Alfresco Training

Version 1.0

Foundation for Developers

Contents

VM Instructions

This is an interactive course where the exercises have been designed to augment what you learn in class and re-enforce your knowledge through progressively more complex labs. A lab environment is provided to you in the form of a virtual machine based on VMWare#'s VMWare Server product. In order to run these labs you must install any of these products on your laptop:

- **Windows:** VMWare Server
- **Windows:** VMWare Player
- **Linux:** VMWare Server for Linux Operating Systems
- **Mac:** VMWare Fusion

VMware Server and Player for Windows and Linux is free, although you will need to register with VMware, VMware Fusion for the Mac is however a purchasable product which can be purchased for a small fee. If you have not used this product before their is a trial version you can download which will allow you to use the product for 30 days.

You can find the latest version for your operating system from VM Wares website:`http://www.vmware.com`

This virtual machine may also be run under other virtualization products such as Virtual Box (**www.virtualbox.com**) or Parallels(**www.parallels.com**) however the machine has not been tested in these environments and your instructor may not be able to help you with any technical issues.

System Requirements

In order to run the VMWare machine your laptop (called the host by VMWare) needs to have some spare capacity. Below we list the machine requirements for different host operating systems.

Windows

- Standard x86 compatible or x86 64 compatible processor with a minimum of 2 cores. 1Ghz or faster CPU minimum.
- Minimum 2GB of RAM (4GB recommended).
- 1.7GB free disk space for VMWare Server installation and at least 8GVB for the Alfresco virtual machine.

Macintosh

- Minimum 1GB of RAM (2GB RAM recommended).
- 700MB free disk space for VMWare Fusion and at least 8GB for the Alfresco virtual machine.
- Mac OS X 10.5.8 or later; Mac OS X 10.6 or later.

Linux

The system requirements for Linux are the same as for a Windows laptop. The currently supported host operating systems are listed below, for a full list including version numbers refer to the VMWare website:

- Red Hat Enterprise Linux
- Mandrake Linux
- Mandriva Corporate Server
- SUSE Linux Enterprise Server

- TurboLinux Enterprise Server
- Ubuntu Linux

The specifications and supported operating systems listed here are current as of this documents origination date and is subject to change. For current specifications please visit the VMWare website.

Downloading and installing VMWare

Visit the VMWare website and download the appropriate version of VMWare. If you have arrived at the course without VMWare pre-installed then your instructor may have the software which you can copy rather than having to download.

Virtual machine details

The virtual machine provided for this course has the following characteristics:

Guest Operating System	Ubuntu 10.04 LTS Desktop (32 bit)
Alfresco	4.x Enterprise Edition running PostgreSQL
Machine name and IP Address	Alfresco, IP through DHCP
CPU and Memory	1 CPU, 1GB (you may decrease this to 768MB minimallay, but this will impact the performance of your virtual machine)
Disk Footprint	8 GB
Login details	• **Into virtual machine:** alfresco:alfresco • **Into Alfresco:** admin:admin
Other tools installed	Depending upon your particular course, there will be additional tools installed on the VM for your use. These will be covered during the course.

This machine is intended to be used to support this course and is not intended for production use.

Accessing the virtual machine

In order to access the virtual machine you will need to open the virtual machine. How you do this will depend upon which VMWare product you are running.

VMWare Server	1. Copy the virtual machine to the *standard* virtual machine location. This will most likely be **c:\Virtual Machines**. 2. Add the virtual machine to the *inventory*. 3. Start the virtual machine and answer the virtual machine re-configuration question. *The machine was copied*. 4. Change the name of the virtual machine.
VMWare Player and VMWare Fusion	With either of these products simply open the virtual machine from the application. Navigate to the location where you put the virtual machine and once opened you can start (play) the virtual machine.

Copy the virtual machine

In order for your system to be able to start the virtual machine you need to copy the machine to the standard location which VMWare expects. This varies from system to system, on Windows it is usually `C:\Virtual Machines`. On Mac it is `Documents > Virtual Machines`.

Add the virtual machine to the inventory

Now we need to add the virtual machine to the inventory so that it can be started and used.

Windows

From within the VMWare console (this is accessed through your browser), you should select `Virtual Machine -> Add Virtual Machine to Inventory`. Then navigate to the standard dictionary and contents and add, finally select `<machine_Name>.vmx` and click **OK**.

Mac

When you are running VMWare Fusion navigate to `File -> Open` and open the virtual machine, this will add the machine to the inventory.

Accessing the virtual machine

In order to access the virtual machine from your laptop or Mac you will need to use either the hostname (alfresco) of the virtual machine or its IP address. To use the hostname for accessing the virtual machine you will need to change your hosts file. This can be protracted and so for this course we recommend that you simply use the IP address. Assuming that you make no changes to the virtual machine it will be assigned an address by the VMWare DHCP server. You should confirm the IP address before using it, see below.

Start your virtual machine

This is done through VMWare web access or the VMWare Fusion application.

On Windows you will need to authenticate again to use VMWare. The username and password required is your Windows username and password combination used to login to Windows #– which may include a domain name.

Once your virtual machine is started, open the console and login as the user alfresco. When starting the virtual machine you may need to wait 45-60 seconds before all the necessary system services are running before you will be able to connect.

Confirming the IP address

Once you have logged into the virtual machine as Alfresco navigate to `Application -> Accessories -> Terminal`, from the menu bar. This opens up a terminal window.

1. Input the following command and hit Enter: `ifconfig`
2. This will show you the status of your networking including the IP address, you will see two lines as follows, you are interested in the second line containing the IP address such as shown here:

```
Link encap:Ethernet HWaddr 00:0c:29:fb:d9:dd
Inet addr:192.182.172.60 Bcast:192.168.78.255
Mask:255.255.255.0
```

3. You may use this address to access your virtual machine from your desktop, make a note of this.

Accessing the Alfresco server from your browser

Inside the Virtual Machine open your browser and point it to:

```
http://localhost:8080/share
```

You may keep the virtual machine afterwards; the license is valid for the installed Alfresco version and does not expire but is limited to a maximum of 10 users.

Exercises and Labs

Each course contains multiple exercises and/or labs to help reinforce the learning process. The directions for these are included in the student guide, however, in the event of any changes to the directions in the student guide your instructor will advise you of the correct procedure to follow.

Sample documents

You will be provided with a sample documents folder which is located on the VM desktop and contains a number of files that will be used through the exercises.

Solutions

Where appropriate solutions are provided, we suggest that you work through the exercises before turning to the solutions, however the solutions are written such that if you are having difficulties with one particular exercise you can turn to the appropriate solution and you won't be held back. Your instructor will advise you of where the solution files are located.

Introducing Alfresco

Why Alfresco

In this introductory Element we examine why a system, such as Alfresco is required, and explore why you would use Alfresco as opposed to other content management systems or technologies. We'll explore the different types of users and what kind of functionality they can look forward to using. Enterprise Content Management is defined and explored. Finally we will look at the different editions of Alfresco and how the product is distributed and supported.

Alfresco is an Open Source Enterprise Content Management system that enables an organization to manage large volumes of content. The Alfresco system offers Document Management, Records Management, Web Content Services, rich collaboration between users and groups, powerful workflow and business process management.

The different Alfresco versions enable both small and large organizations to utilize the product. Alfresco Enterprise is typically deployed to medium and large organizations. Community is often used by those wishing to experiment and explore the functionality of Alfresco or use it within a non-production environment.

Alfresco in the cloud extends the product range, enabling individuals and organizations to use the powerful Enterprise Content Management system in a fully hosted and managed environment.

Alfresco offers content services such as metadata management, version control, workflow, search, renditions, aspects and many others give content a life and visibility which enables users to maximize its use. End users may experience Alfresco in distinct ways as they access and work with content. The product offers a robust set of infrastructure services enabling content to be accessed through an extensive range of protocols, products and user interfaces.

End users

At the simplest level Alfresco can present itself as a shared drive and users may not even realize they are using Alfresco as they create, edit and save documents. In addition to network drive based access protocols, web based access is achieved through Alfresco Share. Alongside this, direct access to the Alfresco content store can be achieved using productivity applications through the SharePoint protocol. Applications may also access Alfresco through WebDav, CMIS, FTP, CIFS and other protocols.

Alfresco offers integration with Google Docs, enabling content of specific formats to be checked out of Alfresco, edited within that environment and saved back to Alfresco.

Social publishing

The landscape for the use and publishing of content has changed dramatically in recent years with social media becoming mainstream. Issues of control and management can arise for organizations due to the sheer ease-of-use and accessibility of these services. Alfresco has the ability to publish content directly to social media services. This introduces a range of benefits such as; centrally managed login credentials, establishing workflow, publishing history, linking posts and status updates to content.

Risk is reduced and the value of using social media sites can be fully realized. Within social networks content comes to life and visibility through trending, being liked and commented upon. Within Alfresco it is possible to follow individuals and the content they create, like content and post comments. Alongside many criteria it is possible to undertake search based upon popularity. Activity streams can also be viewed and filtered.

Alfresco mobile

The Open Source, iPhone and iPad application brings the power of Alfresco to mobile devices; enabling an Alfresco content repository to be accessed from anywhere at anytime.

The application takes the user beyond a simple browsing experience which you might anticipate when using a mobile device. Functionality such as downloading content, local editing and uploading back to the repository is offered, alongside other powerful features which are found in Alfresco Share.

For business

Web Content Services manage the lifecycle of content, from production to publishing. Publishing through an Enterprise Content Management system provides a host of advantages over traditional publishing; such as transformation services, translation support, version control, workflow processes (such as review and approval), auditing and management of the process. Alongside the Web Quick Start functionality Alfresco offers integrations which partner content management with a range of technologies and products. Integrations with SAP, jive, Drupal and Liferay exist. In its application to business Alfresco is a broad solution which is not constrained to a particular vertical industry. Unique components can be deployed to satisfy business requirements.

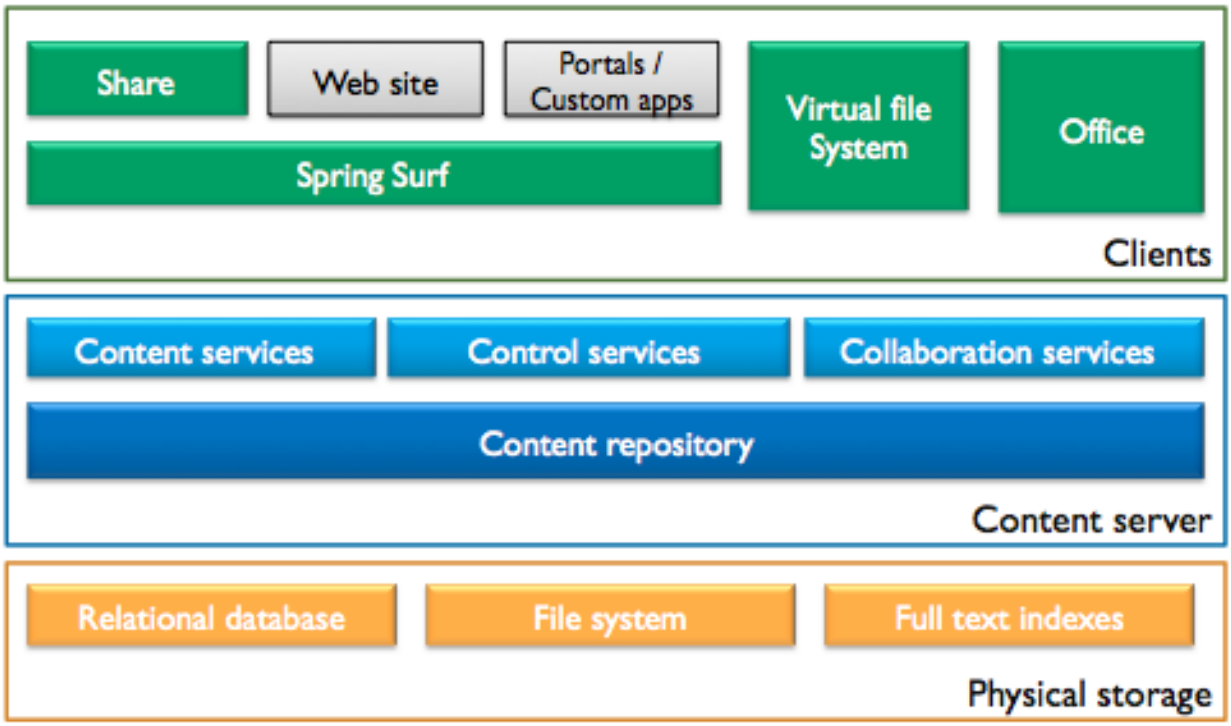
Alfresco is designed to scale (according to demand) and fit the needs of a range of business sizes. Alfresco supports the business through the implementation of business process management and workflow through the Activiti platform.

The requirement of governance, risk management and compliance is satisfied through Alfresco's DoD 5015.02 certified Records Management module.

For the developer

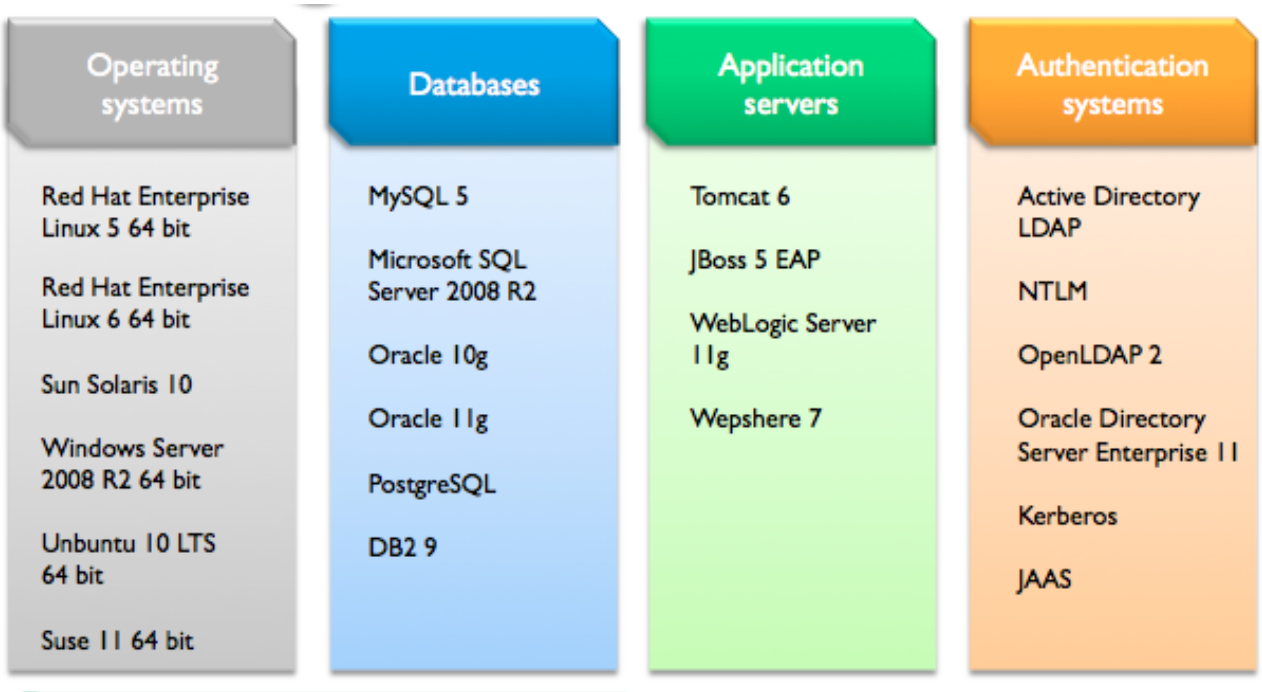
For the developer Alfresco offers a fully featured content management platform. This simplifies the design and implementation of content management applications and bespoke customization of the system.

