the `dataTrigger` and `dataSource` attributes default to the path of the `RecordValue` element. Data is always written to the path where the `RecordValue` is declared. So, it is possible to trigger the `RecordValue` when a data path is present (such as a `null` value) and then to read a value from a completely different location.

1. Activate sample `/audit/alfresco-audit-example-extractors.xml` file.
2. Restart Alfresco (or restart the Audit subsystem).
3. Tail the log to capture `createNode` calls:

   ```
   tail -f ../logs/catalina.out | grep -G "createNode" -A 200 -B 20
   ```

4. Login to explorer and add some content under **Company Home**.

   ```
   20:18:52,817 User:admin DEBUG [repo.audit.AuditComponentImpl]
   New audit entry:
    Application ID: 8
    Entry ID:       177
    Values:
     /auditexampleextractors/args/properties=...
     /auditexampleextractors/args/assocQName={http://www.alfresco.org/model/
   content/1.0}alfresco.log
     /auditexampleextractors/args/parentRef=workspace://
   SpacesStore/37884669-0607-4527-940d-cb34b4f07d75
     /auditexampleextractors/no-error=null
     /auditexampleextractors/args/assocTypeQName={http://www.alfresco.org/
   model/content/1.0}contains
     /auditexampleextractors/args/nodeTypeQName={http://www.alfresco.org/
   model/content/1.0}content
     /auditexampleextractors/result=workspace://
   SpacesStore/37884669-0607-4527-940d-cb34b4f07d75|workspace://SpacesStore/
   c0fabc6d-903f-4317-87d1-ec62de37089c|...
   ```

```
 Audit Data:
  /auditexampleextractors/create/out/a=workspace://
SpacesStore/37884669-0607-4527-940d-cb34b4f07d75|workspace://SpacesStore/
c0fabc6d-903f-4317-87d1-ec62de37089c|...
  /auditexampleextractors/create/derived/parent-node-name=Company Home
  /auditexampleextractors/create/derived/parent-node-null=null
  /auditexampleextractors/create/in/c={http://www.alfresco.org/model/
content/1.0}contains
  /auditexampleextractors/create/in/d={http://www.alfresco.org/model/
content/1.0}alfresco.log
  /auditexampleextractors/create/in/a=workspace://
SpacesStore/37884669-0607-4527-940d-cb34b4f07d75
  /auditexampleextractors/create/derived/parent-node-type={http://
www.alfresco.org/model/content/1.0}folder
  /auditexampleextractors/create/in/b={http://www.alfresco.org/model/
content/1.0}content
```

5. View the audited data using the query API:

```
% curl -u admin:admin "http://localhost:8080/alfresco/service/api/audit/
query/AuditExampleExtractors?limit=1&forward=false&verbose=true"
{
   "count":1,
   "entries":
   [
      {
         "id":177,
         "application":AuditExampleExtractors,
         "user":admin,
         "time":"2010-09-20T20:18:52.761+01:00",
         "values":
         {
                     "\/auditexampleextractors\/create\/out\/
a":"workspace:\/\/SpacesStore\/37884669-0607-4527-940d-cb34b4f07d75|
workspace:\/\/SpacesStore\/c0fabc6d-903f-4317-87d1-ec62de37089c|...
                     ,"\/auditexampleextractors\/create\/derived\/parent-
node-name":"Company Home"
                     ,"\/auditexampleextractors\/create\/in\/c":"{http:\/
\/www.alfresco.org\/model\/content\/1.0}contains"
                     ,"\/auditexampleextractors\/create\/in\/d":"{http:\/
\/www.alfresco.org\/model\/content\/1.0}alfresco.log"
                     ,"\/auditexampleextractors\/create\/in\/
a":"workspace:\/\/SpacesStore\/37884669-0607-4527-940d-cb34b4f07d75"
                     ,"\/auditexampleextractors\/create\/derived\/parent-
node-type":"{http:\/\/www.alfresco.org\/model\/content\/1.0}folder"
                     ,"\/auditexampleextractors\/create\/in\/b":"{http:\/
\/www.alfresco.org\/model\/content\/1.0}content"
         }

      }
   ]
}
```

The `/no-error` path was used as the `dataTrigger` to activate all the `RecordValue` elements, that is, the presence of the path triggered the data rather than any specific value. `/create/derived/...` audit values show how the parent node reference was used to record values that were not part of the inbound data set.

Using the example, to search for values that are not strings, use the following:

```
% curl -u admin:admin "http://localhost:8080/alfresco/service/api/audit/query/
AuditExampleExtractors/ \
                 auditexampleextractors/create/derived/parent-node-type?
                       \
                 valueType=org.alfresco.service.namespace.QName&
                       \
                 value=%7Bhttp://www.alfresco.org/model/content/1.0%7Dfolder"
{
   "count":1,
   "entries":
   [
      {
         "id":177,
         "application":AuditExampleExtractors,
         "user":admin,
         "time":"2010-09-20T20:18:52.761+01:00",
         "values":
null
      }
   ]
}

% curl -u admin:admin "http://localhost:8080/alfresco/service/api/audit/query/
AuditExampleExtractors/ \
                 auditexampleextractors/create/in/a?
                       \
                 valueType=org.alfresco.service.cmr.repository.NodeRef&
                       \
                 value=workspace://SpacesStore/37884669-0607-4527-940d-
cb34b4f07d75"
{
   "count":1,
   "entries":
   [
      {
         "id":177,
         "application":AuditExampleExtractors,
         "user":admin,
         "time":"2010-09-20T20:18:52.761+01:00",
         "values":
null
      }
   ]
}
```

It is not possible to restrict results to a specific value path. The path AND the value are enough to return a result. This does not usually yield duplicate results but it is not as restrictive as it should be. For example, generate the audit data and query for verbose output. Choose to search based on a path and a value and check that you get the correct number of results. Now choose a different path in the value list and query with that, that is, use a path and value that are not related.

Using values that have changed in a post method call

When using the `org.alfresco.repo.audit.AuditMethodInterceptor` Data Producer, which generates audit data for all public service API calls, it is sometimes useful to be able to audit before and after values in a 'post' call application, or to include values from before the call.

For example, the `nodeName` data extractor may only be called on a node that exists, so calling it after a delete has no effect.

The output of 'pre' call applications is available to 'post' call applications, which can be seen in the following example. The example shows auditing the deletion of nodes and includes the node name. The `nodeName` is evaluated in the 'pre' call application and copied in the 'post' call application.

```
<?xml version='1.0' encoding='UTF-8'?>
<Audit
  xmlns="http://www.alfresco.org/repo/audit/model/3.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.alfresco.org/repo/audit/model/3.2 alfresco-
audit-3.2.xsd" >

  <DataExtractors>
```

```
    <DataExtractor name="simpleValue"
 registeredName="auditModel.extractor.simpleValue"/>
    <DataExtractor name="nodeNameValue"
 registeredName="auditModel.extractor.nodeName"/>
  </DataExtractors>

  <PathMappings>
    <PathMap source="/alfresco-api/pre/NodeService/deleteNode" target="/
preDelete" />
    <PathMap source="/alfresco-api/post/NodeService/deleteNode" target="/
postDelete" />
  </PathMappings>

  <Application name="PreCallDataDelete" key="preDelete">
    <RecordValue key="nodeName" dataExtractor="nodeNameValue" dataSource="/
preDelete/args/nodeRef" dataTrigger="/preDelete/args/nodeRef" />
  </Application>

  <Application name="PostDelete" key="postDelete">
    <RecordValue key="error" dataExtractor="simpleValue" dataSource="/
postDelete/error" dataTrigger="/postDelete/error" />
    <AuditPath key="deleteDetails">
      <RecordValue key="deletedNodeRef" dataExtractor="simpleValue"
 dataSource="/postDelete/args/nodeRef" dataTrigger="/postDelete/args/nodeRef" /
>
      <RecordValue key="nodeName" dataExtractor="simpleValue" dataSource="/
postDelete/preCallData/preDelete/nodeName" dataTrigger="/postDelete/
preCallData/preDelete/nodeName" />
    </AuditPath>
  </Application>

</Audit>
```

The `dataSource` attribute of the final `<RecordValue>` element includes the output path of the 'pre' call application ("`preDelete/nodeName`"). This is prefixed by `preCallData/` much like the `args/` prefix for method arguments. To avoid 'pre' call applications from generating audit records themselves, rather than just generating output for the 'post' call applications, give them a name that starts with `PreCallData`.

Administering Explorer from the Administration Console

The Administration Console enables Alfresco administrators to create and manage users and groups, manage categories, import and export spaces and content, and perform other administrative tasks from within Alfresco Explorer.

Managing users

In Alfresco a home space is a place for users to store their items.

By default a user has full control over items in their space and other users are given guest access to that space.

Users can navigate to their home space by clicking **My Home** in the toolbar.

Creating a user

Only an Administrator can create a user.

This functionality may not be available. Please contact your System Administrator for more details.

1. In the toolbar, click **(Administration Console)**.

2. Click **Manage System Users**.
3. In the space header, click **Create User**.
4. In Step One, Person Properties, enter information about the user being created and click **Next**.
5. In Step Two, User Properties, provide user information and click **Next**.

   The password is case sensitive. You must provide a **Home Space Name** to create the user's home space.
6. In Step Three, Summary, check that all information entered is correct and click **Finish**.

   To view the new user, use the search feature in the **Users** pane or click **Show All**.
7. Click **Close** to return to the **Administration Console**.

Editing user details

You cannot change the user’s password with this feature. Only an Administrator can edit user details.

This functionality may not be available. Please contact your System Administrator for more details.

1. In the toolbar, click **(Administration Console)**.
2. Click **Manage System Users**.
3. On the **Manage System Users** page, use the search feature to locate the user whose account you wish to edit or click **Show All** to display all users.

   Selecting to display all users may take some time if there are many users in the system.
4. Click **(Modify)** for the user of interest.
5. In the **Edit User Wizard**, modify the person and/or user properties as desired. Click **Next** and **Back** to work through the steps in the **Edit User Wizard**. Make your changes to the appropriate step(s).
6. Click **Finish** to process the changes.
7. Click **Close** to return to the **Administration Console**.

Changing a user's password

It is important to remember that passwords are case sensitive.

This functionality may not be available. Please contact your System Administrator for more details.

1. In the toolbar, click **(Administration Console)**.
2. Click **Manage System Users**.
3. On the **Manage System Users** page, use the search feature to locate the user whose account you wish to edit or click **Show All** to display all users.

   Selecting to display all users may take some time if there are many users in the system.
4. Click **(Change Password)** for the user of interest.
5. On the **Change Password** page, enter and confirm the new password for this user in the **Password** and **Confirm** boxes.

   The password is case sensitive.
6. Click **Finish** to process the change.
7. Click **Close** to return to the **Administration Console**.