Alfresco is built upon proven and robust industry standards such as; CMIS, REST, SOAP, Java and web based services.



For IT services

For the IT services department, Alfresco represents a high-value, lower cost alternative to closed proprietary systems. Alfresco supports a wide variety of platforms and technologies. It is written in 100% Java making it extremely portable and able to be fit into existing infrastructures.



The Alfresco architecture lends itself to working within a virtualized environment. Alfresco is a product that offers impressive performance and can be implemented as a high-availability system.

Alfresco and open source

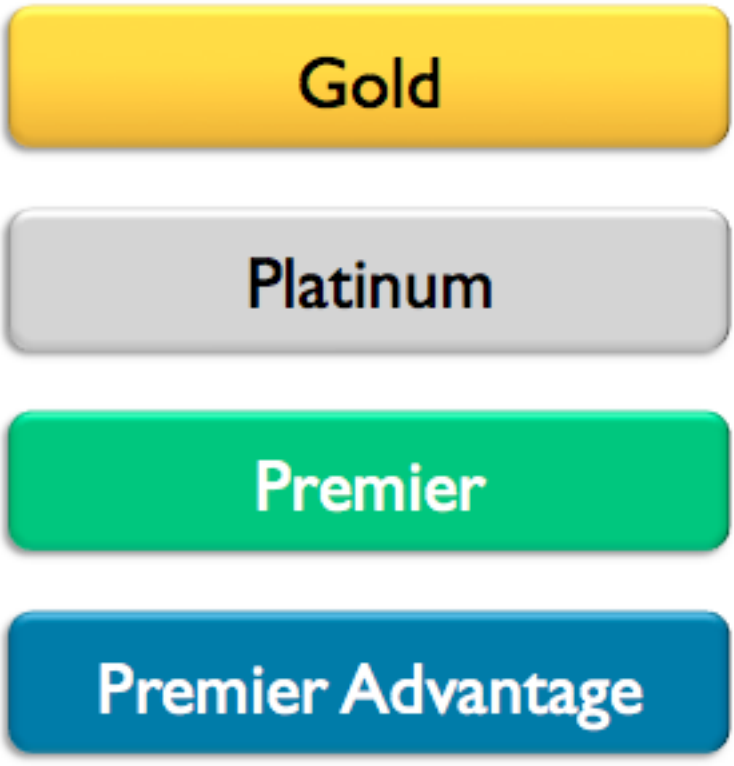
Alfresco is Open Source making it transparent and fully extensible. Combining industry standards with an Open Source development model makes Alfresco significantly easier to integrate into business and technology environments. The strategy also enables Alfresco to offers its products at no licensing cost. Faster development time and wider innovation is possible since other Open Source technologies and solutions can be integrated into Alfresco. The Open Source community is diverse, vibrant and highly productive. Transparency of code enables peer review, which combined with Alfresco's extensive testing program leads to an extremely reliable and robust solution.

Alfresco provides a full range of services, support, training, consulting and documentation. Since Alfresco does not charge for the software products it creates, the company is sustained and grown by providing a chargeable service infrastructure. This includes high quality follow the sun support, training and certification, consulting and a partner program. Despite being a commercial operation Alfresco still strongly contributes to the Open Source community. Examples include new platforms such as Spring and Surf and new projects such as Activiti.

Alfresco editions

Alfresco Community and Enterprise are classed as 'on-premise' products, where an organization installs, configures and maintains the installation.

Alfresco in the cloud is aimed at those wishing to utilize a fully managed and hosted Alfresco implementation. Let's explore the on-premise offerings in further detail.



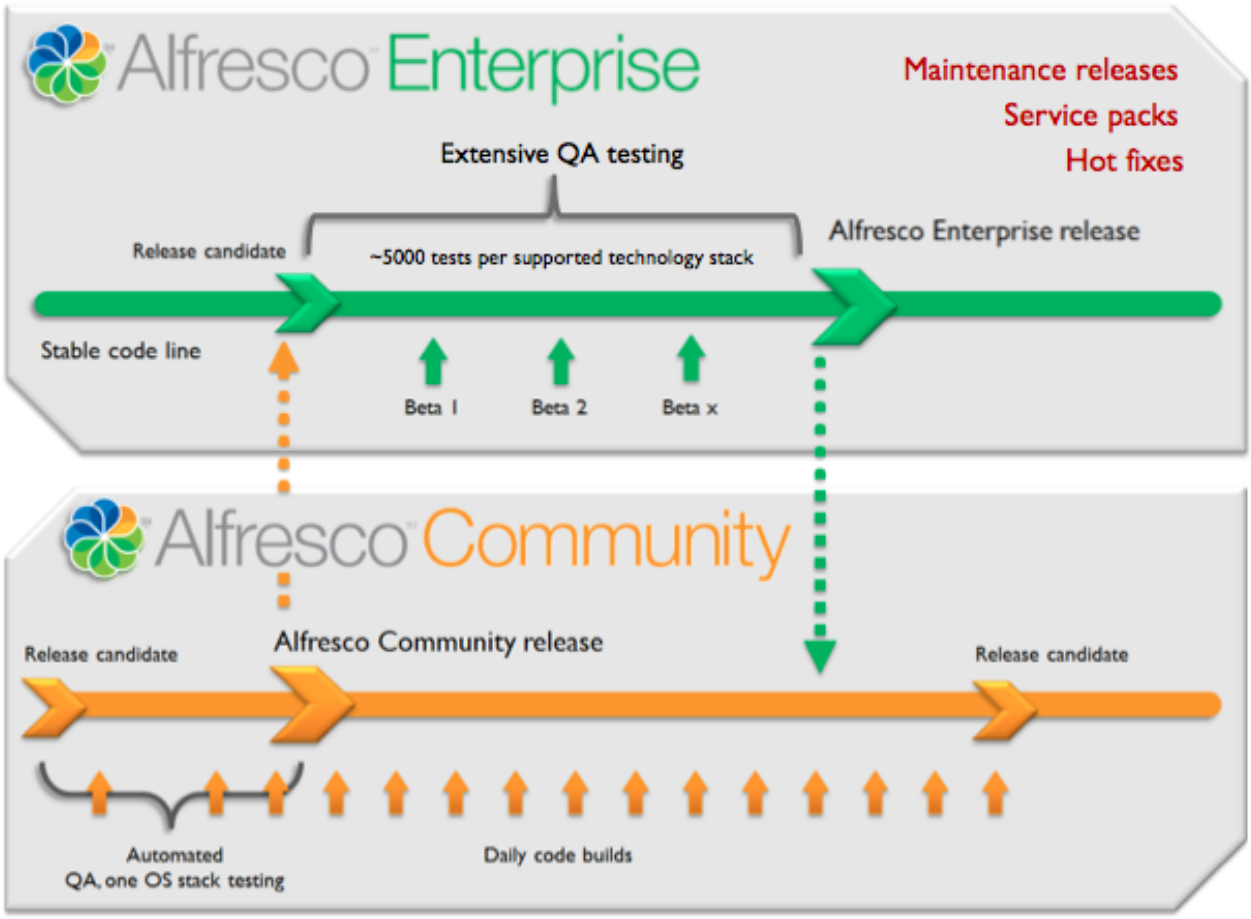
Software releases

With Community you have access to the very latest technology from Alfresco, but expect it to be bleeding edge, there are frequent releases and you have access to the source code through Subversion. Community edition has nightly builds with intermittent major releases.

With the Enterprise edition you will find that there are more releases but no nightly builds, additionally each Enterprise release will go through a full quality assurance cycle.

Development lifecycle

Alfresco Enterprise is a unique code base from Alfresco Community and is a certified and warrantied binary. It has a differentiated development cycle and undergoes a rigorous quality assurance process to ensure stability, scalability and security. The quality assurance process initiates around five thousand tests for each and every supported technology stack.



Audience

Community is recommended for competent technical enthusiasts who are using the software in a non-critical environment. Community is also a good platform for Enterprise users who want to explore upcoming features.

If you are organization or government looking for a robust production quality solution to deploy mission critical applications we recommend that you use the Enterprise edition.

Support

Alfresco Community offers support through the community, this is served by forums and a wiki. With the Enterprise edition you get a service level agreement (SLA) which matches your needs. This SLA provides you with access to Alfresco technical support team and allows you to log calls for product support and upgrades and track and manage issue escalation.

In order to ensure that Alfresco Enterprise is as robust as it can be, a range of tests are run, which include functionality and regression tests and a full set of scalability and high-availability tests. Even with a robust testing scheme some bugs are inevitable. When you log a bug with the support team we track and fix bug those bugs and periodically produce service packs which are quality tested and address those issues.

Licensing

With either Community or Enterprise there is no licence charge, however Community offers support only through the forums whereas Enterprise provides worldwide multi-level support from a team of experts which can be purchased as a yearly subscription service.

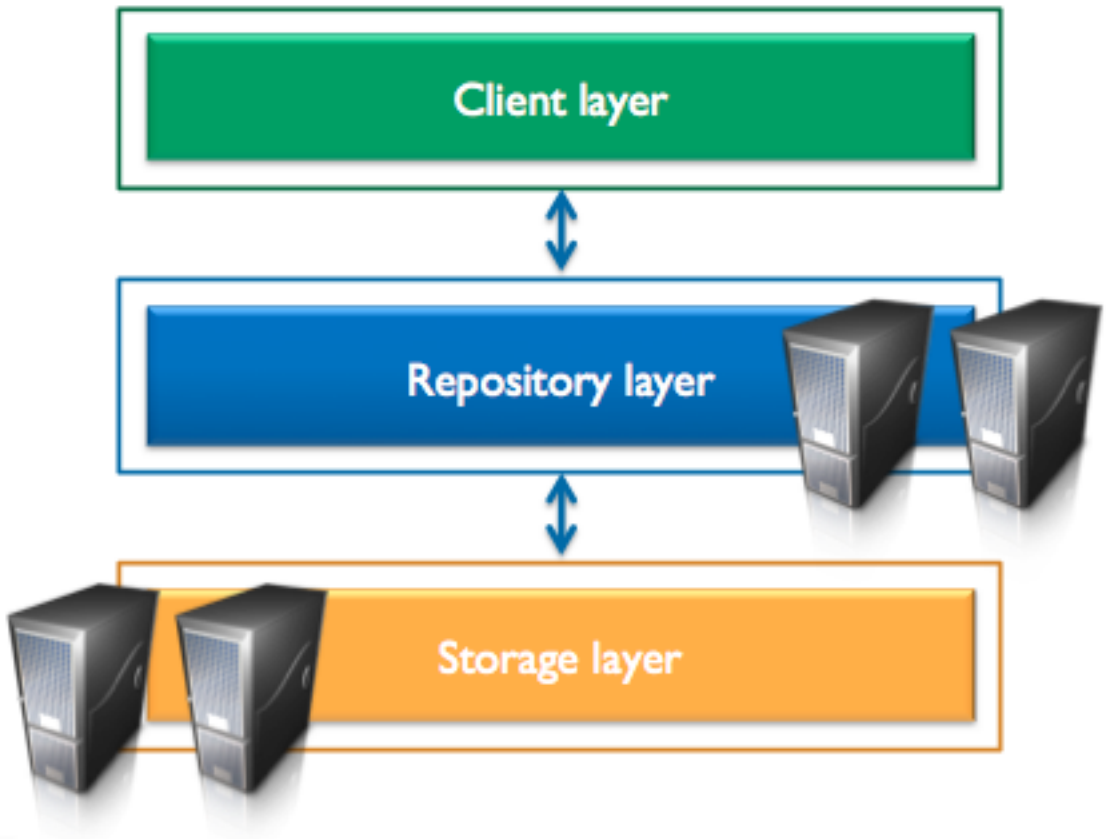
Enterprise service level agreements

Enterprise customers have a choice of four contractual service level agreements, Gold, Platinum, Premier and Premier Advantage providing tailored levels of support all the way to follow-the-sun support.

Enterprise only tools

Finally with the Enterprise edition there are a number of tools, which don't exist in Community, that are directly designed to help you administrate an Alfresco production system and provide significant value for an administrator. For example Enterprise provides Java Management Extensions which can support a variety of JMX consoles. This helps you monitor and run a production environment allowing you to plug Alfresco into your existing monitoring tools such as Hyperic and Tivoli.

Alfresco Community edition cannot be clustered, the Enterprise edition provides for fast and easy clustering for system administrators.



The Enterprise edition also introduces Kofax integration. With this you can front end the Alfresco ECM with the popular Kofax document capture and scanning product. Finally Enterprise edition provides the system administrator with the re-assurance that non-open source databases such Oracle are supported out of the box.

What Alfresco is used for

Used by diverse industries there exists a wide application of the Alfresco solution. Typical uses include;

- Document management.
- Collaboration for content development.
- Shared drive replacement.
- Portals and intranets.
- Web Content Management.
- Information publication.
- Records Managements.
- Case Management.
- Electronic post room.
- Image scanning and management.
- Invoicing.
- Knowledgebases.

Case studies

A growing range of companies and organisations use Alfresco.

Activision is using Alfresco to manage content for a large range of customer web portals, centered on video game franchises. Implemented in six different languages they deliver a personalized one-to-one web experience for 5 million video gamers.

Yellow pages implemented Alfresco to improve the speed, performance and user experience of its business search service, supporting over 20 million customer searches a month.

The City of Denver consolidated 14 separate document management systems, including EMC Documentum and Microsoft SharePoint to Alfresco, reducing complexity, IT costs and increasing service to the public.

KLM implemented Alfresco to meet Document Management needs of 30,000 employees and achieved a full roll out in just 6 months.

North-West University in South Africa were able to leverage Alfresco's open and flexible architecture to overcome local telecommunications constraints as they consolidated their Enterprise Content Management system through Alfresco.

Further information and detailed case studies can be found at alfresco.com/customers.

Summary

In this Alfresco Element, Enterprise Content Management has been explored.

The benefits of using Alfresco, its functionality and the relationship with Open Source has been discussed.

The different editions of Alfresco have been examined.

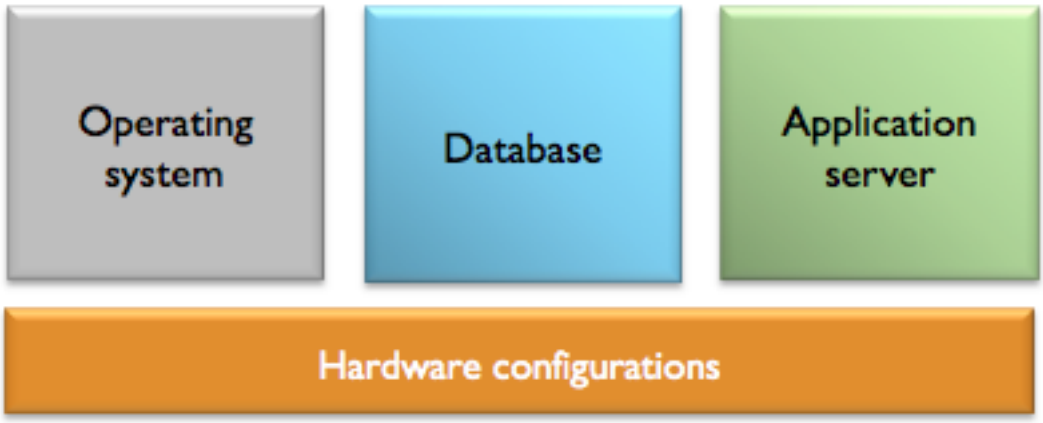
This concludes the introductory Alfresco Element.

Architecture and Technology

Introduction

In this Alfresco Element the architecture and technology of the Alfresco Enterprise Content Management system is explored.

The decisions which can and should be made regarding how to architect your installation are examined. The start-up process is reviewed, with particular reference to the order in which the components of the system initiate. The choice of operating system, database, application server and hardware configurations, which can be used for an Alfresco installation are examined.



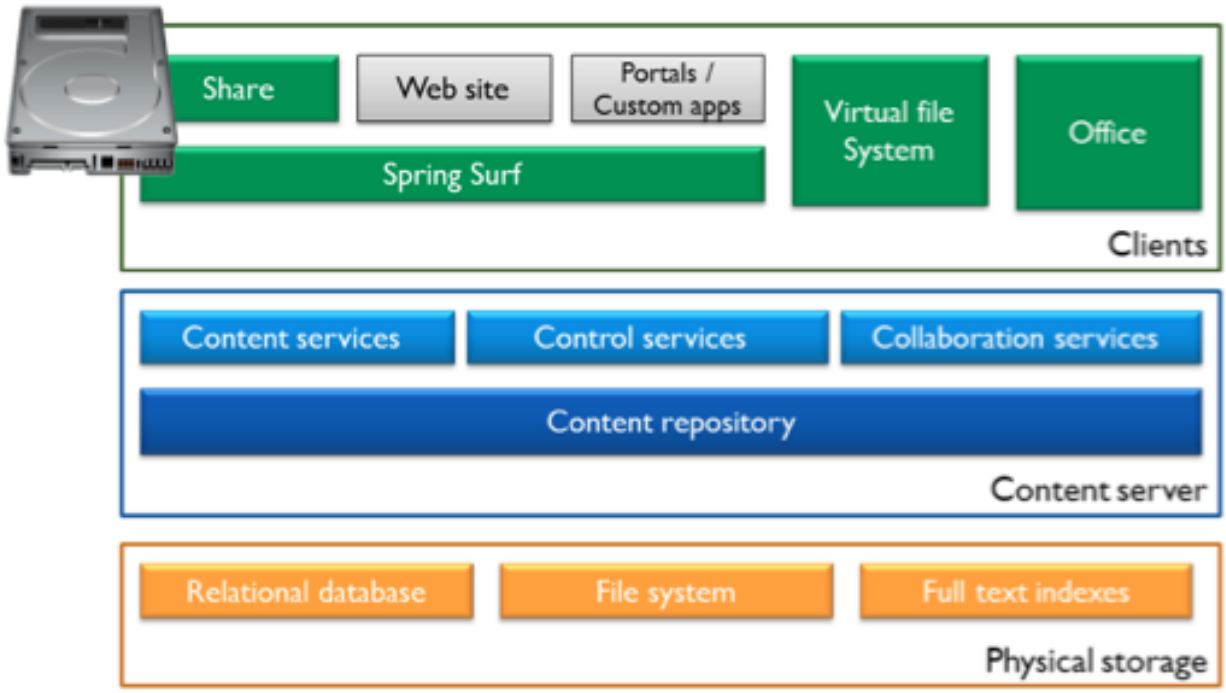
Architecture

Enterprise Content Management covers a broad range of applications. The Alfresco architecture is designed to support the requirements of these applications, resulting in a coherent solution where content and management processes are not forced into silos.

Alfresco recognizes that each discipline has unique but overlapping characteristics. The design of each Alfresco capability is not done in isolation but with consideration to the overall solution.

Complexity of Enterprise Content Management systems is often a barrier to entry. Many deployments do not reach their full potential due to restrictions enforced on developers, IT and users of the system.

Alfresco is driven to deliver a system which is simple to develop against, customize, deploy and to use.



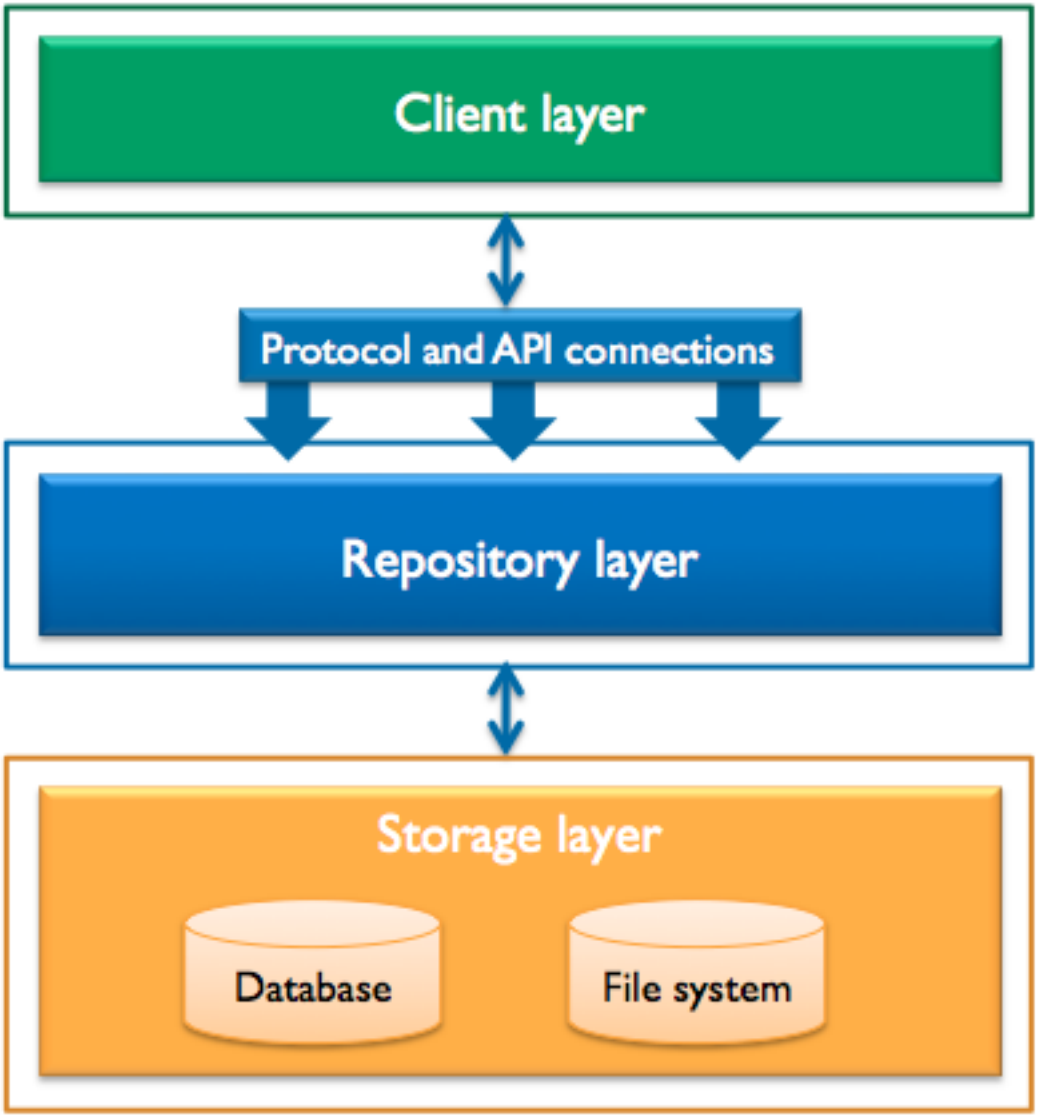
The shared document drive is the most commonly used application for Enterprise Content Management systems. Alfresco can present itself as a shared drive and this is the simplest and easiest user interface which users can access. It is one of many different interfaces which the Alfresco architecture has been engineered to deliver.

Layered architecture

The Alfresco architecture can be thought of in three layers. The storage layer is where the actual data and content is stored. This is managed by the repository layer. Because the two layers are separate they can be run on independent hardware to increase performance and to provide resilience.

The repository layer provides a multitude of protocols and APIs through which client applications can connect. When you install Alfresco various client applications are provided by default. Some organizations will develop their own bespoke applications or integrate existing applications with Alfresco to provide document and content services.

As for the storage and repository layers the client layer can also be deployed to a distinct physical environment.



Scalable

Since the Alfresco architecture is abstracted by layers, flexibility of system deployment is introduced. This is the mechanism through which scaling and clustering can be implemented, thereby affording support for vast volumes of content and large numbers of system users.

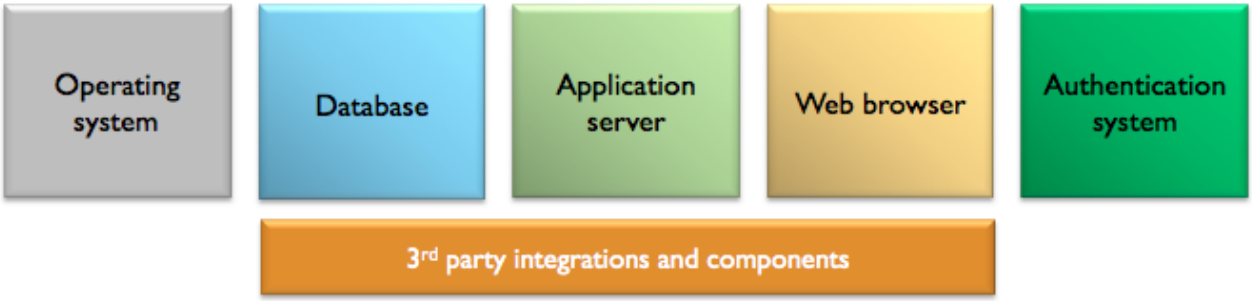
Embedded services

Enterprise Content Management often refers to services embedded within an end-user application. The Alfresco Enterprise Content Management system is written in one hundred percent Java. This is a trusted runtime for many enterprises wishing to deploy applications in their data centers. Each Alfresco capability is implemented as a black-box Java service that is tested independently and tuned appropriately.

As you have seen there are a range of application servers which can be used for Alfresco and you will be familiar with how your particular application server is implemented and structured. If you purchase any Alfresco Element (where there is lab work) you will see we have chosen to use the Tomcat application server within the supplied virtual machine.

Technology agnostic

Enterprise Content Management often forms part of a larger solution. The Alfresco architecture is engineered so it can be deployed to a range of operating systems, databases, application servers, browsers and authentication systems. Alfresco does not dictate this choice, but certifies the system for use within a large number of tested technology stacks and 3rd party integrations.



Supported technologies

Below is a list of Alfresco supported technologies. Use the following link to see specific version information.

`http://www.alfresco.com/services/subscription/supported-platforms/`

Database

- MySQL
- Microsoft SQL Server
- Oracle
- Postgres SQL
- IBM DB2

Application server

- Tomcat
- JBoss
- Oracle WebLogic Server
- Websphere

Java development kit

- JDK
- IBM Java SDK
- OpenJDK

Web browser

- Mozilla Firefox
- Microsoft Internet Explorer
- Safari
- Chrome for Ubuntu

Authentication system

- Active directory LDAP
- NTLM
- OpenLDAP
- Oracle Directory Server
- Kerberos
- JASS