Deleting a user

Only an Administrator can delete a user.

This functionality may not be available. Please contact your System Administrator for more details.

1. In the toolbar, click **(Administration Console)**.
2. Click **Manage System Users**.
3. On the **Manage System Users** page, use the search feature to locate the user whose account you wish to edit or click **Show All** to display all users.

   Selecting to display all users may take some time if there are many users in the system.
4. Click **(Delete)** for the user of interest.

   A message prompts you to confirm the deletion of the selected user.
5. Click **Yes**.
6. Click **Close** to return to the **Administration Console**.

Managing user groups

Groups allow you to quickly and easily give all members of a group abilities in a space that are specific to that group. Once you create a group, you manage its membership by inviting and removing users.

By default, all users are members of the group called **Everyone** with guest abilities. Users can belong to more than one group.

To assist with the management of the Alfresco administrative users, a default group, ALFRESCO_ADMINISTRATORS, has been created for you. This group enables you to easily configure the permissions for all Alfresco administrators. The initial administrator (the default "admin" user) can create additional administrative users by adding users to this group.

When a web project is created and users are invited to that project, a user group is created automatically in the Administration Console for the project. However, web project membership must be managed within the web project and not within the Administration Console.

The **Groups Management** page header provides functionality to toggle between the **Groups** view and the **Details** view. To aid navigation, a breadcrumb showing the groups hierarchy appears above the **Groups** list. Click the links in this breadcrumb to navigate the groups hierarchy.

Creating a user group

Top level groups reside beneath the heading **Root Groups**. A group can contain sub-groups. Only an Administrator can create a user group.

This functionality may not be available. Please contact your System Administrator for more details.

1. In the toolbar, click **(Administration Console)**.
2. Click **Manage User Groups**.
3. On the **Groups Management** page, use the search feature to locate a specific user group or click **Show All** to display the existing root, or top-level, groups.

   If not already displayed, click **Root Groups** beneath the space header to return to the top-level group.

4. Navigate to the user group where you want to create a group.
5. Click **Create Group**. To create a sub-group, click **(Create Sub-Group)** associated with the group you wish to be the parent.
6. On the **Create Group** page, enter the name of the group you are creating in the **Identifier** box.

   Once you provide an **Identifier** for the group, you cannot change it.
7. Click **Create Group**.

   An additional viewing option in the space header allows you to view either all groups and sub-groups beneath the currently selected group or only the immediate child groups of the currently selected group. To the right of the icon , select **All** or **Children**, as preferred. Once you set the filter option, click **Show All** to populate the Groups pane.
8. Click **Close** to return to the **Administration Console**.

Deleting a user group

Deleting a group also deletes all sub-groups and users associated with it. Only an Administrator can delete a user group.

This functionality may not be available. Please contact your System Administrator for more details.

1. In the toolbar, click **(Administration Console)**.
2. Click **Manage User Groups**.
3. On the **Groups Management** page, use the search feature to locate a specific user group or click **Show All** to display the existing root, or top-level, groups.

   If not already displayed, click **Root Groups** beneath the space header to return to the top-level group.
4. Navigate to the user group you want to delete.

   The page header displays the name of the selected group.
5. In the **More Actions** menu, click **Delete Group**.

   A message prompts you to confirm the deletion of the selected user group.
6. Click **Delete**.
7. Click **Close** to return to the **Administration Console**.

Adding a user to a user group

Only an Administrator can add a user to a user group.

This functionality may not be available. Please contact your System Administrator for more details.

1. In the toolbar, click **(Administration Console)**.
2. Click **Manage User Groups**.
3. On the **Groups Management** page, use the search feature to locate a specific user group or click **Show All** to display the existing root, or top-level, groups.

   If not already displayed, click **Root Groups** beneath the space header to return to the top-level group.
4. Navigate to the user group you want to add users to.

   The page header displays the name of the selected group.

5. In the **More Actions** menu, click **Add User**.
6. Use the search feature to locate users.

   You must enter a minimum of one (1) character.
7. Click to select the users you want to add to the group.

   Use SHIFT to select multiple, consecutive users; use CTRL to select multiple, nonconsecutive users.
8. Click **Add** to add the user(s) to the **Selected Users** list.

   Click **(Remove)** to remove a user from this list.
9. Click **OK**.
10. Click **Close** to return to the **Administration Console**.

Removing a user from a user group

Only an Administrator can remove a user from a user group.

This functionality may not be available. Please contact your System Administrator for more details.

1. In the toolbar, click **(Administration Console)**.
2. Click **Manage User Groups**.
3. On the **Groups Management** page, use the search feature to locate a specific user group or click **Show All** to display the existing root, or top-level, groups.

   If not already displayed, click **Root Groups** beneath the space header to return to the top-level group.
4. Navigate to the user group you want to remove users from.

   The page header displays the name of the selected group.
5. On the **Groups Management** page, click **(Remove)** for the user you want to remove from the group.

   The user is removed without a prompt to confirm the action.
6. Click **Close** to return to the **Administration Console**.

Managing categories

An Administrator creates and manages the categories, which are organized into related groups to form a hierarchy. Users then link content items to one or more categories to classify the content items.

The **Category Management** page header provides functionality to toggle between the **Categories** view and the **Details** view. To aid navigation, a hierarchy path appears beneath the page header. Click the links in this path to navigate the category hierarchy.

Adding a category

Only an Administrator can add a category.

1. In the toolbar, click **(Administration Console)**.
2. Click **Category Management**.
3. Click **Add Category** to create a top-level category.

   To create a sub-category, navigate the existing categories, select the category for which you are creating a sub-category, and click **Add Category**.

4. On the **New Category** page, type the relevant information in the **Name** and **Description** boxes.
5. Click **New Category**.
6. Click **Close** to return to the **Administration Console**.

Deleting a category

Deleting a category deletes its sub-categories and breaks any existing links between content items and the categories being deleted. Only an Administrator can delete a category.

1. In the toolbar, click **(Administration Console)**.
2. Click **Category Management**.
3. Navigate to the category you want to delete.

   The page header displays the name of the selected category.
4. Click **(Delete Category)**.

   Deleting a category also deletes its sub-categories. A message informs you if content is linked to the category you are deleting.

   A message prompts you to confirm the deletion of the selected category.
5. Click **Delete**.
6. Click **Close** to return to the **Administration Console**.

Editing a category

Changing the name does not break any links between content and the category. Only an Administrator can edit a category.

1. In the toolbar, click **(Administration Console)**.
2. Click **Category Management**.
3. Navigate to the category you want to edit.

   The page header displays the name of the selected category.
4. Click **(Edit Category)**.
5. On the **Edit Category** page, modify the properties as desired.
6. Click **Finish**.
7. Click **Close** to return to the **Administration Console**.

Importing the ACP file into a space

The Import function in the Administration Console enables you to import an exported space to any space within Alfresco or into another Alfresco repository altogether.

An exported space is bundled into an Alfresco Content Package (ACP). Importing the ACP into a new location expands the package to the original structure of the space.

1. Navigate to the space into which you want to import content.

   The space header displays the name and details of the space.
2. In the toolbar, click **(Administration Console)**.
3. Click **Import**.

   The space header indicates the space into which you will import your file.

4. Click **Browse** then locate and select the ACP file you want to import.
5. Check **Run import in background** if you want the import to occur while you are still working.
6. Click **OK**.

   The ACP file expands, putting the space, sub spaces, and content in the space.
7. Click **Close** to return to the current space.

Exporting a space and its contents

The Export function in the Administration Console enables you to copy an Alfresco space and its contents.

Exporting a space differs from copying a space in that the export function bundles all rules, workflow, properties, and metadata associated with the space into an Alfresco Content Package (ACP). You can import the ACP to a different space within Alfresco or into another Alfresco repository altogether.

1. Navigate to the space you want to export.

   The space header displays the name and details of the space.
2. In the toolbar, click **(Administration Console)**.
3. Click **Export**.

   The space header indicates the space selected for export.
4. On the **Export** page, type a name for the export package (ACP).
5. Select a destination location to store the resulting ACP file.
6. Select **Current Space** as what you would like to export from.
   a. Check **Include Children** if you want to export sub spaces.
   b. Check **Include this Space** if you want to export the selected space as well as the children.
7. Check **Run export in background** if you want the export to occur while you are still working.
8. Click **OK**.

   The ACP file is created and stored in the destination location.
9. Click **Close** to return to the current space.

Viewing System Information

1. In the toolbar, click **(Administration Console)**.
2. Click **System Information**.
3. Click ▶ to expand a pane to view its contents; click ▼ to collapse the pane.
4. Click **Close** to return to the **Administration Console**.

Using the Node Browser

This is a read-only feature with basic search capability.

1. In the toolbar, click **(Administration Console)**.
2. Click **Node Browser**.
3. On the **Alfresco Node Browser** page, click the store of interest.

Each store is an area of the repository and within each store, the nodes of that store are organized hierarchically. The node displayed is the root node of the selected store.

4. Search the selected store, as needed:
   a. Select the search type: **noderef**, **xpath**, **lucene**, **selectnodes**.
   b. Enter the search criteria in the field provided.
   c. Click **Search**.
5. Click **Close** to return to the **Administration Console**.

Administering Records Management

The administrator can manage Alfresco Records Management from the Management Console.

Users can only access the Management Console if they are members of the `ALFRESCO_ADMINISTRATORS` group.

Management Console

The Management Console allows you to manage the Records Management site.

You can manage the following Records Management-specific administration tasks:

- Audit
- Custom Metadata
- Define Roles
- Email Mappings
- Events
- List of Values
- Relationships
- User Rights Report

Accessing the Records Management Console

This task assumes that you have access to the Records Management site dashlet and that you are logged in as a user that is a member of the `ALFRESCO_ADMINISTRATORS` group.

In the Records Management dashlet, click **Management Console**

The Management Console displays with the Audit tool showing, by default.

Records Management Auditing

The function of the Records Management system is to manage and control electronic and physical records according to legislative and regulatory requirements, and it must be capable of demonstrating this compliance.

The Audit tool displays the auditing information collected from the system to show whether business rules are being followed and ensure that unauthorized activity can be identified and traced. The Audit tool is especially important for systems that deal with classified information.

The Audit tool maintains a complete trace of all the actions on every record and it cannot be altered. The information that is captured and stored includes:

- any action on any record, any container, or the File Plan
- user undertaking the action

- date and time of the action

The Audit tool displays by default when you access the Management Console.

Accessing the Audit tool

The Audit tool displays by default when you open the Management Console.

1. In the Records Management dashlet, click **Management Console**.
2. In the tools list, click **Audit**.

   The Audit page displays.
3. Click **Apply**.

The most recent entries in the log (up to 20) display in chronological order. You can see who performed each event, the user's role, and when it was performed. To see more information on a specific event, click **Details**.