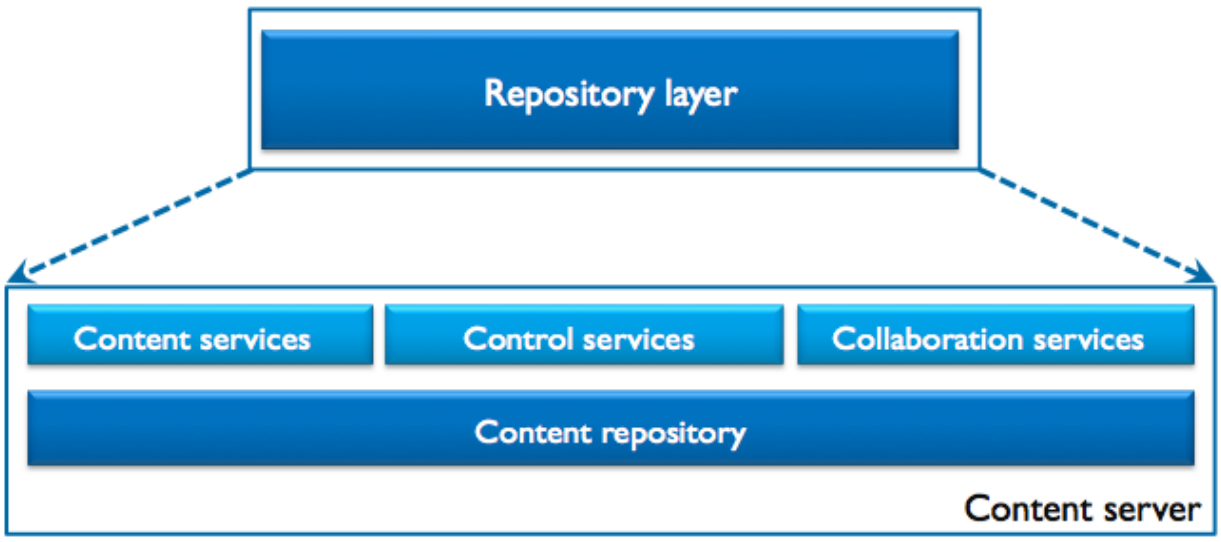
3rd Party integrations

- IMAP - Microsoft Outlook
- SharePoint - Microsoft Office
- Liferay
- Kofax

3rd Party components

- ImageMagick
- SWFTools
- OpenOffice
- VMWare ESXi
- FileZilla FTP
- Finder

Repository layer



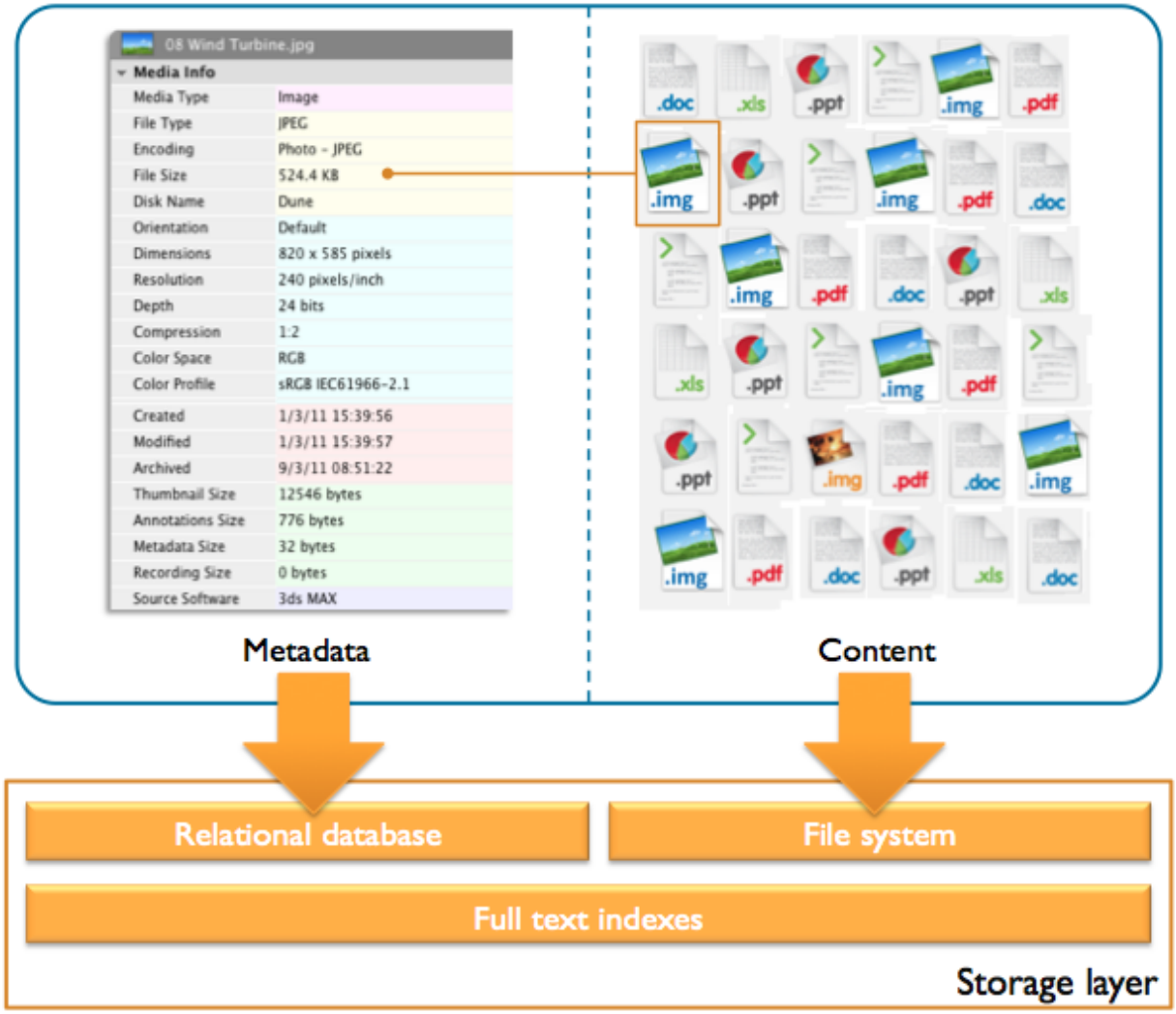
The repository layer is often referred to as the content server. It stores content of all types and provides a set of re-usable content management services such as content storage, query, versioning and transformation which may be used by one or more applications.

Control services enable workflow, records management and changes sets. Collaboration services provide functionality such as wikis, blogs, discussion forums and social media functionality.

Contents objects

Content files are constructed in two parts. The first component is metadata (often described as data about data). This contains information such as the creation date, modification date, keywords, annotations and a host of additional describing detail. The content itself is stored in the secondary component and this is the representation the user sees within an editing or viewing application.

Alfresco utilizes this file structure when managing content.



The metadata is extracted and stored in the database, thereby delivering a range of functionality which can be undertaken against this data.

The content is stored within the file system. Placing it within the file system (rather than in the database as some system do) delivers maximum performance when accessing, retrieving and working with the content. In order to support user searching and queries Alfresco indexes the content within a full text indexing system.

High level architecture

There are many ways to deploy Alfresco, using some or all of the solution components. Each component typically splits functionality across a client and server, where clients offer users and

interface to the solution and the server provides content management services and storage. It is common for a solution to offer multiple clients against a shared server, where each client is tailored for the environment in which it is to be used.

The Alfresco architectural components are discussed here:

Clients

The primary web based client is Alfresco Share. It provides collaboration for content and streamlines the user experience by introducing features such as web previews, tagging, metadata representation and editing, social publishing, blogs, wikis, discussions and many other collaborative functions.

It is implemented in Spring Surf and fully customizable. Alfresco Share can also be deployed to its own tier separate from the Alfresco content server.

Alfresco Explorer also exists as a web based client. This is no longer in development but still ships with Alfresco. It is important to be aware of its historical value. It was implemented in JavaServer Faces, but had to be deployed to the content server rather than deployed to its own tier.

There are a range of other clients and integrations; for example Microsoft Outlook via IMAP, Microsoft Office via the SharePoint protocol, Liferay and Kofax.

The client often overlooked is the shared drive, available to the operating system. This is probably one of the most common home-grown Enterprise Content Management solution where users share documents through a network drive. Using this technology, Alfresco can look and act just like a network drive. This is the only Java server-side implementation of the CIFS protocol. By using CIFS, users interact with Alfresco just as they do any other normal network drive, except the content is now stored and managed in the Alfresco content server. WebDav and FTP are also supported.

Content server

The Alfresco content server is primarily responsible for content definition and storage, retrieval of content, versioning, and permissions. The content server provides a highly reliable, scalable, and efficient implementation.

The Alfresco content server provides the following categories of services: content services, for example, transformation, tagging and metadata extraction; control services, for example, workflow, deployment and records management and collaboration services, for example, blogs, activities, wiki and social publishing.

Physical storage

Content storage, by default, is a combination of relational database for metadata and file system for content. In order to support fast text search in documents and other types of query Solr is used as the full text search engine.

Using a database immediately brings benefits that have been developed over many years such as transaction support, scaling, and administration capabilities.

Alfresco uses a database independent layer for interacting with the database, which brings additional cache control and SQL dialect support and allows Alfresco to be certified against all the prominent database implementations.

Content is stored in the file system to allow for very large content, random access, streaming, and options for different storage devices. Updates to content are always translated to append

operations in the file system. This allows for transaction consistency between database and file system.

Alfresco server startup

Alfresco can be installed as a service for Windows, a daemon for Linux or manually for either platform.

The modular nature of the Alfresco architecture means that the startup procedure will be different for each deployment. Certain components are core and this description is designed to give you a flavor for when these components are initiated and what part they play in an installation.

Early in the startup procedure checks are made to see if the database to be used with Alfresco is running, if not it is launched.

When the web application server initiates (in our case Tomcat) the file `web.xml` (~tomcat/webapps/alfresco/WEB-INF/) is loaded. This in turn loads Spring configuration and property files which launch the Alfresco server.

The Alfresco Element 'Repository configuration' examines these files and the configuration in greater detail.

Configuration and property files

In sequence the web application server reads the following files, which configure and launch the server:

- web.xml (~tomcat/webapps/alfresco/WEB-INF/)
- web-application-context.xml (~tomcat/webapps/alfresco/WEB-INF/)
- application-context.xml (~tomcat/webapps/alfresco/WEB-INF/classes/alfresco/)
- application-context-core.xml (~tomcat/webapps/alfresco/WEB-INF/classes/alfresco/)

Actions:

- Read Spring configuration files.
- Read property files (these enable changeable item configuration and system configuration).
- Check database for property values.

Initialize subsystems

The first subsystems from the content repository are initiated. Multiple subsystems launch asynchronously as dictated by the specific configuration. One of the first subsystems to launch is 'sysAdmin'.

Database schema updates

Check for and apply any database schema changes within the Alfresco installation.

OpenOffice

OpenOffice provides rendering and transformation services to Alfresco. A connection is attempted and if not successful re-attempted later in the startup process.

Rendering includes generating full screen previews of content. Transformation includes functions such as converting a file from one format to another, for example converting a Microsoft Word file to an Adobe PDF.

Alfresco module packages

Alfresco module packages (AMPs) are functional additions or patches, which can be applied to Alfresco in the startup procedure.

A single AMP file bundles XML, images, CSS and code which collectively extends the functionality or data provided by the Alfresco repository.

Server running

The final steps of the server startup procedure is to

- Deploy the share.war file which implements Alfresco Share. (~tomcat/webapps/)
- Register Web Scripts. (Web Scripts are services which responds to HTTP methods such as GET, POST, PUT and DELETE to perform functions within Alfresco. A number are supplied and they can easily be created without knowledge of Java.)

This is the basic ordering to the Alfresco server startup. Be mindful that certain components rely on each other, some components are started asynchronously, while others, for example patches, may be applied in several cycles.

For greater details see the 'Repository configuration' Alfresco Element.

Demo: Alfresco server startup

Having explored the startup process by component, let’s look in greater detail.

In this example Alfresco has been installed manually. The alfresco.sh command is used to initiate the server. This checks if the database to be used with Alfresco is running. In this instance it’s not, and Postgres is launched. The Tomcat web application server is next started. The alfresco.sh script ends as tomcat is now launching.

The alfresco.log file records the actions and errors of the Alfresco server and we could examine this, however I tail the tomcat log file catalina.out since this shows the same detail with some additional information I wish to highlight.

The tomcat server reads the web.xml file and then a number of other Spring configuration and property files which define and control the Alfresco startup. The first of the sub systems is started, in this instance the ‘sysAdmin’ subsystem. Many others will be launched at various points in the process.

The system checks if any database schema changes are required for the Alfresco database. A connection to the rendering and transformation service, OpenOffice, is attempted. The service is not presently available. Next checks for any AMP files or patches to be applied are made. These would be written to the alfresco.war file (which contains the application), prior to that file being expanded and deployed.

Illustrating the asynchronous nature of the startup process a number of other subsytems are initiated. These include, file systems, email, Google Docs, and others Apache Solr is next started.

A second attempt to connect to the OpenOffice server succeeds now that service is running. Alfresco Share is deployed and expanded from the share.war file.

Web Scripts are registered and deployed. Web Scripts are an enabling and extensible technology, core functionality of Alfresco is written in Web Scripts. The Alfresco server is now fully launched and available.

Architectual decisions

When you are planning out your Alfresco installation there are some key architecture decisions which you will make, these relate principally to where you place different components of the system. The location of components can have a dramatic effect on the performance and reliability of the system, so it is important to get this right.

There are decisions about which you have no choice over and there are some general rules which you can use to ensure that you get a good and solid architecture.

Architecture deployment

Whilst for development purposes you may often run all the components of Alfresco on a single system, for production it is best practice to deploy your database and the content server on different machines. This gives you greater flexibility when configuring the hardware and tuning the system's independently.

This configuration should be treated as the minimum configuration for production. Obviously your real configuration will vary depending on the application for which you are using Alfresco. For example if you are using Alfresco for Image Management and therefore doing scanning you will need other servers to host the scanning software.

The flexibility in the Alfresco architecture affords the possibility of moving the front end application server to one or more servers; allowing for improved performance, load balancing and redundancy.

There are some hardware and software considerations you should take into account when building your production Alfresco infrastructure. Firstly with respect to hardware you should use server class machines with SCSI Raid disk arrays. The performance of reading and writing content is almost solely dependent on the speed of your network and the speed of your disk infrastructure, the overhead of the Alfresco content server itself for reading and writing content is very low. So SCSI should be considered the entry level configuration, you should seriously consider using a SAN infrastructure for enterprise scale performance.

In some cases it may be tempting to make the file system remote and mount onto the content server machine across a LAN or WAN, this is not advised and will cause performance problems. The file system should always be attached to the content server through a high-speed connection such as fibre channel.

As you have seen the operating system must be 64bit since the size of a Java Virtual Machine is restricted to 2GB on a 32bit operating system. This will not support the requirements of a production Alfresco deployment.

Apache Solr

Alfresco version 4 introduced a change in architecture for indexing and user search; Apache Solr is now the default full text indexing technology. Lucene was used prior to this and can be enabled in favour of Solr if required.

Solr embeds and leverages Lucene, however Solr provides a functionally abstracted layer and can be installed on a dedicated server.

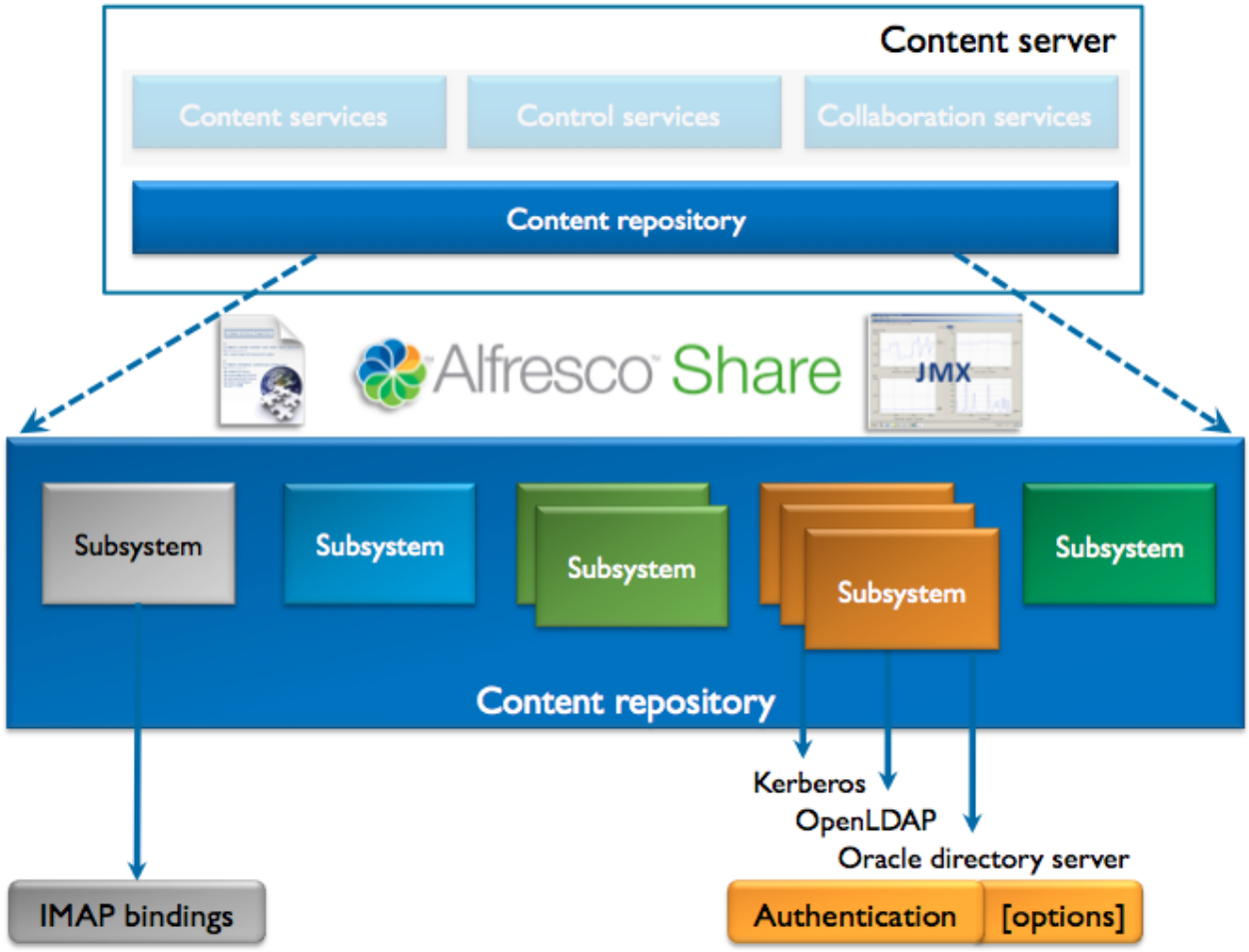
Solr delivers a number of advantages over Lucene including scalability, improved performance, enhanced language support and fine control over what gets indexed.

A principle issue for Lucene is transactional dependence; the indexes are updated as content is loaded or edited. This delivers fully accurate and synchronised indexes, but means the content

server is locked during the transaction. This is a particular problem if the indexes ever have to be rebuilt from scratch, an operation that can take many hours with large content repositories.

The drawback and consideration when using Solr, is that it works asynchronously and is known as “eventually consistent”. Content is indexed at small regular intervals and until the indexes are up to date.

Subsystems



Let’s examine the content server and the architecture of the content repository.

The content repository is comprised of a number of subsystems. These are configurable modules 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.

As a unit, subsystems have their own lifecycle and they may be shut down and restarted while the Alfresco server is running. This is useful if you wish to disable an aspect of the server, or reconfigure parts of it, for instance how LDAP synchronization is mapped.

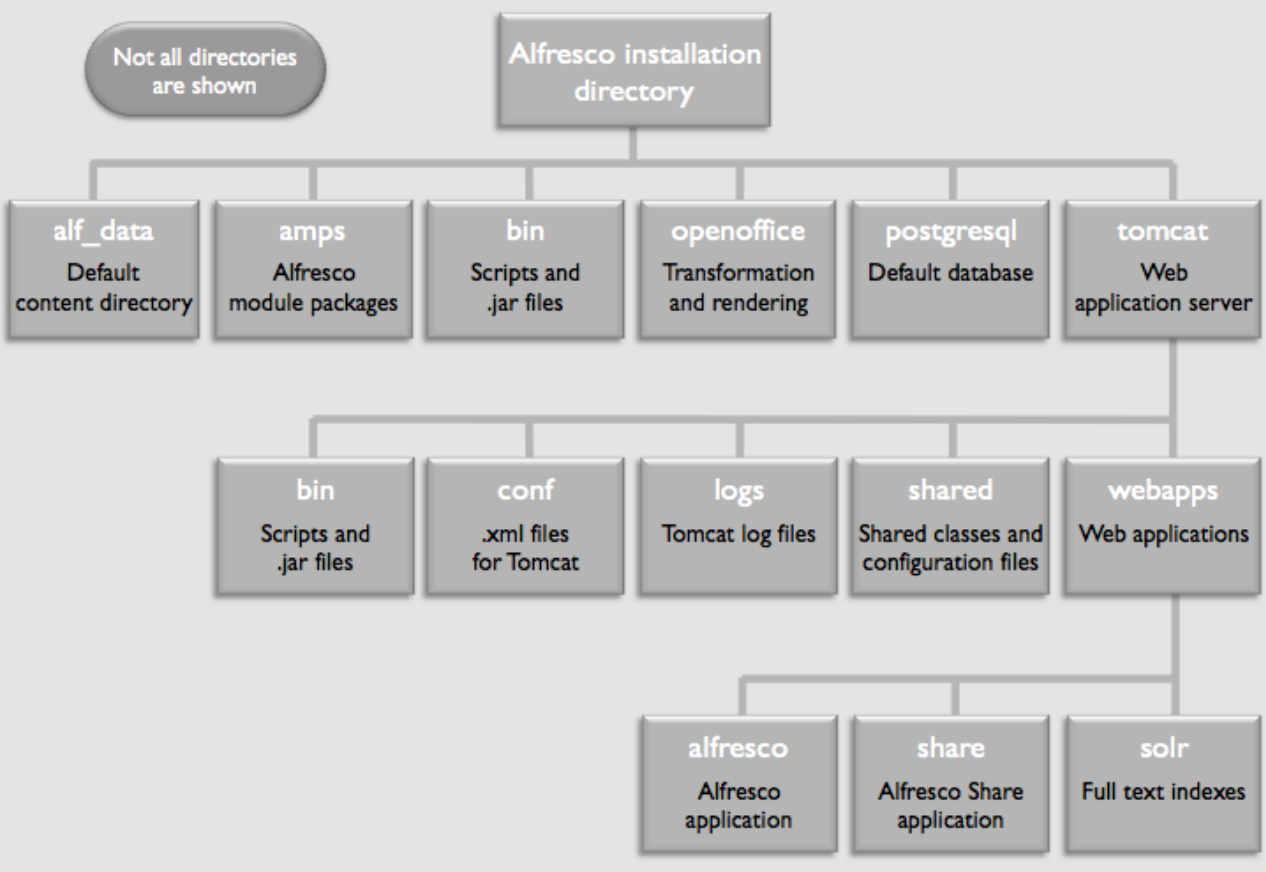
Each subsystem supports its own administration interface that is accessible through the administrator’s console of Alfresco Share, within property files, or directly through a management console using Java Management Extensions (JMX).

Subsystems are implemented so that there may be multiple instances of the same type where this makes sense. For example there may be more than one way of facilitating authentication.

Within the ‘Managing the Repository’ Alfresco Element you have the opportunity to configure and manipulate subsystems.

Deployed file strucutre

This diagram below depicts an Alfresco installation. This is the file structure as created by the installer.



In production systems it is unlikely you would use the installer to deploy Alfresco, components are typically installed manually, for example directly to the web application server. Additionally remember that individual components can be split out to their own tier or server and therefore this is only a general representation.

alf_data

This is default location for storing Alfresco content and includes the data dictionary for all nodes, full text indexes and the actual content.

Naturally this is one of the most important directories for your backup and recovery regime.

The Solr indexes are an obvious target for moving to a separate server, as can the content (when used with an appropriate network infrastructure).

tomcat

This directory is the web application server, responsible for running and presenting the Alfresco instance. In a production environment it is likely other applications will also be installed here.

This is the default location when Alfresco is installed using the supplied installer. You may well place tomcat at a different location or you may be using a different web application server.

Within the tomcat directory the Alfresco application installation is split over two directories; shared and webapps.

shared

This directory contains java classes and configuration files which are shared across the whole installation, we refer to ~tomcat/shared/classes as extensionRoot. This is important because many of the configuration files are noted as being relative to this point in the structure.

Configuration files are either .properties or .xml files. These define functionality and characteristics for Alfresco when initiated and includes the important Alfresco global properties file.

webapps

In this directory are the Alfresco web applications, they are structured in the conventional way for your application server.

There will always be an alfresco directory containing the server and its components. Depending on how your system is installed and the applications you have deployed you may see application directories such as share, solr, mobile and others.

alfresco

The alfresco directory contains the deployed alfresco server. This is extracted and deployed from the alfresco.war file (which is held in the webapps directory) when the server initiates.

The alfresco server manifests and presents the content server (i.e. the content services and repository).

It is of course possible to deploy the server without the related share directory (which deploys Alfresco Share) as the alfresco server can be accessed from many different user interfaces other than a web browser.

share

The share directory contains the Alfresco Share application, the web user interface to the content server.

This is extracted and deployed from the share.war file (which is held in the webapps directory) when the server initiates.

It is possible to deploy the alfresco content server without share as any one of the other possible user interfaces can be used to access the content server.

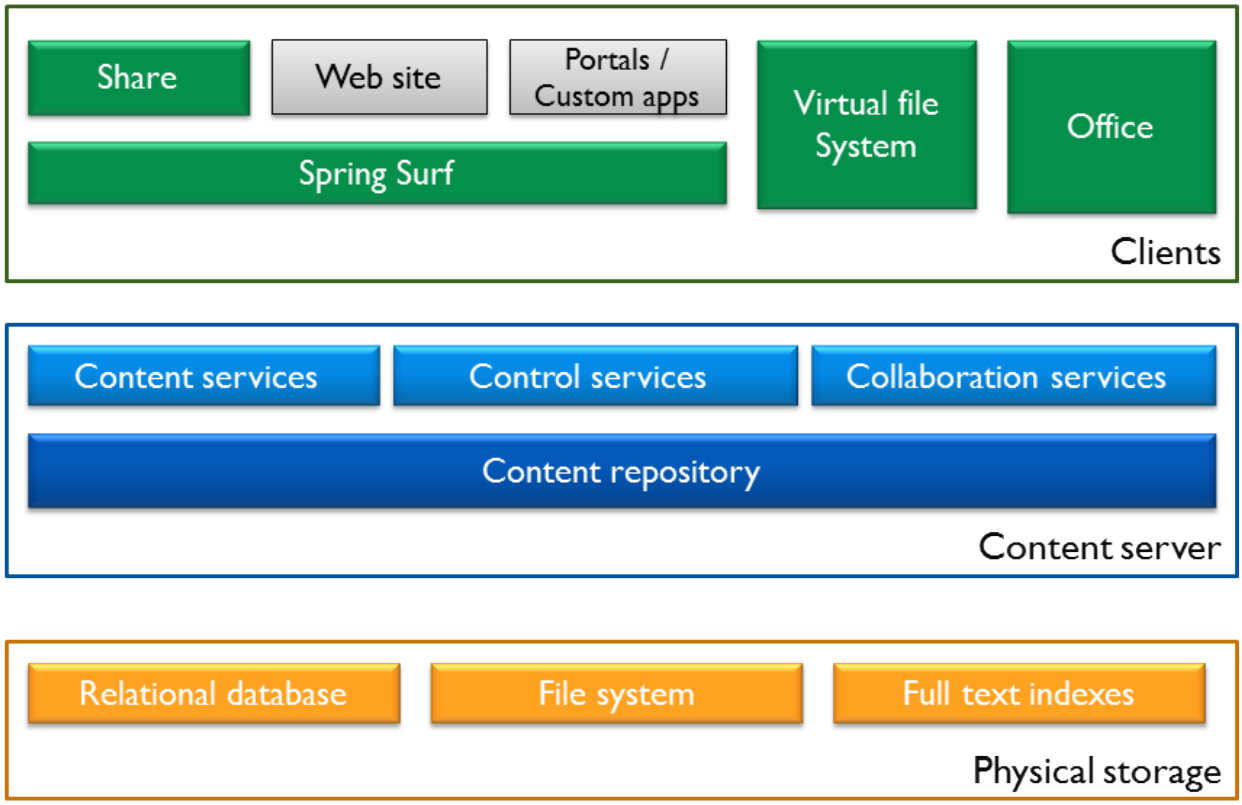
User Interfaces

Introduction

In this Alfresco Element we will take you through a high-level review of the Alfresco user interfaces. These include Alfresco Share, Records Management, Web Content Services, Mobile, SharePoint, email and virtual file systems such as CIFS, WebDav and FTP. You will be introduced to each interface, understand their respective strengths and weaknesses and practice what you have learnt through labs.

Alfresco Share

Alfresco Share is a collaborative user interface which enables you, and those in your organization, to access, edit and manipulate content in the repository. There are many ways of accessing the repository, however Share is the most flexible and functionally rich interface, which Alfresco offers. It is also where administrative functions are undertaken.



Most organizations will be using Share, a customized Share environment or their own bespoke interface. Even if you are using your own bespoke interface familiarization with Share helps you to understand the concepts behind Alfresco. This Alfresco Element covers the basic functionality of Share and how to use it for collaboration. A deeper exploration of Share is conducted in the Collaboration for End Users course.

When collaborating using Share in an Alfresco repository the collaboration process revolves around a site. Sites may be public or private and will have a number of components designed for collaboration, such as wikis, blogs, discussion forums, calendars, links and a document library. Any documents created in the site will appear in the repository and the site as will any blogs, wiki pages and other items. The administrator or site owner has control over which of these components appears for any individual site.

Sites can have any number of members, these are users within the Alfresco repository, and a user can be a member of any number of sites. When a site is private, or moderated, users have to be invited to become a member of the site before they can see it, public sites, however, are visible to everyone.

If you are deploying the Records Management module in your organization you will see that it is implemented as a special Share site of which there can be only one, the Records Management site also has special functionality which is exclusive to records management.

As a user you can search either in sites of which you are a member or you can search across all sites. Global search does not circumvent the permissions so the results which are returned will be only those objects which you have access to see.

Finally the Alfresco repository supports the notion of a global document library, documents which exist in the global document library are available to everyone, but access can be restricted on a document by document basis. Documents existing in the global document library may also be linked into one or more sites. Now that you understand the underlying structure of Share, let's login and explore.

Demo: Share

When you login to Alfresco Share you will be taken to the user dashboard, this is your home page. The dashboard is a customizable area where you can select various preconfigured layouts or construct your own layout. The Welcome banner highlights immediate actions you may wish to undertake; viewing tutorials, creating a Share site and editing your profile. The components of the dashboard are called dashlets.. These are individual modules, which express some unique functionality; such as a list of Sites you are a member of, your task list and documents recently edited. Each dashlet has a help icon which describes it’s functionality. Dashlets are Web Script components and it’s naturally possible to extend the range of available dashlets by writing your own or using those written by the community. The Alfresco add-ons web site (found at `addons.alfresco.com`) is a good place to explore such examples.

To customize the dashboard use the customize dashboard button. Here you can change the layout, for example to a three-column view. You can remove dashlets from the dashboard by simply dragging them to the trash icon. To add a dashlet use the add dashlets button. Drag the desired dashlet to the column in which you wish it to be presented. The OK button confirms your changes.

Demo: Create site

In this demonstration a site for Green Energy will be created and also a document structure. A lab follows this demonstration where you will have the opportunity to undertake this process for yourself.

Heloise Dufresne (Green Energy’s chief executive officer) logs in. She chooses Create Site from the sites menu. Any user can create a site and the site creator will become the site manager. In the Create Site dialog she enters the site name, URL and description. There is only one site type in Alfresco at this time, a collaboration site. This site will be public and non-moderated. When the site is created the site dashboard automatically appears. This is unique to the site and distinct from the user’s home dashboard. The banner offers typical functions you would wish to undertake within a collaboration site, such as uploading content and inviting others to join the site.

Heloise will now create a folder structure, which will be associated with and contained in this site. She chooses Document Library, then new folder. She creates the Quality Assurance folder. Note that by default folders are not shown, only content. Heloise clicks Show Folders. Entering the Quality Assurance folder Heloise now creates two sub-folders; Testing and, Safety and Compliance. She further creates the folders; Manufacturing, Processes and Best Practices, Specifications and Reference. These folders will of course be accessible in the global document library, the Repository. They are primarily associated with the Geo-NRG Product Development Share site.

Lab - Site and folder creation

1. In your first lab please complete the following tasks:
    1. Login as admin
        - The Share login URL in your virtual lab environment is `http://localhost:8080/share`
        - The account and password are both: `admin`
    2. Create a site called Wind-NRG Product Development.
        - The URL name is `windnrg`, give it your own description.
    3. Create a folder structure in your new site of;
        - Requirements
        - Quality Assurance
            - Testing
            - Safety and Compliance
        - Processes and Best Practices
        - Specifications

User interfaces

Demo: Customizing the site

As the site owner Heloise has the ability to customize the site. She adds the Calendar, Wiki, Blog and Data List functions, which will appear across the site banner bar.

Customizing the site dashboard she adds the wiki dashlet to the Geo-NRG Product Development site. When users join this site they will experience this layout. Share sites can be branded with their own unique theme to differentiate sites.

The standard site components can be renamed to create a more meaningful relationship to the site. Heloise chooses to rename both the Document Library and the Calendar.

- rename Document Library to Geo-NRG docs
- rename Calendar to Geo-NRG events

These changes will be seen on the site dashboard.

At any time Heloise can update the site details. She first updates the description. add “ , due for launch in 2012” Heloise could make this site moderated. If she did users within the system would still be able to see the site, however it would only be possible to join the site if invited. The alternate action is to make the site private. This would hide the site from all users, unless invited to join the site.

It is possible to resize dashlets by holding the bottom of the box and dragging to the desired size. These are some of the customisations available to a site manager to enable them to create a unique Share site for collaboration.