Starting and stopping the audit log

This task assumes that you are on the Audit page of the Records Management Console and that the auditing tool is running.

1. Click **Stop**.

   A dialog box prompts you to confirm the action.
2. Click **Yes**.

   The auditing tool stops capturing and storing the activity in the Records Management system.
3. To start the audit log again, click **Start**. When prompted, click **Yes** to confirm the action.

Filtering the log entries

By default, the Audit page displays the last 20 entries in the log. You can filter by date and user to see a specific set of entries.

This task assumes that you are on the Audit page of the Records Management Console.

1. Use the fields in the box at the top of the page to filter the log entries. Use one, all, or a combination of filters to customize the entries returned.
   a. In the **Number of entries** field, specify the number of records you want to see.
   b. In the **From** and **To** fields, use the date picker to specify a single day (enter same date in both fields) or a date span.
   c. Click **Specify** and type the full or partial name of the user whose activities you want to see. Click **Search**. Click **Add** to select the user.

      You must enter a minimum of one (1) character. The search is not case sensitive.
2. Click **Apply** to view the matching log entries.

   Leaving the Audit page clears the fields and the results.

Filing the audit log as a record

This task assumes that you are on the Audit page of the Records Management Console and that it displays a list of log entries that you want to file.

1. Click **File as Record**.

   The **Select location of Audit Record** dialog box displays.
2. Choose the destination folder for the audit record.

3. Click **OK**.

   A message confirms that the audit log has been filed as an undeclared record in the selected folder in the File Plan.
4. Click **OK** to dismiss the message. You can also click **View Record** to display the audit report in the Records Management site.

Exporting the audit log

Export allows you to archive the audit log periodically and enables, for example, external auditors to examine or analyze system activity. When you export the audit log, this does not affect the audit log in the system. The audit log is exported as an HTML file.

This task assumes that you are on the Audit page of the Records Management Console and that it displays a list of log entries that you want to export.

1. Click **Export**.

   You are prompted to open or save the file.
2. Choose to save the file to your computer.

   Depending on your browser, you are either prompted to specify a destination or the item is automatically downloaded to a default location.
3. Close the dialog box once the download is complete.

Viewing the full log

You can view the full contents of the log file in a separate window. From there you can save an HTML version of the report on your computer or in the Records Management File Plan.

This task assumes that you are on the Audit page of the Records Management Console.

1. Click **View Full Log**.

   A separate window opens displaying the audit log.
2. Optionally, save the log report in one or both of the following ways:
   - Click **Export** to save the report to your computer.
   - Click **File as Record** to file the report as an undeclared record in the File Plan.
3. Close the window.

Clearing the audit log

Clearing the audit log deletes all captured actions.

This task assumes that you are on the Audit page of the Records Management Console.

1. Click **Clear**.

   A message prompts you to confirm the action.
2. Click **Yes** to clear the audit log.

Auditing actions

The type of action that is recorded in the audit log includes the following:

- capture of all electronic records: file, declare, undeclared
- re-categorization of an electronic record within the file plan: a move
- any change to any Disposition Schedule (instructions): create, modify, destroy
- any disposition actions carried out by authorized roles: cutoff, retain, transfer, review, close folder, reopen folder

- the placing or removal of a disposal hold (freeze) on an object: freeze, unfreeze
- any change made to any metadata associated with File Plan or electronic records, for example, change to vital record indicator
- amendment and deletion of metadata by a user
- any internal or user event triggered by the system or by the user, for example, SUPERSEDED, GAO Audit, End of Fiscal Year, and so on.
- changes made to the access permissions
- creation, amendment or deletion of a user or group
- changes made to the capabilities (functional access permissions)
- changes made to supplemental markings
- export and import
- deletion / destruction of records
- changes to the auditing levels and settings
- search operations carried out by users
- any access to any record or aggregation should be logged, if the access is for viewing, printing or otherwise presenting it, then the access should be marked as retrieval

Creating custom metadata

The Records Management installation defines a default set of metadata. The metadata is logged against each level in the File Plan and at the record level itself.

Accessing the Custom Metadata tool

1. In the Records Management dashlet, click **Management Console**.
2. In the tools list, click **Custom Metadata**.

   The Custom Metadata page displays.

Creating custom metadata

You can create custom metadata for record categories, record folders, records, and non-electronic documents. Once you create custom metadata, you cannot delete it.

This task assumes that you are on the Custom Metadata page of the Records Management Console.

1. Select an option in the Object column: **Record Category**, **Record Folder**, **Record**, or **Non-Electronic Document**.

   The right column lists any custom metadata that has already been defined for the object selected.
2. Click **New**.

   The **New Metadata** page displays.
3. Type a name for the metadata in the **Label** field.

   This name is used as the label on the Edit Metadata page.
4. Select the expected data value in the **Type** field.

   The type can be of the following values:

Type	Description
Text	Sets the value of this metadata to be a text string. When you select this option, you can set the **Use selection list** check box.

Type	Description
Boolean	Sets the value of this metadata to be either True or False.
Date	Sets the value of this metadata to be a numeric date sequence.

The **Use selection list** option appears only when a list has been created in the List of Values tool.

5. To configure this metadata field as a selection menu,:
   a. Select the **Use selection list** check box.
   b. Select a list name from the menu.
6. Select the **Mandatory Field** check box to set this metadata to be a mandatory entry on the Edit Metadata page.
7. Click **Create**.

The new metadata displays in the right column of the Custom Metadata page.

Editing custom metadata

This task assumes that you are on the Custom Metadata page of the Records Management Console.

1. Select an option in the Object column: **Record Category**, **Record Folder**, **Record**, or **Non-Electronic Document**.

   The right column lists the custom metadata defined for the object.
2. Click **Edit** to the right of the metadata you want to work with.

   The **Edit Metadata Property** page displays.
3. Make the required changes.
4. Click **Save**.

Defining roles and capabilities

User permissions are set in Records Management using a combination of roles and capabilities.

A role is a named collection of functional user access. A capability refers to the functions that can be assigned to each user.

A role can be assigned to one or more users; however, a user can be assigned only one role at a time.

Roles

The Administrator user has permission to add new roles. The Administrator defines the range of capabilities for each role. Each role can be assigned many capabilities.

Capabilities are not hierarchical, so when the Administrator assigns a capability, it does not grant further capabilities.

Roles for Records Management are:

- Records Management Administrator
- Records Management Power User
- Records Management Records Manager
- Records Management Security Officer
- Records Management User

Capabilities

A capability is an ability that can be granted to a user, which controls the behavior of the system. With respect to the user, this may be to grant a certain operation or privilege, or it may be to alter the behavior of the system for that user.

Capabilities cannot conflict and are not hierarchical. A user can be granted a single capability and that capability will not grant any further capabilities. Any user may have zero or more capabilities within the system. A user that has no capabilities is effectively barred from the records management system.

Accessing the Define Roles tool

1. In the Records Management dashlet, click **Management Console**.
2. In the tools list, click **Define Roles**.

   The Roles page displays. A table lists the current Records Management user roles.

Viewing the capabilities for a role

This task assumes that you are on the Roles page of the Records Management Console.

1. In the Roles column, select a role to view.

   The list of capabilities assigned to that role display in the Capabilities column.
2. Select another role to view its assigned capabilities.

Adding new roles

This task assumes that you are on the Roles page of the Records Management Console.

1. Click **New Role**.

   The **New Role** page displays all available capabilities, which are organized into groups. You can choose individual items or an entire group to define the permissions for the role you are creating.
2. Enter a name for the role.
3. Select the capabilities that you wish to apply to the role.
   a. To select an individual capability within a group, click the check box.
   b. To select a group of capabilities, click **Select All**.

      For example, to select all capabilities for controlling folders, select all the capabilities in the Folder Control group.
4. Click **Create**.

The new role displays in the list of available roles.

Editing a role

This task assumes that you are on the Roles page of the Records Management Console.

1. In the Roles column, select the role you want to edit.
2. Click **Edit Role**.
3. Edit the name and capabilities as necessary.
4. Click **Save**.

Deleting a role

This task assumes that you are on the Roles page of the Records Management Console.

1. In the Roles column, select the role you want to delete.
2. Click **Delete Role**.

   A message prompts you to confirm the action.
3. Click **OK**.

Mapping emails

The Alfresco Records Management system supports accessing the Alfresco server using IMAP. The protocol allows email applications that support IMAP to connect to and interact with Alfresco repositories directly from the mail application.

The Email Mappings tool provides a facility to map the fields of an email header to the properties stored in the repository.

For example, an email `Subject` heading is mapped to the Alfresco property `title`. When you are viewing emails within the Records Management system, the `title` property shows the email's `Subject` heading.

Accessing the Email Mappings tool

1. In the Records Management dashlet, click **Management Console**.
2. In the tools list, click **Email Mappings**.

   The Email Mappings page displays.

Default email mappings

The Email Mappings page shows a list of the current maps between email headers and Alfresco properties.

For example, the email Subject heading mapping to the property title is shown as: `messageSubject` to `cm:title`.

The email header field `messageSubject` is on the left and is separated by the word “to”, which indicates that it is mapped to a property `cm:title`.

Adding an email map

The pre-defined email mappings cover the most commonly used email headers. You can include additional email header mappings using the Email Mappings tool.

This task assumes that you are on the Email Mappings page of the Records Management Console.

1. Type the email header field in the **Map** box or select an email header from the menu.
2. Select the Alfresco property name from the **to** menu.

   You can select an Alfresco property or a custom property.
3. Click **Add**.

   The new mapping displays in the list of email mappings.
4. Click **Save**.

   A message prompts you to confirm the action.
5. Click **Yes**.

If you do not wish to save your changes, click **Discard Changes**.

Deleting an email map

This task assumes that you are on the Email Mappings page of the Records Management Console.

1. Browse the list to find the mapping you want to delete.
2. Click **Delete**.

   This removes the mapping from the list. You have to save the changes to the list for the deletion to take effect.
3. Click **Save**.

   A message prompts you to confirm the action.
4. Click **Yes**.

Managing events

In the Records Management system, events can control the life cycle of a record. There are several events that are defined internally to the system.

A system event is generated whenever an action occurs on a record or folder, such as versioned, cutoff, closed, superseded, and so on. These events can then be used in disposition instructions.

The following events are available in Records Management:

- Abolished
- All Allowances Granted Are Terminated
- Case Closed
- Case Complete
- No longer needed
- Obsolete
- Redesignated
- Related Record Transferred to Inactive Storage
- Separation
- Study Complete
- Superseded
- Training Complete
- WGI action complete

Accessing the Events tool

1. In the Records Management dashlet, click **Management Console**.
2. Click **Events**.

   The Events page displays.

Creating a new event

This task assumes that you are on the Events page of the Records Management Console.