Demo: Wiki

In this demo Heloise will create a collaboration wiki for the Geo-NRG Product Development site. She first heads to the Share site, then chooses wiki. She edits the main page, adding a description. When she clicks Save the page will be displayed.

To add a new page to the wiki she selects New Page and populates this with a title and description. Clicking Wiki Page List will display both pages. To create a link from the Main Page

to the Requirements page, Heloise will edit the Main Page. She uses the standard wiki link tag and adds the page name. Clicking Save the page is displayed and you can witness the link working.

Lab - Wiki

1. In this lab you will create a site wiki. Your tasks are:
    1. Edit the main wiki page.
        - Adding your own description.
    2. Create a requirements page, adding the text:
        - This page is for us to collaborate on the requirements for the product line.
    3. Create a link on the Main Page to your new Requirements page.

User interfaces

Demo: Editing document

In this demonstration document uploading is examined. Within the Geo-NRG Product Development site Heloise accesses the Document Library and navigates to the Reference folder. She clicks the Upload button and chooses the Geo-Thermal Backgrounder document. Once loaded this document can be viewed within Alfresco share. A PDF rendering is automatically generated alongside the document metadata.

Heloise notices a typo in the document. She chooses Edit Offline, which checks the document out for editing and initiates its download. Having corrected the document, she uploads it by using Upload New Version. Alfresco Share uploads the document recording it at version 1.1.

Lab - Document editing

1. In this lab you will upload a document, edit this offline and upload the new version to the Document Library. Your tasks are:
    1. Within your Wind-NRG Product Development site navigate to the Specifications folder.
    2. Upload the document 1.3kW Specification.doc
        - `Desktop > Sample Documents` folder
    3. View the document.
    4. Choose Edit Offline and save the document.
    5. Locate and edit the document using Open Office Write (which is installed).
    6. Upload the new version back to the Document Library ensuring you add a comment.

User interfaces

Demo: Data lists

Within your Share site it is possible to create an arbitrary list, known as a Data List. There are a number of inbuilt Data List types. Each exposes a slightly different set of fields in the user interface, in which to record information. (Being extensible it is possible to create your own custom Data List type, through content modeling.)

In this example Heloise selects the Issue Type Data List giving it a title of Design issues and a suitable description. Clicking on the newly created Data List she chooses to create a new list item. Heloise gives this an Issue ID, title, priority, description and due date. Other attributes exist such as the ability to assign this to an individual, set a status or add an attachment (an object from the repository).

Once created the Data List item is shown. Heloise now creates a second Data List item. Using the column headings it is possible to sort the Data List.

Lab - Data Lists

1. In this lab you will explore Data Lists.
    1. Create a Data List of the type Task List.
    2. Name this **Requirements gathering tasks** and add your own description.
    3. Create a number of data list items.

User interfaces

Demo: Blog

Share sites naturally support blogs. Within the site, Heloise clicks on Blog and then chooses New Post. A title for the blog can be entered, along with some text. The text entry supports a range of formatting options.

In this blog Heloise is going to add a title graphic. She saves the blog as draft and navigates within the Document Library to a graphic that is already loaded in the repository. She copies the document URL. Navigating back to her draft blog Heloise uses the Insert/edit image button to add the graphic into her blog entry.

She also wishes to link to the data list she previously created. Navigating to the Design issues data list she copies the URL. Within her blog entry she can now create her link. There is the option to publish a blog externally, in this instance she chooses Publish Internally. This will now be shown in the blog list and can be viewed.

Lab - Blog

1. You will create a blog entry in this lab.
    1. Navigate to the site blog and create a new post.
    2. Give this a title and add some text of your choice.
    3. Experiment with the formatting controls.
    4. Create a link in your blog post to the requirements gathering data list you created in your previous lab.

User interfaces

Records managment

Alfresco offers Records Management. This is certified to the DoD 5015.02 specification. This is presented as a unique Share site.

Records are information recorded as evidence and information by an organization, to fulfill legal compliance. Records can follow a lifecycle whereby they are identified, classified, archived, preserved and destroyed.

Classical industries that use Records Management are healthcare, government and financial institutions. Of course the practice of Records Management is not limited to these.

Demo: Records managment

In this demonstration we will take a small tour of the Alfresco Records Management functionality. The Records Management Share site is created via the Records Management dashlet. Once created users will be able to access and join the site.

Green Energy's business development executive vice president Bill Loman logs in. Searching for the Records Management site he joins. Bill will create a file plan, this is the structure under which records are filed. It has a defined hierarchy of series, category and then folder. Records can only exist within a folder. He defines two different classification structures, one for contracts and one for purchase orders. Before uploading any content Bill will define a disposition schedule. This defines rules and actions regarding a record's lifecycle, and is defined at the category level. This is first defined by a cutoff event. In this instance the cutoff event is the expiration period of a contract. When the cutoff has occurred the record should be destroyed in 6 years time.

Now the disposition schedule is created Bill will file two records. He navigates to the Procurement series, Contracting category and then the Contracts folder. He chooses File and selects the document Joint Venture Contract.pdf This is loaded as an undeclared record. Undeclared records can be edited or removed from the repository. Before declaring the uploaded document as a record, mandatory data must be added, which he enters. Now this is complete Bill will declare the record. Bill will now upload a purchase order document into the Requisition category, purchase order folder. Again to be able to declare the record he must complete the mandatory data. Once complete he declares the record.

A relationship can be created between two documents. Bill will relate the contract and the purchase order. He first selects the Purchase Order document, then moves to the References section and clicks Manage and enters a reference name. Bill chooses Select to link the two records. And finally sets the relationship to be Cross-Reference. The relationship is now established.

To show the lifecycle of the contract record, Bill navigates to the Records folder and completes the cutoff event. Heading to the Contract document we now see the cut off date has been set and therefore the document will be destroyed in 6 years time. Auditing is an essential component of the Records Management specification. The Alfresco Records Management module provides full auditing capability from simple detail such as failed logins to a full account of a particular user's actions.

Alfresco Records management provides sophisticated capabilities for searching and saving searches, freezing and holding records, support for vital records. Additionally it has functional access capabilities coupled with an enhanced security model which includes supplemental markings.

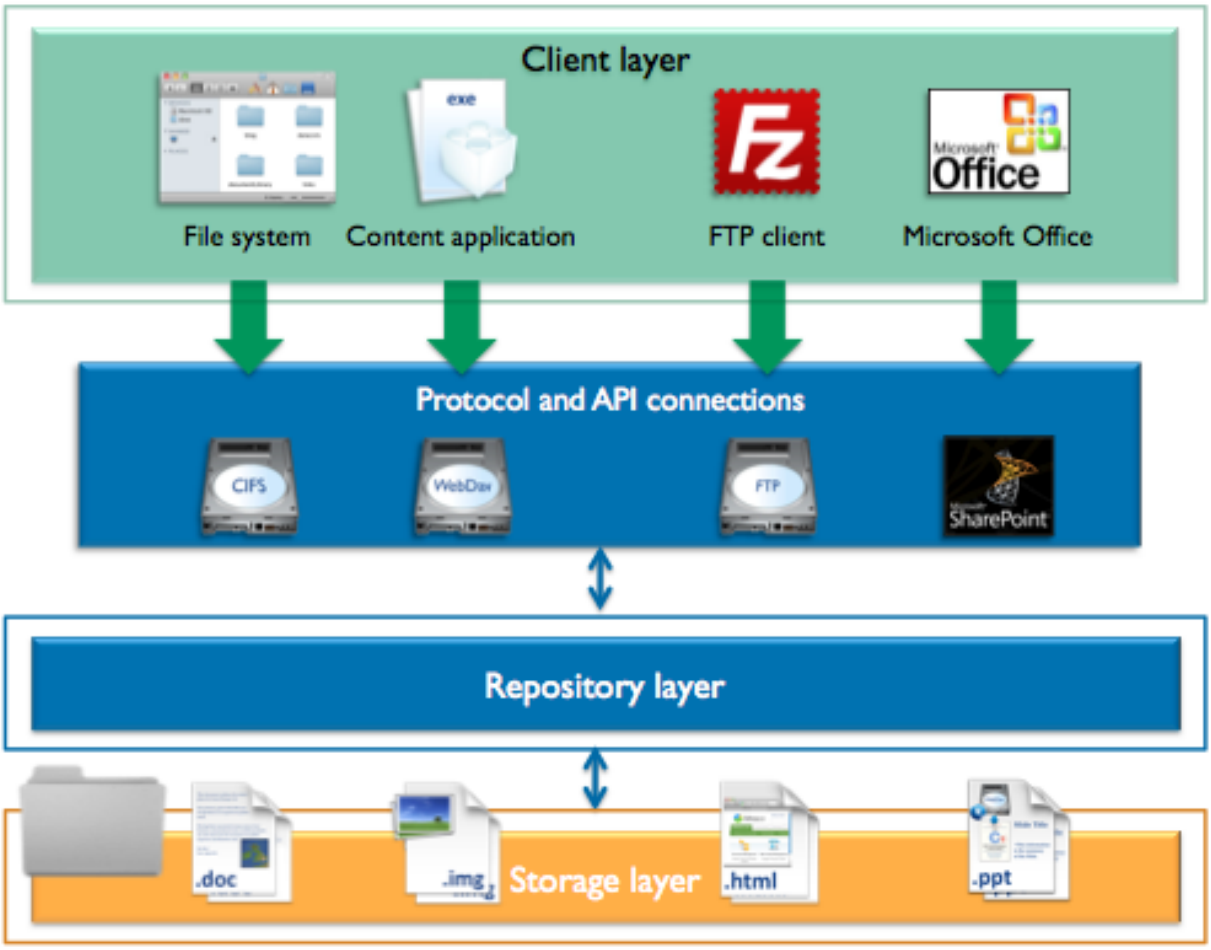
Records managment courses

Records Management is explored in greater detail on the courses: Records Management for End Users and Records Management Administration.

Web quick start

Alfresco Web Quick Start provides a collaborative environment to enable individuals to work together to publish dynamic, rich, industrial grade web sites. The primary user interface is Alfresco Share allowing users to create, modify and publish content without having to understand how the content is stored. Users utilize their existing knowledge of Alfresco Document Management.

Content creation and management can also be undertaken directly through Alfresco virtual file system interfaces.



Developers can customize and enhance a web site using Java, Java Script, Freemarker, XSLT and many other industry standards.

Demo: Web quick start

In this demo I will use the Share interface and the Alfresco Web Quick Start. The Web Quick Start provides a set of web site design templates, and the ability to instantly publish a working web site.

The example web site includes a number of elements, including search, section navigation, blog, news stories, a banner and others. As a content author I select blog to show the current blog entries. Selecting the first item I am presented with the in context editing tools, where I can create a new blog, edit or delete entries. Choosing edit I am able some new text. Formatting tools allow for text customization. When I submit the change, the blog entry is immediately updated.

The banner provides site navigation.

To add a new section I return to Alfresco Share and the document library and simply choose New Folder where I can enter name and title. The title will be the item that appears on the web site. Examining the folder metadata you can see it is possible to hide (or exclude) this folder from the navigation. Templates enable customization and rendition controls how content is displayed. Returning to the published web site, I can see this section has been automatically published.

One of the inherent processes of Enterprise Content Management is workflow. The Web Content Management within Alfresco naturally enables workflows to be applied to web content. For a review, approve and publish process I start a workflow for this blog entry. I enter a message for the approver, choose a due date and as this is an important post set the priority to high. Next I choose an assignee, the person who should approve the blog. Finally I start the workflow. This icon indicates a workflow is in process against this document.

The web site home page is comprised of dynamic content collections. This includes a section for blog entries, news and analysis and featured items. If I wish to change the featured items I head back to the document library and select collections, then details for the featured items. I delete the two existing entries and add two blog items. After I submit these I return to my web site and witness the featured items list updated with my changes.

Social publishing

Alfresco Share offers Social Publishing, that is the ability to post content directly from the Alfresco repository to social media and social networking sites. This functionality is explored in detail within the Alfresco Element “Social Publishing”.

Alfresco mobile

The Open Source, iPhone and iPad application provides a further interface to the Alfresco content repository, affording access from any location at anytime. The functionality of this application is explored in the Element “Alfresco Mobile”.

Virtual file systems

Alfresco Virtual file system support enables a user to interact with the content repository through interfaces other than Alfresco Share.

Through an operating system, application or the SharePoint protocol a user accesses, edits and saves documents to the Alfresco repository. The functionality of the repository is fully implemented and a user may not even realise they are using an Enterprise Content Management system. The virtual file systems are CIFS, WebDav and FTP.

CIFS

The Common Internet File System (or CIFS) virtual file system is unique to Alfresco and presents the repository as a file system. Applications do not need to be aware of the underlying protocol as Alfresco interacts with the operating system to present it as a native file system, so any application can interact with it.

Demo: CIFS

This demo will show the user experience of CIFS. Settings within the `alfresco-global.properties` file are made to configure CIFS for your particular environment. Once made CIFS can be enabled or disabled within the administrator’s console.

A user can use their operating system to connect to the Alfresco repository. The directory structure presents itself as a shared drive, where you can see and work with content. You will have noticed some additional files in each repository folder, that is the `_CheckInOut.exe` and `_ShowDetails.exe` programs. These don't literally exist in each directory but offer the additional functionality of CIFS server. As an example it is possible to check out a file, which creates a working copy. Once the file is edited the natural step is to check the file back into the repository.

To show the experience on Macintosh and Ubuntu Sebastian logs in using these different operating systems.

WebDav

Web-based Distributed Authoring and Versioning, or WebDav, is a set of extensions to HTTP that lets you manage files collaboratively on web servers. It has strong support for authoring scenarios

such as locking, metadata, and versioning. Many content production tools such as Microsoft Office suite support WebDav. Additionally, there are tools for mounting a WebDav server as a network drive.

WebDav vs CIFS

CIFS has a number of advantages over WebDAV. With CIFS, it is easy for anyone to mount the repository as a shared drive without any client software. It can be a replacement for your S:drive. Microsoft has put in a lot of effort to allow the Windows file system to be able to synchronize changes with a shared drive. You can do that with Alfresco without any client software, you cannot do this with WebDAV.

CIFS has a lot more properties visible in Explorer: Just right mouse click on one of the columns in your Explorer window to see the options available: author, title, date created, comments, etc. These are captured automatically by Alfresco, with WebDav, you really only get name, date and URL.

Through the CIFS protocol, Alfresco can see better what programs like Microsoft Word are intending when they open, save and seek information. Therefore we can predicatively version content and provide for recovery of old versions of documents. Something that you can't even get with normal shared file drives.