1. Click **New Event**.

   The New Event page displays.
2. In the **Label** field, enter a name for the event.
3. In the **Type** field, select the event type.
4. Click **Save**.

The new event displays on the Events page.

Editing an event

This task assumes that you are in the Events page of the Records Management Console.

1. Browse the list to find the event you want to edit.
2. Click **Edit**.

   The Edit Event page displays.
3. Change the details as necessary.
4. Click **Save**.

Deleting an event

This task assumes that you are on the Events page of the Records Management Console.

1. Browse the list to find the event you want to delete.
2. Click **Delete**.

   A message prompts you to confirm the deletion.
3. Click **Yes**.

   The event is deleted from the Events page.

Creating a list of values

Throughout the Records Management system, there are metadata entry fields. The metadata can be simple text strings, Boolean values, or dates.

Where the value is a text string, you may also enter the value using a list of values menu. For example, on the Edit Metadata page, you enter the value for the Mimetype field by selecting a value from the menu.

There are two predefined lists in the Records Management installation that you can modify:

- Supplemental Markings
- Transfer Locations

Initially, these lists are empty. The administrator should populate these lists with appropriate values. You can also add your own lists of values.

Accessing the List of Values tool

1. In the Records Management dashlet, click **Management Console**.
2. Click **List of Values**.

   The Lists page displays.

Creating a list of values

Creating a list is a two step process. First you create the empty list and then you edit it to add the values. Once you create a list, you cannot delete it.

This task assumes that you are on the Lists page of the Records Management Console.

1. Click **New List**.

   The New List dialog box displays.
2. In the **Name** field, enter a name for the list.

   ✎ The list name must be unique.
3. Click **OK**.

The name of the new list displays on the Lists page.

Editing a list of values

Use the Edit feature to add and delete values for a list. You can also control the user and group access to the values in the list.

This task assumes that you are on the Lists page of the Records Management Console.

1. Locate the list you want to modify, and then click **Edit**.

   The Edit List page displays.
2. To add values to the list:
   1. In the empty field at the top of the page, type the new value.
   2. Click **Add**.

      The value name displays in the Values table.
3. To delete values from the list:
   1. In the Values table, locate the entry you want to remove.
   2. Click **Delete**.

      A message prompts you to confirm the deletion.
   3. Click **Yes**.

      The value name is removed from the table.
4. To control the user and group access to the individual values in the list:
   1. In the Values table, click the value you want to set access for.

      The selected value is highlighted.
   2. On the right side of the page, click **Add**.

      The Add Access dialog box displays.
   3. In the search field, type the full or partial name of a user or group.

      You must enter at least three (3) characters.
   4. Click **Search**.

      A list of users and groups matching the search criteria displays.
   5. Click **Add** to the right of the user or group you want to have access to the selected value.

      The user or group displays in the right column. You can add as many users and groups as required.
5. When you have finished editing the values and access, click **Done** to save all changes.

Renaming a list of values

This task assumes that you are on the Lists page of the Records Management Console.

1. Locate the list you want to rename, and then click **Rename**.

   The Rename List dialog box displays.
2. Edit the list name and then click **OK**.

The modified name displays on the Lists page.

Managing relationships

The References feature in the File Plan lets you create associations between records. Each reference is defined by a Relationship. Several default relationship types are included in the

Records Management Console. You create new and manage existing relationships with the Relationships tool.

Accessing the Relationships tool

1. In the Records Management dashlet, click **Management Console**.
2. In the tools list, click **Relationships**.

   The Link Relationships page displays.

Creating a new relationship

The Records Management Console has several default relationships. Add to these as necessary to meet your needs. Once you create a relationship, you cannot delete it.

This task assumes that you are on the Link Relationships page of the Records Management Console.

1. Click **New Relationship**.

   The New Relationship page displays.
2. Select the relationship type:
   - Bi-directional
   - Parent/Child
3. Complete the appropriate field(s) appropriate for the relationship type.

   The values you enter display in the File Plan when a user creates the reference.
4. Click **Save**.

   The new relationship appears in the list, which is sorted alphabetically.

Deleting a relationship

This task assumes that you are on the Link Relationships page of the Records Management Console.

1. Locate the relationship you want to modify, and then click **Edit**.

   The Edit Relationship page displays. You can't change the relationship type; only the field values can be modified.
2. Make the necessary changes:
   - If the relationship type is Bi-directional, edit the Label field.
   - If the relationship type is Parent/Child, edit the Source and Target fields.
3. Click **Save**.

Viewing the User Rights report

Access the User Rights report to view a summary of the Records Management site users, groups, and roles.

1. In the Records Management dashlet, click **Management Console**.
2. In the tools list, click **User Rights Report**.

   The User Rights page displays the report.

Administering Enterprise to Cloud Sync

This section describes the Enterprise to Cloud Sync feature available in Alfresco Share. It provides instructions on how to configure Cloud Sync and also troubleshoot issues that you might encounter.

Overview

This topic provides an overview of Enterprise to Cloud Sync.

Enterprise to Cloud Sync allows Alfresco Share on-premise users to synchronize content on to the Alfresco Cloud. Once content has been setup to synchronize, the cloud and on-premise instances of the documents are automatically synchronized with each other whenever either version is updated. This allows the on-premise Alfresco instance to act as the system of record and the cloud instance provides a system of engagement for external collaboration without requiring access to the on-premise system by 3rd parties. To set up Enterprise to Cloud Sync, you need an Alfresco Cloud account.

Enterprise to Cloud Sync features

In addition to the standard cloud features, Enterprise to Cloud Sync has many special features, it allows automatic synchronization of content between on-premise Alfresco repositories and Alfresco's cloud service. This helps in bringing about ease of engagement and collaboration within the Enterprise and the Cloud. This topic describes some of the features and capabilities available within Enterprise to Cloud Sync.

Enterprise to Cloud Sync allows content to be sent to an Alfresco's cloud service and also provides the ability to automatically synchronize content between on-premise Alfresco repositories and Alfresco's cloud service, thereby keeping the on-premise system in sync with any changes. The Enterprise to Cloud Sync capabilities include:

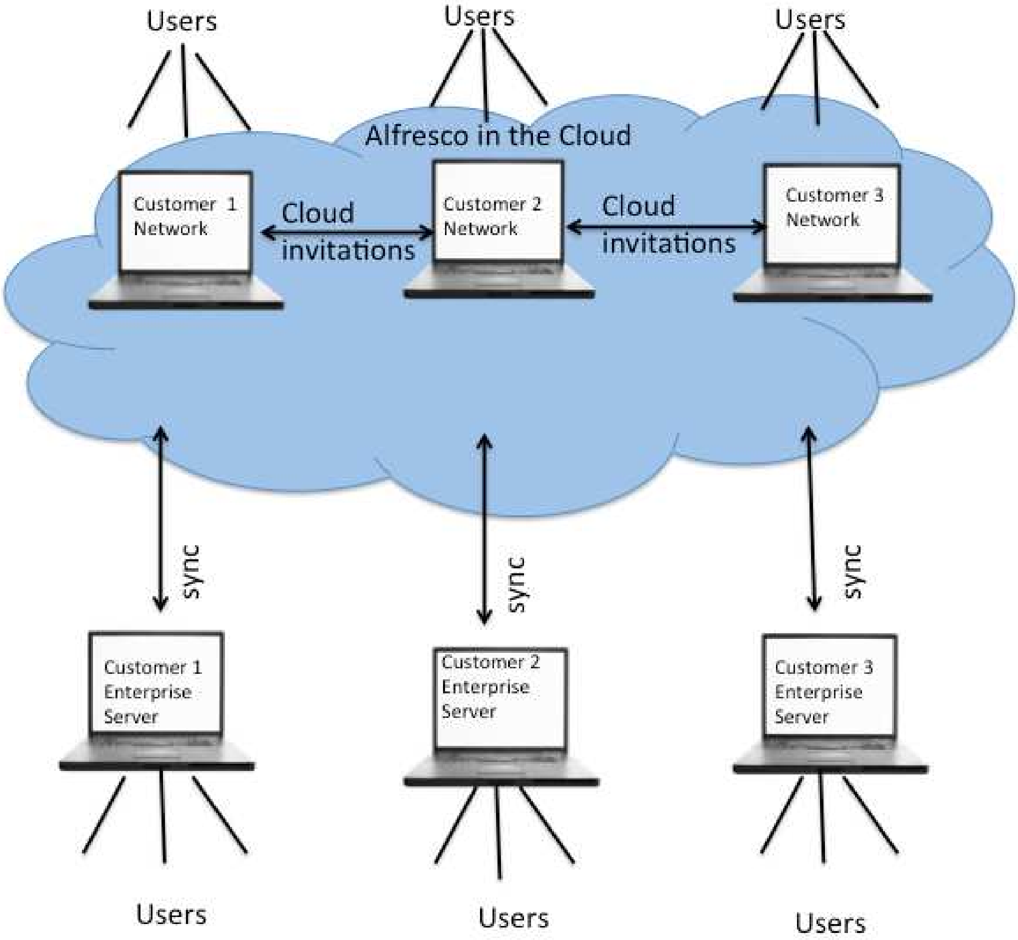
- Synchronization of individual and multiple files, folders, and folder hierarchies between on-premise and Alfresco Cloud.
- Inclusion of content and common metadata within synchronized payloads
- Automatic synchronization
- Secure exchange of information over HTTPS connection
- Initiation of all actions by the on-premise system. This makes the opening of a firewall redundant
- Choice over what can be synchronized to ensure sensitive content remains on-premise

Enterprise to Cloud Sync Logical overview

This topic gives a high level description of the architecture of the Enterprise to Cloud Sync feature.

Alfresco on the cloud has various Networks within it and every Network in turn includes users. These users can send cloud invitations to each other and interact with each other, thereby sharing content and information.

The Enterprise to Cloud feature allows Enterprise Alfresco (on-premise) users to synchronize content between their local Enterprise Alfresco and a Network on Alfresco Cloud, thereby providing access to other Alfresco Cloud users. The following diagram describes this flow.



Configuration

This topic provides you information on configuring the Enterprise to Cloud Sync feature within Alfresco Share. Configuring Enterprise to Cloud Sync involves either enabling the synchronization between Alfresco Cloud Network and your on-premise Alfresco server or turning it off.

Enabling Synchronization

This topic provides instructions on enabling the Enterprise to Cloud Sync feature within Alfresco Share.

To enable synchronization from your on-premise Alfresco server you need a Standard or Enterprise Alfresco Subscription and an Alfresco license file which enables the synchronization features. For more information on Alfresco subscriptions, see Pricing & Subscriptions. Ensure that you have access to port 443 and that you are able to access https.

Synchronization is enabled by default if your Alfresco license has synchronization enabled.

1. Copy the license file to your machine. The license file has a file extension of `.lic`.
2. From the Alfresco installation directory, browse to the `<extension>` directory.

   For example, for Tomcat on Windows, this is: `C:\Alfresco\tomcat\shared\classes\alfresco\extension`.
3. Create the `license` directory if it does not exist and move the `.lic` file into the license directory.
4. In Alfresco Share, click **More > Application > License Descriptor** .
5. Click **Edit**.
6. Click **Load License**.

You have now applied the license.

7. Verify that you have successfully enabled synchronization.
   a. Make sure that the **Sync to Cloud** action is available for documents and folders in the Alfresco Share Document Library.
   b. Make sure that the log contains the following message:

```
2012-09-04 13:38:50,458 INFO [repo.sync.SyncAdminServiceImpl] [main]
 A key is provided for cloud sync
```

To enable synchronization, you must set up Enterprise to Cloud Sync in your on-premise Alfresco. See Setting up Enterprise to Cloud Sync.

Disabling Synchronization

This topic provides instructions on how to disable synchronization between the Alfresco Cloud Network and your on-premise Alfresco server.

Synchronization is enabled by default if your Alfresco license has synchronization enabled. To disable synchronization, complete the following steps.

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

   For example for Tomcat 6, this is: `$TOMCAT_HOME/shared/classes/` directory.

2. Synchronization can be disabled either permanently or temporarily.

   To disable synchronization permanently, locate the `alfresco-global.properties` file and add the `sync.mode=OFF` property.

   To disable synchronization temporarily:

   - Locate the `alfresco-global.properties` file.
   - Leave the on-premise `sync.mode property` value unchanged.
   - Change both the `sync.pullJob.enabled` and the `sync.pushJob.enabled` properties to `False`.

   Alfresco recommends not using the `sync.mode` property to disable synchronization temporarily as this may result in the loss of internal synchronization related system data.

3. Restart your Alfresco server.

Troubleshooting

This section provides you information that will help you in locating and understanding unexpected Enterprise to Cloud Sync behavior and correcting it.

Best Practices

This topic describes the process of developing and following a standard way of doing things within Enterprise to Cloud Sync and provides guidance on some best practices while working with Enterprise to Cloud Sync.

- Make sure that the content you are syncing is not sensitive in nature. It is important to keep in mind that other users of the network may have access to your synced content.

Error Messages

This topic helps you understand and resolve various error messages in Enterprise to Cloud Sync.

The table below provides an explanation of some of the common error messages you might encounter while working on Enterprise to Cloud Sync.

End user error message	Description	Possible Causes	Solutions
Could not create sync	Authorized account is not suitable for synchronization	The network you are trying to authorize is not a standard, enterprise or partner network.	Create and authorize a standard, enterprise or partner network
Unable to connect to the sync server	Unable to connect to Alfresco Cloud.	The server on which Alfresco Cloud is running is disconnected.	Connect to the synchronization server. If the server is down, contact Alfresco Cloud support.
Could not remove sync	Unable to remove synchronization	Files within the synchronized folder or its subfolders are locked for editing in Alfresco Cloud.	Make sure that all files within the synchronized folder and its subfolders are unlocked for editing in Alfresco Cloud.
Could not request sync	Unable to put in a request to synchronize content from on-premise cloud into Alfresco Cloud.	Your on-premise Alfresco, cannot communicate with the Alfresco Cloud.	Make sure that your on-premise Alfresco is up and running and that you are logged in to it. Make sure that Alfresco Cloud is up and running.
A node already exists in the target folder with the same name	Unable to synchronize content between Alfresco Cloud and Alfresco Share on-premise	The node with the same name exists on Alfresco Cloud.	Rename the on-premise node and try syncing again or delete the node with the same name from Alfresco Cloud and try synchronizing again
Target folder could not be found	The folder specified as the target folder does not exist in Alfresco Cloud.	The folder specified as the target folder for the synchronization does not exist on Alfresco Cloud.	Specify a different target folder on Alfresco Cloud or create a new folder with a matching name.
Content cannot be created, it is already synchronized from somewhere else	Content with the same name cannot be synchronized twice to the same location in Alfresco Cloud	Different users are trying to synchronize content item at the same time to the same Alfresco Cloud target location.	Synchronize content to a different location
Content has already been synchronized from somewhere else	A content item can be synchronized only once and to one location in Alfresco Cloud.	The content item that you are trying to synchronize, has already been synchronized	Make sure that the content item does not exist anywhere else on Alfresco Cloud or that the content item has not already been synchronized.
Content no longer exists on the remote system	Unable to synchronize content as the content item no longer exists on cloud.	Content does not exist on Alfresco Cloud.	Make sure that the content item exists on Alfresco Cloud.
Content can not be updated, access denied	Unable to update content on Alfresco Cloud.	The user does not have permission to update content on Alfresco Cloud	Make sure that the user has the correct permissions to update content.

End user error message	Description	Possible Causes	Solutions
Content size violation (limit exceeded)	Unable to synchronize content on to Alfresco Cloud	The user has exceed the allocated content size limit for an individual file on a network.	Try to reduce the size of the content item, if that is not possible, please contact Alfresco Support to request an increase to content size limit for individual file for your cloud Network.
Quota violation (limit exceeded)	Unable to synchronize content on to Alfresco Cloud	User has exceeded the allocated quota of storage space on cloud	Try to reduce the size of the content item and/or empty your trashcan via the Account settings. If that is not possible, please contact Alfresco Support to request an increase to the overall storage space quota for your cloud Network.
Unable to push changes for this node. The authentication details are no longer valid.	Unable to make any changes to the content on to this node in Alfresco Cloud.	The user has not provided valid authentication details	Make sure that the user has valid authentication details to gain access to the cloud
Unable to push changes for this node. The owner no longer exists.	Unable to make any changes to the content on to this node in Alfresco Cloud.	The owner of this node no longer exists	Unsynchronize the content.
No network is enabled for sync	No network is enabled for synchronization	The user has not set the correct URL for Alfresco Cloud in alfresco-global.properties file	Set a valid URL for AlfrescoCloud in alfresco-global.properties file and run on-premise Alfresco again .

Frequently asked questions

These FAQs are designed to provide a better understanding of Enterprise to Cloud Sync.

Why can't I synchronize my content?

Synchronization problems in Enterprise to Cloud Sync can be caused by any of the following issues:

- You are logged on to the wrong cloud network type.
- You do not have network access to cloud.
- You do not have a valid on-premise licence key.
- Your new licence key is not sync- enabled.
- Your global cloud property has a wrong value.

To troubleshoot these issues perform the following steps, testing after each step to determine if the issue is resolved. If the issue is not resolved, continue to the next item in the list.

Check cloud network type

Make sure that your cloud network type is **Enterprise**. To do this, log into cloud and check your network type.

Check network access to cloud

Make sure that you have network access to cloud. There may be a communication problem either on the Alfresco Cloud side or with your on-premise instance. Please contact Alfresco customer support if the Alfresco Cloud Server is down.

Check validity of your on-premise licence key

Make sure that your on-premise licence has not expired.

Check that the `sync-mode` property has the right value

Make sure that the `sync-mode` property has the right value. You will not be able to synchronize, if this property has a wrong value. The default value for this property is ON_PREMISE and you do not need to amend it.

Check the value of the global cloud property

Make sure that the value for the Global cloud property is set to `https://a.alfresco.me/alfresco/a/{network}/`. This is the default value for this property and you do not need to amend it. You can set this property as follows.

```
sync.cloud.url=https://a.alfresco.me/alfresco/a/{network}/
```

How do I know if my content has only partially synchronized?

You encounter either of the following errors:

- This file exceeds the content limit. The file is too large to perform the action.
- You have exceeded the content quota. There isn't enough free space to perform the action.

Troubleshooting

This section provides help for diagnosing and resolving any Alfresco issues you may encounter.

For additional help, refer to the following:

- Alfresco Support Portal (http://support.alfresco.com)
- **Admin Console** in Alfresco Explorer to view various installation and setup information
- Alfresco Installation forum (http://forums.alfresco.com/)

Handling a higher rate of outbound TCP connections

1. Open the Registry.
2. Under the following registry entry:

   ```
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TCPIP\Parameters
   ```
3. Key in the registry of the Windows client machine.
4. Add the following registry entries:

   TcpTimedWaitDelay
   Add this DWORD with a value of 30.

   MaxUserPort
   Add this DWORD with a value of 32768.
5. Refer to the Windows documentation for further details on these registry entries.

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.
   For example, `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.
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. You can set the log level dynamically using the JMX client.

When using log4j, you should:

- Keep local customizations and licenses outside of the web application. For example, in the extension directory:

  ```
  $TOMCAT_HOME/shared/classes/alfresco/extension/...-log4j.properties
  ```
- The Alfresco supplied configuration files should be stored or installed within the web application. For example:

  ```
  WEB-INF/classes/alfresco/extension/...-log4j.properties
  ```

A `dev-log4j.properties` file should never be used in an ongoing during production, nor packaged as a part of any product.

Logging uses Log4J's `HierarchyDynamicMBean`.

This is not cluster-aware. If needed, the log level change 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.

- Editable attributes are a dynamic list of loggers with the `logLevel` attribute, which can be changed to ERROR, WARN, INFO, or DEBUG (editable).
- Operations with impact are `addLoggerMBean` - add logger, if it has been loaded.

Error messages

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 clustering

This topic provides additional troubleshooting tips for testing cache clustering.

- On Linux and Unix environments, you can use `netstat -ln` to check that the correct ports have been opened by the Alfresco server on the correct network adapters. You can use `telnet <hostname><port>` to check if each open port can be reached by each cluster member.
- If your cluster members are using NAT and IPv4 addresses, you may find it necessary to force the server to listen on IP V4 addresses rather than IP V6. To do this, add:

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

  to the startup options of Alfresco's JVM. In a standard Linux/Unix installation, this would require editing of the `JAVA_OPTS` variable in the script, as follow:

  ```
  tomcat/scripts/ctl.sh
  ```

  On a standard Windows installation, this would require adding the parameter just before `;-Dalfresco.home` in:

  ```
  tomcat/bin/service.bat
  ```

  and then running the scripts:

  ```
  tomcat/scripts/serviceinstall.bat REMOVE
  tomcat/scripts/serviceinstall.bat INSTALL
  ```

  to re-register the Alfresco service with the new option.

  For more information on the process of initiating clustering and the options available for configuring Alfresco clustering, see Initiating clustering.

Troubleshooting OpenOffice subsystems

This section provides help for troubleshooting the OpenOffice subsystems.