FTP

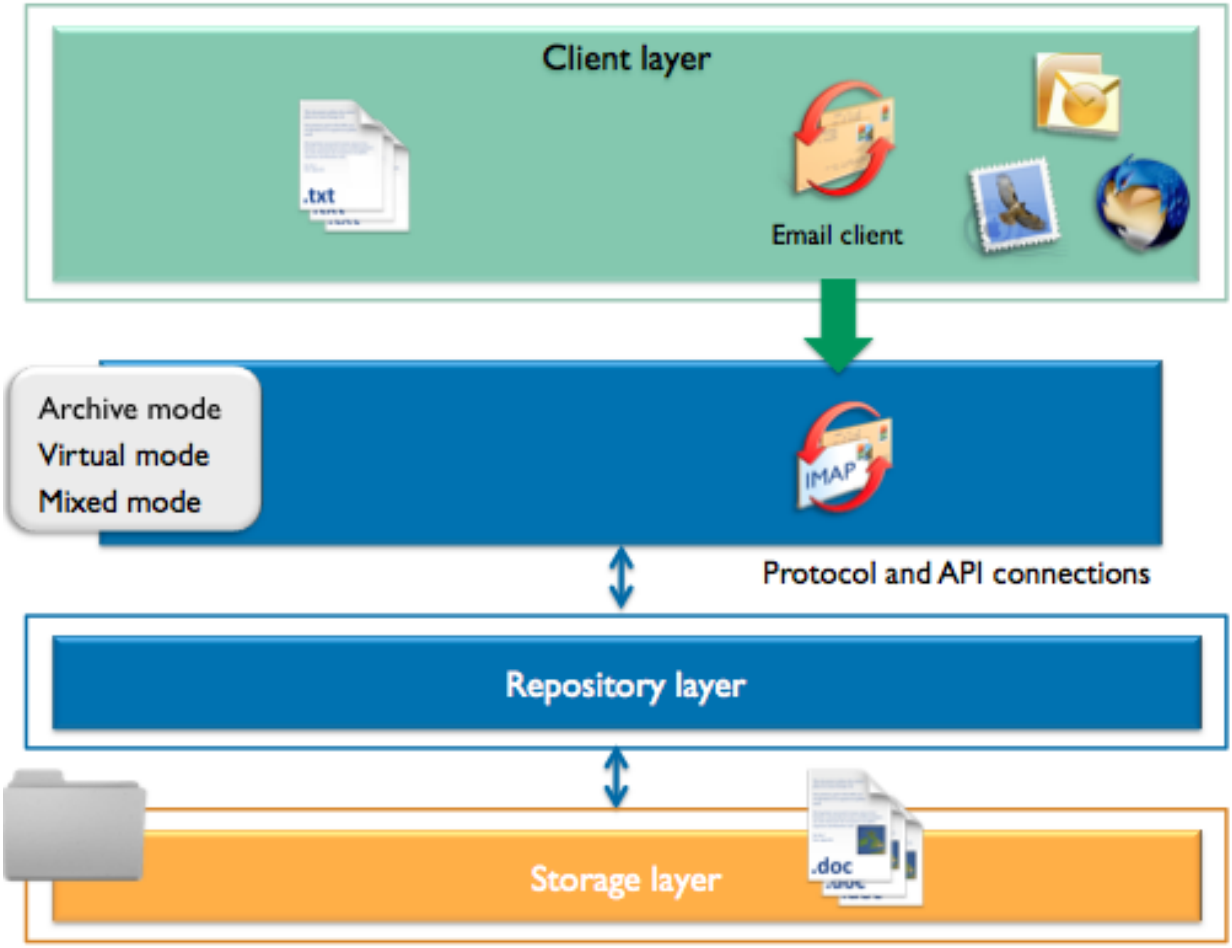
FTP stands for File Transfer Protocol, this is a standard network protocol for exchanging and manipulating files over a network. This protocol is particularly useful for bulk loading folders and files into the Alfresco content repository.

Share point protocol

Microsoft SharePoint protocol support allows Alfresco to act as a SharePoint server allowing tight integration with the Microsoft Office suite. This allows a user who is familiar with the Microsoft task pane to view and act upon documents held within the Alfresco content repository. The collaborative features of Microsoft SharePoint such as Shared Workspace are all mapped to Alfresco Share site capabilities.

IMAP protocol

Alfresco provides support for accessing the Alfresco server via the IMAP protocol. The protocol allows email applications that support IMAP (including Outlook, Apple Mail and Thunderbird) to connect to and interact with Alfresco repository, directly from the mail application.



The archive mode allows emails to be written to and read from the Alfresco repository by the mail client, by dragging and dropping or copying and pasting.

A virtual mode allows documents managed by Alfresco to 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.

The third, mixed mode allows a combination of both archive and virtual modes to be used, so document access and email management are available.

Summary

In this module the Alfresco user interfaces have been examined. The simplicity of web browsing is harnessed by Alfresco Share, Records Management and Web Content Services. Virtual file systems provide access to the Alfresco repository, via a shared network drive. Email and Microsoft Office integration is afforded through IMAP and SharePoint protocol respectfully.

Users and Groups

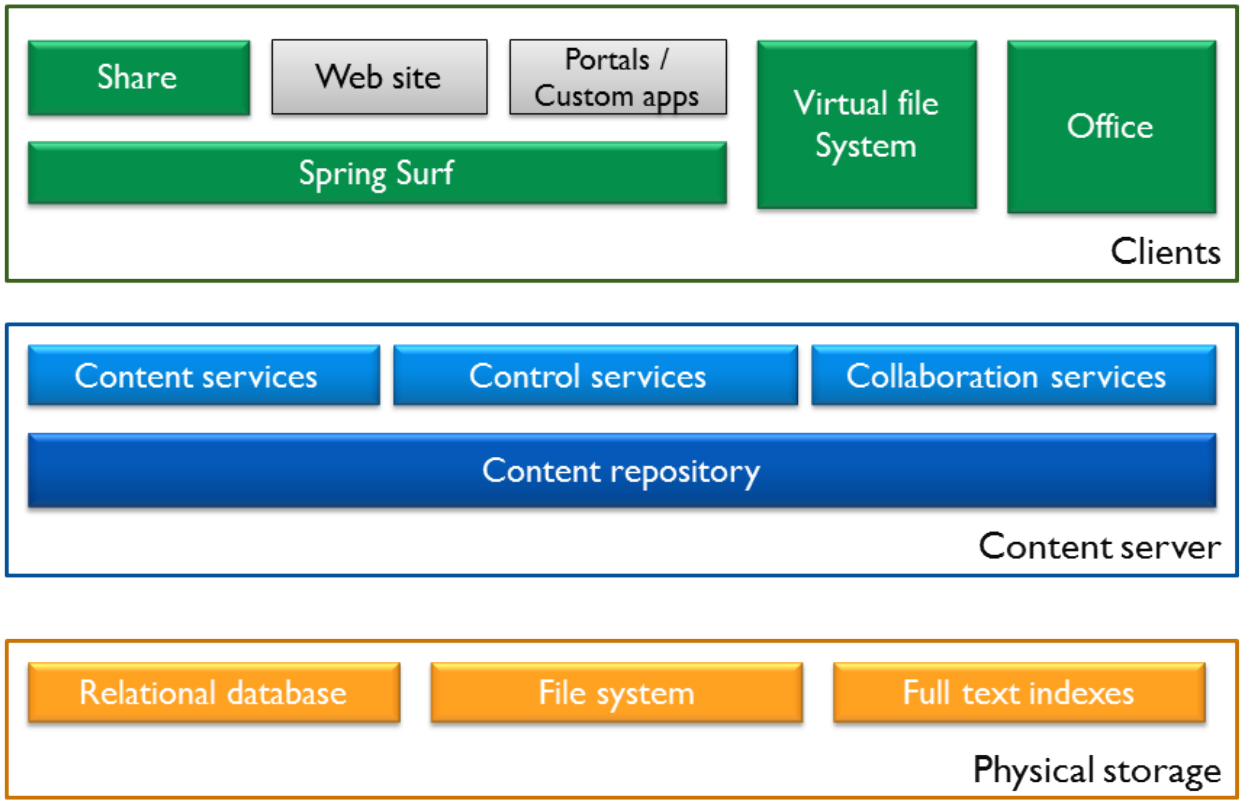
Introduction

In this Alfresco Element we will look at the creation and management of users and groups within Alfresco. These tasks are undertaken by the administrator, which is the role you will assume for this Alfresco Element. We look at the relationship between users and roles and the options for authentication.

At the end of this section you will understand the relationships between users and groups and how to create groups. You will also understand the different roles in Alfresco and be able to list the different authentication methods possible.

Users

In order for a person to access Alfresco they must have a user entry within the Alfresco database, this is called an authority. Typically a user corresponds to an individual person and if you do not have a user account in Alfresco then you have no access. There is a special user called admin, created during the installation process, which carries system administration privileges and this is the account you will be using for your administrative tasks, but administrative privileges are assigned through a group so you may have other users with these privileges.



As an administrator you may create users directly in Alfresco or you may synchronize your users with an external authentication system. When creating users directly in the internal database you must specify a password for authentication, when synchronizing users with external systems the authentication is carried out by the external system, for example LDAP or Active Directory.

Demo: Users and groups

In the Alfresco Element, Share for administrators, the creation of users and sites was demonstrated. We'll now examine user roles in relations to sites.

As a user of Alfresco it is possible to automatically join any existing non-moderated, public site (such as the Business Development site). Joining in this fashion sets the user role to consumer, which is the role with the least number of permissions. Sebastian will leave the site as he requires the role of collaborator within the Business Development site. When in a site it is also possible to leave by using the Actions menu.

Since Matt Black created the Business Development site he becomes the owner and manager of the site automatically. A site owner can invite others to join the site and also set their role. We will examine the different roles in detail later in this course Element. Any role can be assigned, which includes creating more than one Manager for the site, if required.

When Sebastian next logs in he will receive an invitation to join the site in his My Tasks dashlet. When he accepts he becomes a member of the site with the role as issued in the invitation. There are now three members of the Business Development site; Matt, Heloise and Sebastian, who hold site Manager and Collaborator roles.

Groups

Alfresco allows suitably privileged users to create groups, which as their name suggests are a collection of users. As you can see the human resources group contains two users. These users may exist in more than one group. For instance you can see that the group manufacturing also contains Michelle. We can expand this by also including groups within groups, which you can see in the group Directorate, which contains the user Sebastian and the group Human Resources.

This shows that groups can be hierarchical, however they cannot be recursive – so in this case we could not have the group Directorate appearing in the Human Resources group.

When you initially setup Alfresco there are two predefined groups; **alfresco_administrators** and **email_contributors.**

Demo: Users

In this demonstration the creation of groups will be shown. Creating groups is undertaken in the admin console and is therefore a function only available to administrators.

First I will create the group US sales. A unique group identifier is given and a display name. I will add the users Heloise and Sebastian to this group. Demonstrating that groups can be hierarchical I will now create a group called Worldwide sales and add the US sales group to this as a subgroup. Something to highlight for database administrators and developers is that Alfresco prefixes group names in the database with GROUP_

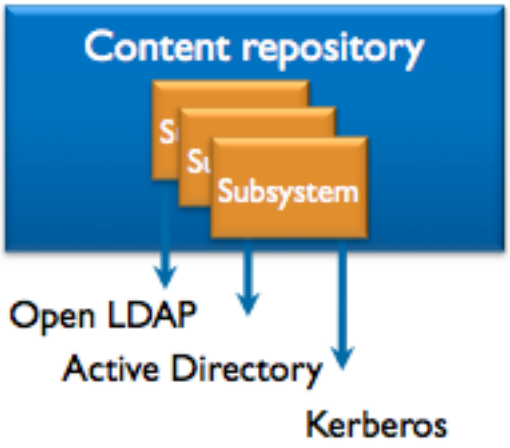
In the creation and management of groups the following information needs to be considered: Once a group has been created it is not possible to change its identifier. It is possible to change the display name. The Alfresco system also creates internal groups for its own use, alfresco_administrators and email_contributors are an example of this.

Authentication

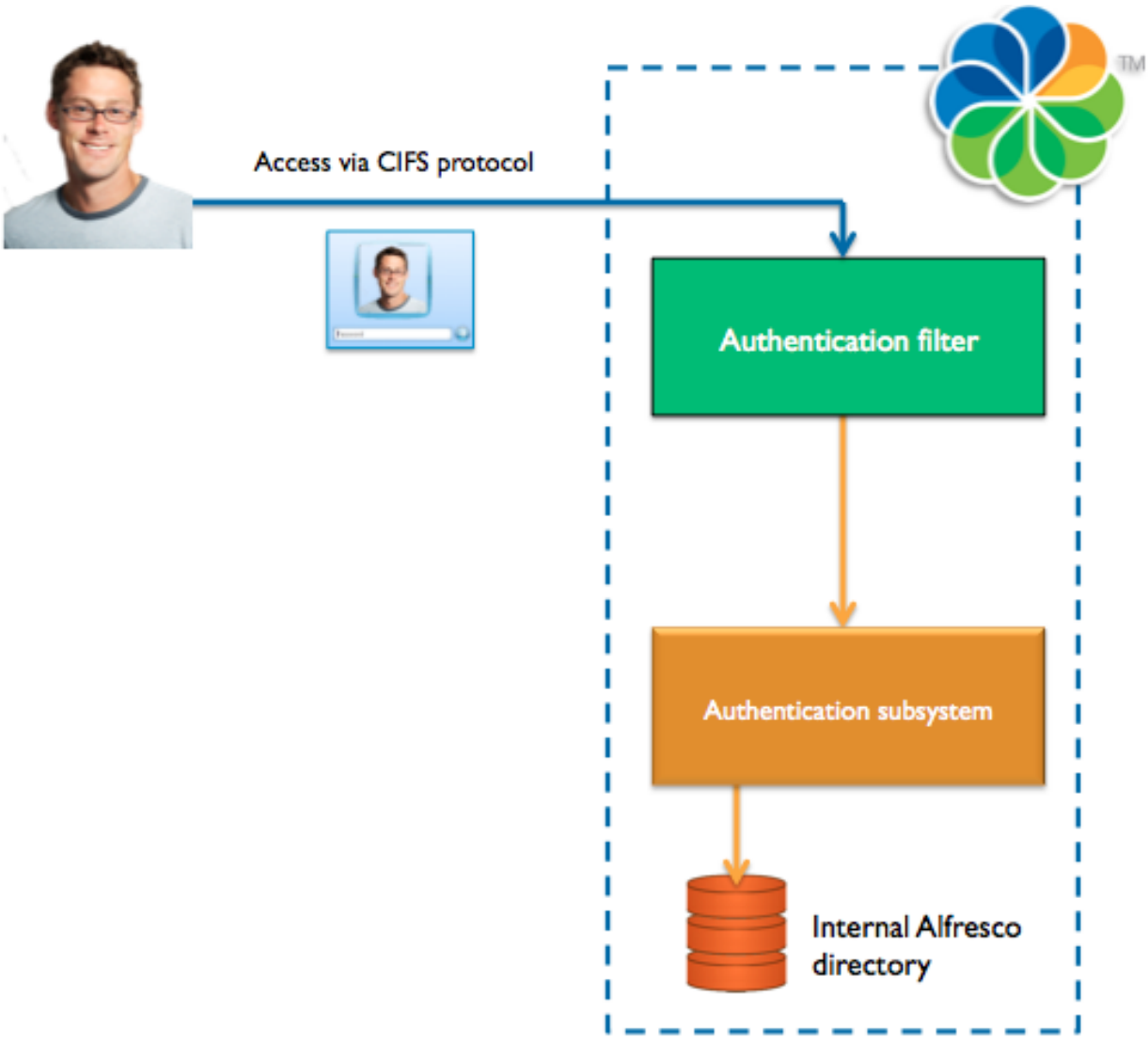
Alfresco supports a wide range of authentication mechanisms, as it installs out of the box it will be using the internal Alfresco authentication mechanism. As an administrator you have the option of configuring the system so that authentication responsibility is delegated to an external central directory server to remove the need to set up users manually in the administration console.

Alfresco authentication is provided by a set of configurable software modules called subsystems.

A number of alternative authentication subsystem types exist for the most commonly used authentication protocols. These include LDAP, Active Directory and authentication through a Kerberos realm.