1. Enable the following log4j properties to debug:

   ```
   log4j.logger.org.alfresco.enterprise.repo.content=DEBUG
   log4j.logger.org.artofsolving.jodconverter=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.

   The correct value for the `jodconverter.officeHome` property varies with host operating system.

   - For Mac OS X, it should be set to the directory that contains `MacOS/soffice.bin`, which is `/Applications/OpenOffice.org.app/Contents` by default.
   - For other operating systems, it should be set to the directory that contains `program/soffice.bin`. For example, for Debian/Ubuntu, this may be `/usr/lib/openoffice`, for Fedora, `/opt/openoffice.org3`, and for Microsoft Windows, `C:/Alfresco/OpenOffice.org`.
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.
6. The JodConverter can run a pool of multiple reusable instances of the soffice OpenOffice process. To use this capability, set the `jodconverter.portNumbers` property to a comma-separated list of port numbers, all of which must be available for use. For example, `2022, 2023, 2024` for a pool of three `soffice` processes.
7. The JodConverter supports configurable restart behavior for the OpenOffice `soffice` process. To ensure that potential memory leaks within OpenOffice do not accumulate and affect performance, the JodConverter will restart an `soffice` process after a given number of tasks (transformations, metadata extractions) have been performed. The default for `jodConverter.maxTasksPerProcess` is 200.
8. The JodConverter allows long-running or hung tasks to be timed out. The first timeout is controlled by `jodconverter.taskQueueTimeout`, which is 30000 by default (30000 milliseconds = 30 seconds). If a task spends this long in a JodConverter queue awaiting execution, it will be dropped from the queue. The second timeout is controlled by `jodconverter.taskExecutionTimeout`, which is 120000 by default (120000 milliseconds = 2 minutes). If a task has been executing within an `soffice` process for longer than this period, that `soffice` process will be terminated and restarted.
9. Throughput of OOo-related tasks, such as transformations, can be balanced against available hardware resources (memory, CPU) by altering the pool size and the two timeout timers.

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.

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

Issue

Unable to connect to locally installed Alfresco server via CIFS.

Troubleshooting

To troubleshoot this issue, perform the following steps, testing after each step to determine if the issue is resolved. If the issue is not resolved, continue to the next item in the list.

- Check if Alfresco has finished loading. Look for a *Server startup* message in the log file.
- Does the connection work if you use the IP address instead of the host name.
- Check if Alfresco Share shows the `nbtstat -n` message on the Windows server.
- Check if Alfresco Share shows the `nbtstat -na <server ip>` message on the client machine.
- Enable all the `NB-Name-In` and `NB-Session_In` rules by navigating to **Windows Firewall Advanced Settings > Inbound Rules > File and Printer Sharing** tab.
- Check if Alfresco can bind to the necessary ports. To do so, check for errors in the log file related to `java.net.BindException` errors.
- Change the `cifs.serverName` property and restart Alfresco.
- If using Windows server, verify that the following DLLs are available to Alfresco (usually in `C:\Alfresco\tomcat\bin`):
  - Win32NetBIOS.dll

  - Win32NetBIOSx64.dll
  - Win32Utils.dll
  - Win32Utilsx64.dll

    These DLLs handle the connection between the native CIFS server and Alfresco.
- Finally, if you can connect to the Alfresco server but cannot authenticate your login details, check if you can use the same user name and password to login to Share.

Troubleshooting NTLM

This section provides help for diagnosing and resolving any issues that might arise when configuring NTLM.

Alfresco supports NTLM v2 protocol, which is more secure than NTLM v1 protocol. However, NTLM v2 cannot be used with pass-through authentication. You will have to switch to NTLM v1 if you want to use pass-through authentication, where Alfresco passes the log on request to an Active Directory or other server to validate the login credentials. For more information, please see the Configuring pass-through topic.

To authenticate using NTLM v1, set the following registry key on your client machines:

```
[HKLM\SYSTEM\CurrentControlSet\Control\Lsa]
 "LmCompatibilityLevel"=dword:00000001
```

Issue:

Failure of NTLM logon on machines running Windows 7 or Internet Explorer 8.

Troubleshooting

This problem is most likely caused by enhanced security in Windows 7, Vista and Windows 2008. Previous versions of Windows (XP) would fall back to NTLM v1, if NTLM v2 failed.

1. On Windows 7 clients, navigate to **Control Panel > Administrative Tools > Local Security Policy**.
2. In the left pane, navigate to **Security Settings > Local Policies > Security Options**.
3. In the right pane, find **Network Security: LAN Manager authentication level**.

   By default, the value of **Network Security: LAN Manager authentication level** is set to **Send NTLMv2 response only. Refuse LM & NTLM**.
4. Set the value of **Network Security: LAN Manager authentication level** to **Send LM and NTLM - use NTLMv2 session security if negotiated**.

This setting allows Windows 7 to use the more secure NTLM v2, if available, and fall back to NTLM v1 for Alfresco. If the machines are in a domain, it may be possible to change this setting on all of them via the group policy editor on the domain controller.

Troubleshooting WebDAV

Issue

Unable to connect to locally installed Alfresco server via WebDAV.

Troubleshooting

To troubleshoot this issue, perform the following steps, testing after each step to determine if the issue is resolved. If the issue is not resolved, continue to the next item in the list.

- Check if Alfresco has finished loading. Look for a *Server startup* message in the log file.

- Does the connection work if you use the IP address instead of the host name.
- Check if you can browse folders using `https://<alfresco_ip>/alfresco/webdav` in a web browser.
- Add your Alfresco server IP to the Trusted sites list in Windows Internet Explorer.
- Make sure the Webclient service is running. To do so, follow the steps below:
  1. Start `services.msc`.
  2. Start the Webclient service.

  For details on running the WebClient service, see Step 1 of Mapping an Alfresco space to a drive.
- Make sure to set the value of `BasicAuthLevel` as shown below:

  ```
  [HKLM\SYSTEM\CurrentControlSet\services\WebClient\Parameters]
   "BasicAuthLevel"=2
  ```

  For details on setting the Basic Authentication Level key in the Registry Editor, see Step 2 and Step 3 of Mapping an Alfresco space to a drive.
- Finally, if you can connect to the Alfresco server but cannot authenticate your login details, check if you can use the same user name and password to login to Share.

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
```

Also, to enable efficient inbound mail server logging in debug mode, you need a `log4j` option that allows you to track mails, including the sender details, recipient details, subject and the reason for rejection/acceptance. To do so, enable `DEBUG` logging for the class `org.subethamail.smtp.server.ConnectionHandler` as shown below:

```
log4j.logger.org.subethamail.smtp.server.ConnectionHandler=debug
```

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.

Troubleshooting IMAP

IMAP server error message

```
Exception in thread "Thread-53" java.lang.RuntimeException:
java.net.BindException: Cannot assign requested address:
JVM_Bind at com.icegreen.greenmail.imap.ImapServer.run(ImapServer.java:53)
Caused by: java.net.BindException:
Cannot assign requested address: JVM_Bind
```

This error message is related to the IP address or hostname that has been provided for binding. To resolve this issue:

- Check that the IP address or hostname you provided is correct for your `imap.server.host` setting.
- Check that the port you are using is not blocked. The default port to use is 143.
- Check that firewalls are not blocking this IP address or hostname.
- Use the command line tool Netstat to check your network connections.

  You should not use localhost as the imap.server.host - update this value with the IP address (or corresponding DNS address) of your external IP interface. A value of 0.0.0.0 in Unix will make it listen on the specified port on all IP interfaces.

Troubleshooting schema-related problems

This topic provides an introduction to the Schema Difference Tool.

The Schema Difference Tool provides a way of identifying and troubleshooting problems in Alfresco database schemas. Such problems can sometimes arise when performing certain version upgrades or customized installations.

Background

The Schema Difference Tool may be used when troubleshooting or examining the database schema for an Alfresco repository. The tool has two main functions:

1. Producing schema dumps as XML files.
2. Validating a database schema.

Schema dumps were available in previous versions of Alfresco. However, prior to the introduction of the Schema Difference Tool, the only way to judge the validity of the schema was to examine the file manually and compare schemas with simple text tools such as the Unix diff command. The Schema Difference Tool performs a certain amount of automatic comparison that removes much of the effort needed in making these comparisons.

If any changes are made to the database schema during server start-up (such as when installing Alfresco afresh) then the tool performs both schema dumping and validation as described below. The dumps and validation are made both pre-upgrade (that is before the schema changes) and post-upgrade.

Definition of terms used

The terms below are used throughout the rest of this document.

Database object
A schema, sequence, table, column, index, primary key or foreign key.

Reference schema
The definitive representation of an Alfresco repository schema for a given schema version on a vendor specific RDBMS. The reference schema is a model for what should be present in the database after installing or upgrading an Alfresco repository to a particular version. A reference schema is presented in the same XML format as a schema dump. For example a schema reference may be produced for MySQL on version 5025 of the Alfresco repository schema.

Target schema
The database schema that will be compared and validated with respect to a reference schema. For example, if installing an Alfresco repository from scratch, then the newly created schema will be a target schema for comparison against the appropriate reference schema.

Performing schema dumps

Schema dumps are XML representations of the database schema.

Schema dumps can take place in two situations:

1. The dump is triggered automatically on startup due to a difference being found between the reference and actual database schema.
2. The dump is manually triggered via a JMX client.

Each of these scenarios is described in the following sections.

Automatic dumps

Schema dumps are performed automatically on Alfresco server startup, if changes in database schema are detected.

Schema dumps are XML representations of the RDBMS schema. They should conform to the XSD: `http://www.alfresco.org/repo/db-schema/db-schema.xsd` The XSD file is embedded in the repository.

A schema dump is performed automatically during repository server startup if there were changes made to the database schema. The Alfresco log will indicate if any dumps were performed - entries such as these will be present:

```
2012-01-30 17:46:58,517  INFO  [domain.schema.SchemaBootstrap] [main]
 Normalized schema dumped to file
/tomcat/temp/Alfresco/Alfresco-schema-PostgreSQLDialect-pre-upgrade-
alf_-5548956643327704619.xml.
2012-01-30 17:46:58,518  INFO  [domain.schema.SchemaBootstrap] [main]
 Normalized schema dumped to file
/tomcat/temp/Alfresco/Alfresco-schema-PostgreSQLDialect-pre-upgrade-
avm_-2166257481854030130.xml.
2012-01-30 17:46:58,518  INFO  [domain.schema.SchemaBootstrap] [main]
 Normalized schema dumped to file
/tomcat/temp/Alfresco/Alfresco-schema-PostgreSQLDialect-pre-upgrade-
jbpm_-2230905975269998715.xml.
2012-01-30 17:46:58,519  INFO  [domain.schema.SchemaBootstrap] [main]
 Normalized schema dumped to file
/tomcat/temp/Alfresco/Alfresco-schema-PostgreSQLDialect-pre-upgrade-
act_-8103448407472298481.xml.
```

Similar entries for the post-upgrade files will also be present.

The legacy tool is still included and will create dumps of its own - the log messages look similar but should not be confused with the new format dumps.

Triggering dumps via JMX

Schema dumps can also be triggered manually via a JMX client.

In addition to automatic dumping, dumps can be manually invoked by use of the JMX interface.

This is an Enterprise only feature.

The JMX category **Alfresco**, **DatabaseInformation**, **SchemaExport** contains two operations:

1.
```
java.util.List dumpSchemaToXML()
```
2.
```
java.util.List dumpSchemaToXML(String prefixList)
```

The first operation takes no parameters and when invoked will create four dump files one for each prefix 'alf_', 'act_', 'jbpm_' and 'avm_'. The prefix means that only tables and sequences whose names begin with the prefix will be included in the dump. Related items, such as the indexes belonging to a particular table, will be dumped regardless of name.

The second variation takes a single String parameter and is a comma-separated list of prefixes that you wish to dump. If this operation were invoked with the parameter "alf_acl_, alf_node_" for example, then two files would be created (one for each prefix). The tables dumped in the first file would include `alf_acl_change_set` and `alf_acl_member`. Tables in the second file would include `alf_node_aspects` and `alf_node_assoc`. Neither file would include `alf_locale` or `alf_permission` since they do not carry one of the supplied prefixes.

Both of these calls will result in the log showing the location of the dumped files, but they also return a `List` of path names. JConsole will helpfully display these lists in a copy/paste friendly manner.

Performing schema validation

.

As for schema dumps, schema validation can happen either due to a schema change during repository startup, or triggered manually via JMX. Schema validation is performed in two steps:

1. Differencing
2. Validation

Each of these steps are described in the following sections.

Differencing

Differencing produces similar information to that obtained by using the Unix tool `diff` against a known 'good' reference schema dump and a potentially problematic target schema dump.

Differencing produces similar information to that obtained by using the Unix tool `diff` against a known 'good' reference schema dump and a potentially problematic target schema dump. However, since the tool is designed for performing a comparison between two database schemas, rather than arbitrary text, the output is more specific about the types of difference. The types of difference that can be reported are:

- A database object appears in both the reference and target schemas, but has differences in its properties. For example if an index appears in both schemas but has a different name.
- A database object appears in the reference schema but no corresponding object has been identified in the target database.
- A database object appears in the target schema but no corresponding object has been identified in the reference database.

One advantage of the Schema Differencing Tool differencing over traditional diff tool comparisons is that an index is not recognised by the exact text appearing in a dump. Instead it is identified by which table the index belongs to, which columns are indexed and in what order. If an index has the expected name and belongs to the correct table but has the wrong columns, or the correct columns in the wrong order, then differences will be reported. Or conversely, if the correct table has an index with the correct columns in the correct order, but has the wrong index name, then this will be reported. The name can be ignored during comparisons (useful for auto-generated index names) or can be taken into account. Part of the task of producing reference schema files is to specify this behaviour using `DbValidator` objects, which are explained in the following sections.

Index related example

Supposing we have the following index defined in the reference schema:

Index name	`permission_id`
Parent table	`alf_access_control_entry`
Columns	`permission_id`, `authority_id`, `allowed`, `applies`

This index is specified in the schema reference file in this way (parts omitted for brevity):

```
<table name="alf_access_control_entry">
  <!-- column definitions, primary keys and foreign keys ommitted -->
  <indexes>
```

```
    <index name="permission_id" unique="true">
      <columnnames>
        <columnname>permission_id</columnname>
        <columnname>authority_id</columnname>
        <columnname>allowed</columnname>
        <columnname>applies</columnname>
      </columnnames>
    </index>
    <!-- further index definitions ommitted -->
  </indexes>
</table>
```

When the target schema's index is compared against this reference then firstly a list of candidate matches are produced. There may be more than one matching index in the target schema, in which case a redundant database object warning is issued.

Candidate matches are produced dependent on object type. For indexes:

1. If the parent table is the same and the index name is the same, then it is considered the same index.
2. If the name is different but the parent table is the same and the columns indexed are the same, and in the same order, then it is is considered to be the same index.

Taking the first scenario for matching and using the `permission_id` index defined above, then if the `permission_id` index in the target database has the `allowed` and `applies` columns in the reverse order than is expected, the log file would notify us of validation problems:

```
2012-01-31 11:24:24,280  WARN  [domain.schema.SchemaBootstrap] [RMI TCP
 Connection(11)-10.244.50.71]
Schema validation found 2 potential problems, results written to:
/tomcat/temp/Alfresco/Alfresco-PostgreSQLDialect-Validation-
alf_-5903917616348258838.txt
```

The contents of the report file would look similar to the following:

```
Difference: expected
 index .alf_access_control_entry.permission_id.columnNames[2]="allowed",
but was .alf_access_control_entry.permission_id.columnNames[2]="applies"
Difference: expected
 index .alf_access_control_entry.permission_id.columnNames[3]="applies",
but was .alf_access_control_entry.permission_id.columnNames[3]="allowed"
```

Each line shows a problem with a particular database property. Here it indicates that the property at the path `.alf_access_control_entry.permission_id.columnNames[2]` has the value `applies` but according to the reference schema should be allowed. The leading dot of the path can be ignored (the schema name would be present before the leading dot in the case of Oracle for example), then there is the table name `alf_access_control_entry`, the index name `permission_id` within that, and a zero-indexed list property within that. The third item (index 2) is the property at fault: `columnNames[2]`.

Similarly, the next line indicates that the next item in the column name list, `columnNames[3]`, has the value `allowed` but was expected to be `applies`.

Validation

The Schema Difference Tool can use schema reference XML files to perform validation in addition to that performed by simple differencing.

Validation allows the application of more complex rules than whether there is a difference between two property values. Validation is performed by `DbValidator` objects. A chain of `DbValidator` objects is associated with each database object in the reference schema. Each of these is executed in turn and given the chance to create validation errors based on the corresponding object in the target schema.

If an index has not been given a specific name then the RDBMS will auto-generate one at creation time. This means that the reference schema cannot specify the exact name that the index in the target database will have. This would lead to schema differences being reported if it were not for the use of validators. A `NameValidator` may be specified for such an index:

```
<index name="SQL120116153558430" unique="true">
  <validators>
    <validator class="org.alfresco.util.schemacomp.validator.NameValidator">
      <properties>
        <property name="pattern">SQL[0-9]+</property>
      </properties>
    </validator>
  </validators>
  <columnnames>
    <columnname>ID</columnname>
  </columnnames>
</index>
```

This example is from a DB2 schema reference file `Schema-Reference-ALF.xml`) and indicates that although in the original reference schema the index was named `SQL120116153558430` any index having the appropriate parent table, column names (and column order) is valid as long as the name matches the regular expression `SQL[0-9]+`.

When the validator is invoked, it checks that the name property of the index matches the supplied regular expression. In addition to this, the validator reports, when configured to, that it takes responsibility for the name property of the index. This stops the Schema Difference Tool from applying the differencing logic to the property. A `DbValidator` can choose to apply its validation in addition to the differencing logic by not taking sole responsibility for any properties. Conversely a validator can also take sole responsibility for an entire database object in which case no differencing logic is applied to any part of the object.

A similar problem to the auto-generated name problem is when a database object is created automatically. DB2 creates indexes on the fly under certain circumstances. It is not known whether these indexes will exist at the time the Schema Difference Tool will be run. Furthermore, the indexes are an implementation detail for DB2 rather than an explicit declaration on how the Alfresco schema should appear. To suppress such errors an `IgnoreObjectValidator` may be used - it takes responsibility for validation of the associated database object, but performs no actual validation.

Another index related example

Supposing an index is expected to be auto-generated and is defined in the schema reference file as:

```
<index name="SQL120116153558430" unique="true">
  <validators>
    <validator class="org.alfresco.util.schemacomp.validator.NameValidator">
      <properties>
        <property name="pattern">SQL[0-9]+</property>
      </properties>
    </validator>
  </validators>
  <columnnames>
    <columnname>ID</columnname>
  </columnnames>
```

```
</index>
```

Perhaps a specific unsupported upgrade path has introduced an unexpected schema change - it may not be a problem, but it is important that differences are highlighted so that a decision can be made on whether the difference represents a problem and whether a fix will need to be made. On running the Schema Difference Tool, the following might be observed in the log files:

```
2012-01-31 14:28:50,697  WARN  [domain.schema.SchemaBootstrap] [main] Schema
 validation found 1 potential problems, results written to:
/tomcat/temp/Alfresco/Alfresco-DB2Dialect-Validation-Post-Upgrade-
alf_-4048062354335481885.txt
2012-01-31 14:28:51,440  INFO  [domain.schema.SchemaBootstrap] [main] Compared
 database schema with reference schema (all OK):
class path resource [alfresco/dbscripts/create/
org.hibernate.dialect.DB2Dialect/Schema-Reference-AVM.xml]
2012-01-31 14:28:53,326  INFO  [domain.schema.SchemaBootstrap] [main] Compared
 database schema with reference schema (all OK):
class path resource [alfresco/dbscripts/create/
org.hibernate.dialect.DB2Dialect/Schema-Reference-JBPM.xml]
2012-01-31 14:28:54,682  INFO  [domain.schema.SchemaBootstrap] [main] Compared
 database schema with reference schema (all OK):
class path resource [alfresco/dbscripts/create/
org.hibernate.dialect.DB2Dialect/Schema-Reference-ACT.xml]
```

The AVM, JBPM and ACT database objects are all as expected, but there is a difference between the target schema and the ALF (alf_ prefixed database objects) schema reference. Looking at that file it can be seen that an index that is expected to have been auto-generated has been created with an explicit name:

```
Validation: index
 ALFUSER.ALF_ACCESS_CONTROL_ENTRY.SQL120131142718040.name="idx_alf_ace_auth"
 fails to match rule: name must match pattern 'SQL[0-9]+'
```

Specifically, the error report is stating that the index defined in the schema reference having the name `SQL120131142718040` belonging to the table `ALF_ACCESS_CONTROL_ENTRY` is expected to be named in the same way: prefixed with SQL then a string of one or more digits.

Triggering validation by JMX

In addition to automatic validation, validation can be manually invoked by use of the JMX interface.

Please note: this is an enterprise only feature.

The JMX category **Alfresco**, **DatabaseInformation**, **SchemaValidator** contains one operation:

```
            void validateSchema()
```

The operation takes no parameters and returns nothing. However, if the operation is invoked then validation will be performed and the Alfresco log will show the results:

```
2012-01-31 14:51:46,770  INFO  [domain.schema.SchemaBootstrap] [RMI TCP
 Connection(13)-10.244.50.71] Compared database schema
with reference schema (all OK): class path resource
[alfresco/dbscripts/create/org.hibernate.dialect.PostgreSQLDialect/Schema-
Reference-ALF.xml]
2012-01-31 14:51:47,360  INFO  [domain.schema.SchemaBootstrap] [RMI TCP
 Connection(13)-10.244.50.71] Compared database schema
with reference schema (all OK): class path resource
[alfresco/dbscripts/create/org.hibernate.dialect.PostgreSQLDialect/Schema-
Reference-AVM.xml]
```

```
2012-01-31 14:51:49,847  INFO  [domain.schema.SchemaBootstrap] [RMI TCP
 Connection(13)-10.244.50.71] Compared database schema
with reference schema (all OK): class path resource
[alfresco/dbscripts/create/org.hibernate.dialect.PostgreSQLDialect/Schema-
Reference-JBPM.xml]
2012-01-31 14:51:50,910  INFO  [domain.schema.SchemaBootstrap] [RMI TCP
 Connection(13)-10.244.50.71] Compared database schema
with reference schema (all OK): class path resource
[alfresco/dbscripts/create/org.hibernate.dialect.PostgreSQLDialect/Schema-
Reference-ACT.xml]
```

In the example above there were no problems found in the target schema.

Reference

Properties available 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`
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 while 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=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.

The Alfresco Full Text Search (FTS) query text can be used standalone or it can be embedded in CMIS-SQL using the `contains()` predicate function. The CMIS specification supports a subset of Alfresco FTS. The full power of Alfresco FTS can not be used and, at the same time, maintain portability between CMIS repositories.

Alfresco FTS is exposed directly by the interface, which adds its own template, and is also used as its default field. The default template is:

```
%(cm:name cm:title cm:description ia:whatEvent ia:descriptionEvent lnk:title
 lnk:description TEXT)
```

When Alfresco FTS is embedded in CMIS-SQL, only the CMIS-SQL-style property identifiers (`cmis:name`) and aliases, CMIS-SQL column aliases, and the special fields listed can be used to identify fields. The SQL query defines tables and table aliases after `from` and `join` clauses. If the SQL query references more than one table, the `contains()` function must specify a single table to use by its alias. All properties in the embedded FTS query are added to this table and all column aliases used in the FTS query must refer to the same table. For a single table, the table alias is not required as part of the `contains()` function.

When Alfresco FTS is used standalone, fields can also be identified using `prefix:local-name` and `{uri}local-name` styles.

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.

```
big AND yellow AND banana
TEXT:big and TEXT:yellow and TEXT:banana
```

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.

If not otherwise specified, by default search fragments will be ORed together.

```
big yellow banana
big OR yellow OR banana
```

```
TEXT:big TEXT:yellow TEXT:banana
TEXT:big OR TEXT:yellow OR TEXT:banana
```

These queries search for nodes that contain the terms "big", "yellow", or "banana" in any content.

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

Description	Type	Example
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
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"
"?????"
```

When performing a search that includes a wildcard character, it is best to wrap your search term in double quotation marks. This ensures all metadata and content are searched.

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
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

Search query syntax APIs

The following show how to embed queries in CMIS.

Embedded in CMIS contains()

```
- strict queries
SELECT * FROM Document WHERE CONTAINS('\'zebra\)
SELECT * FROM Document WHERE CONTAINS('\'quick\)

- Alfrecso extensions
SELECT * FROM Document D WHERE CONTAINS(D, 'cmis:name:\'Tutorial\)
SELECT cmis:name as BOO FROM Document D WHERE CONTAINS('BOO:\'Tutorial\)
```

Search Service

```
ResultSet results = searchService.query(storeRef,
 SearchService.LANGUAGE_FTS_ALFRESCO, "quick");
```

```
SearchService.LANGUAGE_FTS_ALFRESCO = "fts-alfresco"
```

Node Browser

Alfresco FTS is supported in the node browser.

JavaScript

```
search
{
   query: string,          mandatory, in appropriate format and encoded for the
 given language
   store: string,          optional, defaults to 'workspace://SpacesStore'
   language: string,       optional, one of: lucene, xpath, jcr-xpath, fts-
alfresco - defaults to 'lucene'
   templates: [],          optional, Array of query language template objects
 (see below) - if supported by the language
   sort: [],               optional, Array of sort column objects (see below) -
 if supported by the language
   page: object,           optional, paging information object (see below) - if
 supported by the language
   namespace: string,      optional, the default namespace for properties
   defaultField: string,   optional, the default field for query elements when
 not explicit in the query
   onerror: string         optional, result on error - one of: exception, no-
results - defaults to 'exception'
}

sort
{
   column: string,         mandatory, sort column in appropriate format for the
 language
   ascending: boolean      optional, defaults to false
}

page
{
   maxItems: int,          optional, max number of items to return in result
 set
   skipCount: int          optional, number of items to skip over before
 returning results
}

template
{
   field: string,          mandatory, custom field name for the template
   template: string        mandatory, query template replacement for the
 template
}
```

For example:

```
 var def =
  {
     query: "cm:name:test*",
     language: "fts-alfresco"
  };
  var results = search.query(def);
```

Templates

Alfresco FTS is not supported in FreeMarker.

Search query templates

The FTS query language supports query templates. These are intended to help when building application specific searches.

A template is a query but with additional support to specify template substitution.

%field

Insert the parse tree for the current `ftstest` and replace all references to fields in the current parse tree with the supplied field.

%(field1, field2)
%(field1 field2)

(The comma is optional.) Create a disjunction, and for each field, add the parse tree for the current `ftstest` to the disjunction, and then replace all references to fields in the current parse tree with the current field from the list.

Name	Template	Example Query	Expanded Query
t1	%cm:name	t1:n1	cm:name:n1
t1	%cm:name	t1:"n1"	cm:name:"n1"
t1	%cm:name	~t1:n1^4	~cm:name:n1^4
t2	%(cm:name, cm:title)	t2:"woof"	(cm:name:"woof" OR cm:title:"woof")
t2	%(cm:name, cm:title)	~t2:woof^4	(~cm:name:woof OR ~cm:title:woof)^4
t3	%cm:name AND my:boolean:true	t3:banana	(cm:name:banana AND my:boolean:true)

Templates may refer to other templates.

```
nameAndTitle -> %(cm:name, cm:title)
nameAndTitleAndDesciption -> %(nameAndTitle, cm:description)
```

Search query literals

Everything is really a term or a phrase. The string representation you type in will be transformed to the appropriate type for each property when executing the query. For convenience, there are numeric literals but string literals may also be used.

String literals for phrases may be enclosed in double quotes or single quotes. Java single character and \uXXXX based escaping are supported within these literals.

Integer and decimal literals conform to the Java definitions.

Dates as any other literal can be expressed as a term or phrase. Dates are in the format ...... Any or all of the time may be truncated. All of the date must be present.

The date type also supports NOW and may be extended in the future to support some date math - akin to what is used by SOLR.

In range queries, strings, term, and phrases that do not parse to valid type instance for the property are treated as open ended.

```
test:integer[ 0 TO MAX]
- matches anything positive
```

Date ranges do not currently respect the truncated resolution that may be presented in range queries.

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

selectmany.ftl

The `selectmany` control renders a standard HTML select list allowing multiple selections.

The following parameters are available:

- `options` (mandatory, comma separated string): A comma separated list of options to display, for example "First,Second,Third". If a value for an option also needs to be specified the "First|1,Second|2,Third|3" format can be used.
- `size` (optional, int): The size of the list i.e. how many options are always visible, the default is 5.
- `styleClass` (optional, string): Allows a custom CSS class to be applied to the select list
- `style` (optional, string): Allows CSS rules to applied directly to the select list
- `forceEditable` (optional, boolean): Forces the control to be editable, default is false

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 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.

Frequently occurring tasks

This section describes tasks that are frequently used or referred to in this guide.

Adding folder paths to the Windows path variable

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.

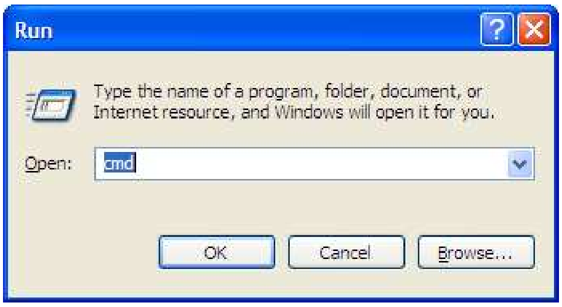
   For example: `path C:\jdk`
8. Click **OK** in each open window.

   The new path will be used the next time a command prompt is opened, or a service is started.

Opening a Windows command prompt

1. On the Windows task bar, click **Start > Run**.

2. In the **Run** dialog box, type `cmd`.

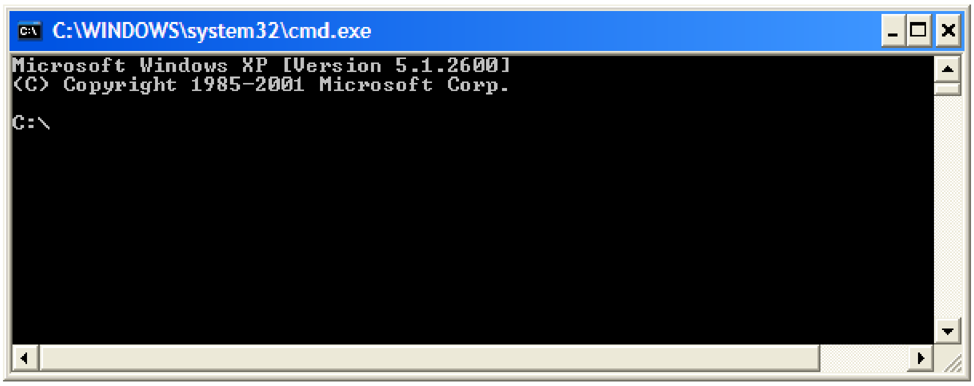


Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: cmd

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:\
```

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 `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
```

Administrator best practices

This section provides best practice guidelines for Alfresco administrators.

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>` and `<web-extensions>` directories, 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 the Alfresco Support Portal, a member-only (Enterprise subscription) benefit that provides a customer portal, downloads, 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 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 Search 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 the Customizing section in Alfresco documentation.
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 contains settings for the location of the content and index data, the database connection properties, the location of third-party software, and database driver 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.

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

Alfresco Enterprise 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/`

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).

Copyright

The United States export control laws and regulations, including the Export Administration Regulations of the U.S. Department of Commerce, and other applicable laws and regulations apply to this documentation which prohibit the export or re-export of content, products, services, and technology to certain countries and persons. You agree to comply with all export laws, regulations, and restrictions of the United States and any foreign agency or authority and assume sole responsibility for any such unauthorized exportation.

You may not use this documentation if you are a competitor of Alfresco, except with Alfresco's prior written consent. In addition, you may not use the documentation for purposes of evaluating its functionality or for any other competitive purposes.

If you need technical support for this product, contact Customer Support by email at support@alfresco.com. If you have comments or suggestions about this documentation, contact us at documentation@alfresco.com.

This copyright applies to the current version of the licensed program.