Some of these also support single sign-on. Single sign-on enables automatic login using operating system credentials to remove the need for a login page. Additionally some of these can also be used to provide CIFS authentication.



User roles

When you are using Share for collaboration Alfresco groups permissions into roles. This allows for a set of permissions to be given easily to a user. A user may have multiple roles because a role is assigned to a user when they are invited to join a site in Share.

The records management module implements its own special roles within its site and we do not cover those in this course, if you are interested in Records Management we offer a special course for Records Management Administration.

The Alfresco system comes with four pre-defined roles, these are: consumer, contributor, collaborator and manager. These roles are used when users are invited to join a site. A user can only have one role per site, but as an administrator you can change a user's role on a site at any time. Whilst a user can only have one role per site, they may hold a different role in many sites.

Users have a default consumer role for any public sites for which they are not members, this is how public sites can be accessed, searched and viewed by people who are not site members.

The roles within Alfresco are hierarchical, at the lowest level there is the consumer, above the consumer is the contributor, a contributor has their own permissions but additionally because of the hierarchical nature of the roles, they also have the permissions for consumers. This hierarchical nature continues through collaborator and manager.

Matrix

The abilities a user can undertake within a Share site, for each specific site role can be seen here.

The user abilities translate to the following actions which are presented within Share. These actions target either a Share site folder or document library item. A detailed matrix can be found in the Alfresco online documentation, which we encourage you to examine now. This also

explores the actions it is possible to initiate when using Share site components, such as the wiki, blog and calendar.

Demo: Members dashlet

The easiest way to see who is a member of a site is by looking at the site colleagues dashlet. This typically appears on the site dashboard and works well for a site with up to a dozen members, once a site has more that this number of members this dashlet becomes unwieldy and as an administrator you may want to remove it from the site dashboard. For sites which have a large number of members the Members item on the banner works much better as it allows for searching for specific members. This also has additional functionality allowing you to change the user's role within a site.

Lab - Create users and groups

1. 1. Login to Alfresco Share as the administrator.
      - `user = admin, password = admin`
   2. Create the following Green Energy employees.
      - Heloise Dufresne
      - Jonathan Bradshaw
      - Luisa Rueda Salgado
      - Sebastian Koenig
   3. Create yourself as a user and add yourself to the administrators group.
   4. Create the `Business Development` site.
   5. Add the above users to this site with the roles listed below.
      - **Heloise**:Manager
      - **Jonathan**:Contributor
      - **Luisa**:Consumer
   6. Create the `Wordwide Sales` and `US Sales` groups.
      - Ensure `US Sales` is a subgroup of `Worldwide sales`.
      - Add Heloise and Sebastian to the US Sales group.
      - Additionally add Sebastian to the `Manufacturing` group, a group that already exists in the repository.
   7. Your optional stretch goal is to add an avatar picture to each of the above users.
      - `Desktop > Assets > Users and groups`

Users and Groups

Demo: Site members

The creation of users is a function of the administrator. From the More menu, the Users tool can be accessed. Here it is possible to create a new user.

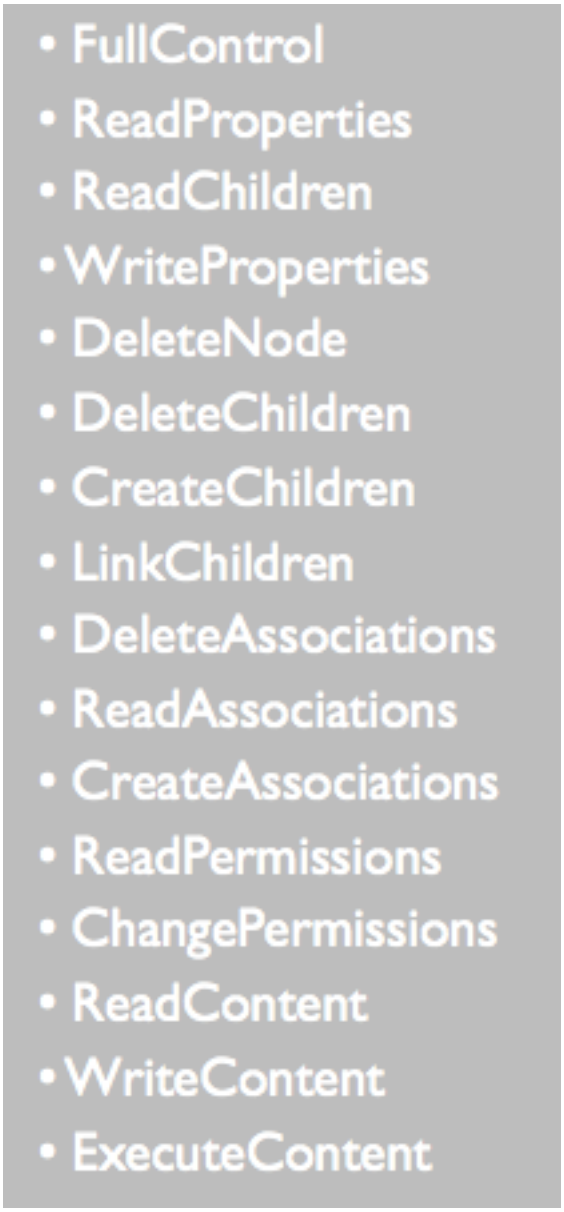
Any Alfresco user can create a Share site. To create a site use the My Sites dashlet found on the dashboard and use the Create Site link. By default a site will be visible to the public and non-moderated. There is only one site type in Alfresco at present, a collaboration site. When a Share site is created, it will be automatically displayed. The creation of users and Share sites is explored in detail in the Share for administrators Alfresco Element.

Permissions

Introduction

Alfresco provides a very sophisticated and flexible security model, in this section we look at this model from a high level perspective. In order to provide a smooth user experience the security model is simplified through the use of roles and permission groups, we look at how these work and how you can easily manage security based on these methods.

Alfresco defines a very basic set of permissions, some of which are shown here.



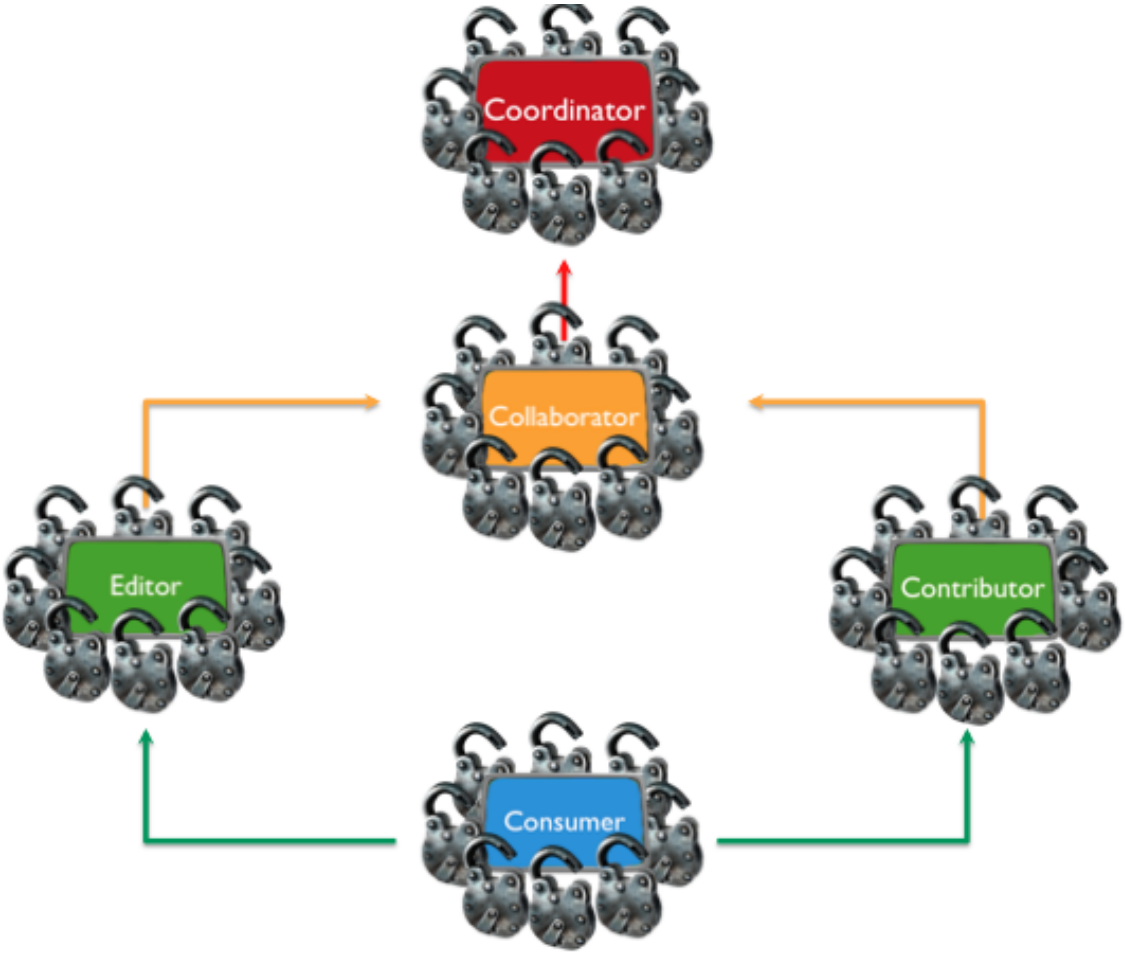
Some of these are applicable to all nodes in the repository and some are applicable only to nodes which have content. This wide range of permissions allows for very fine grain security levels, although security itself cannot be placed on anything other than a node. So for example you cannot have a read only node with a property that has write access.

Don't worry if this seems an excessively long list, you don't have to manage these permissions individually, it's just to show you the level of sophistication available in the Alfresco security model. Keep this in mind as we progress, since these low level permissions would form the basis of any customization of the Alfresco security model.

Permission groups

In order to make security more convenient and manageable, the basic permissions are bundled together into permission groups. Permission groups can be nested, however as an administrator you will not have to create or change the in-built permission groups. The Alfresco provided permission groups should be enough for the vast majority of situations. In practice we find that rarely is the Alfresco permission model customized.

The diagram shows that the lowest level permission group is consumer, with each group build up with more permissions until reaching the coordinator group.



Demo: Permissions

Permission groups dictate the actions, which a user can undertake in Alfresco. This presents itself as the actions, which can be executed against a repository item. Permissions also determine whether a folder within the repository is visible to an individual or group of individuals.

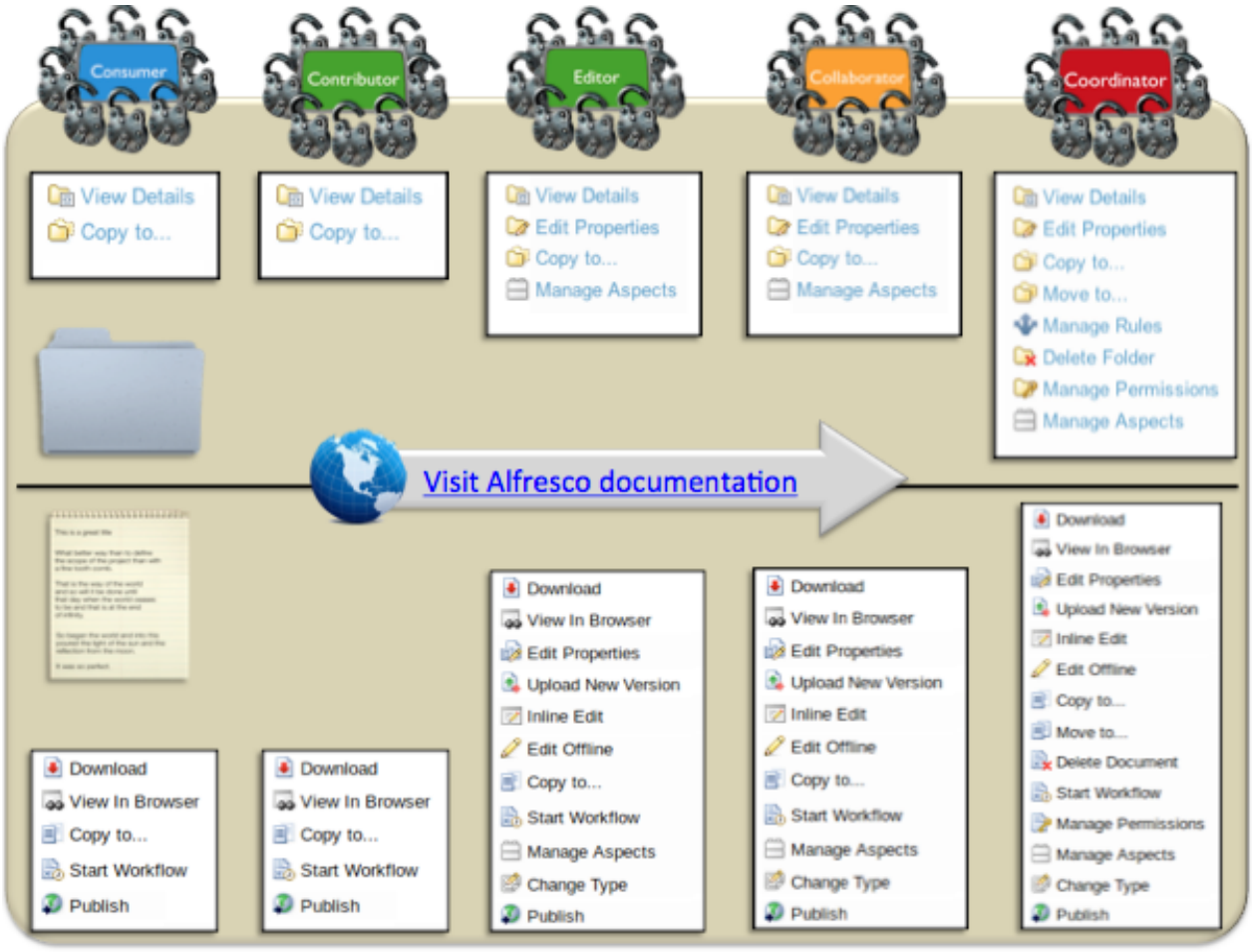
It is the administrator who has the ability to set permissions against a repository item and additionally grant the ability to a user to manager permissions themselves.

Permission group actions

Permission groups dictate the actions, which a user can undertake in Alfresco. This presents itself as the actions, which can be executed against a repository item. Permissions also determine whether a folder within the repository is visible to an individual or group of individuals.

It is the administrator who has the ability to set permissions against a repository item and additionally grant the ability to a user to manager permissions themselves.

The actions available to a user holding each specific permission group is shown here. These actions target either a repository folder or repository item.



Remember this is not an exhaustive list of abilities and there are subtle differences between the permission groups. For example a user with the consumer permission cannot create or add new content, whereas a contributor can.

A detailed matrix can be found in the Alfresco online documentation, which we encourage you to examine now.

Permission groups and share permissions

It is important to draw distinction between Permission groups and Share site roles.

In this Alfresco Element we discuss Permissions, Permission groups and their use within the repository. You have just explored the five permission groups and the functionality they bestow to a user. Permissions are set by the administrator.

Within a folder, permissions can also be managed by a user who holds the coordinator permission group, for that folder and its subfolders. The four Share site roles are given to a user and set by the site manager. This defines user abilities within that site. Permissions and Share site roles are both implemented through the individual underlying set of permissions.

Access control lists

An Access Control List (ACL) is an ordered list of Access Control Entries (ACEs).

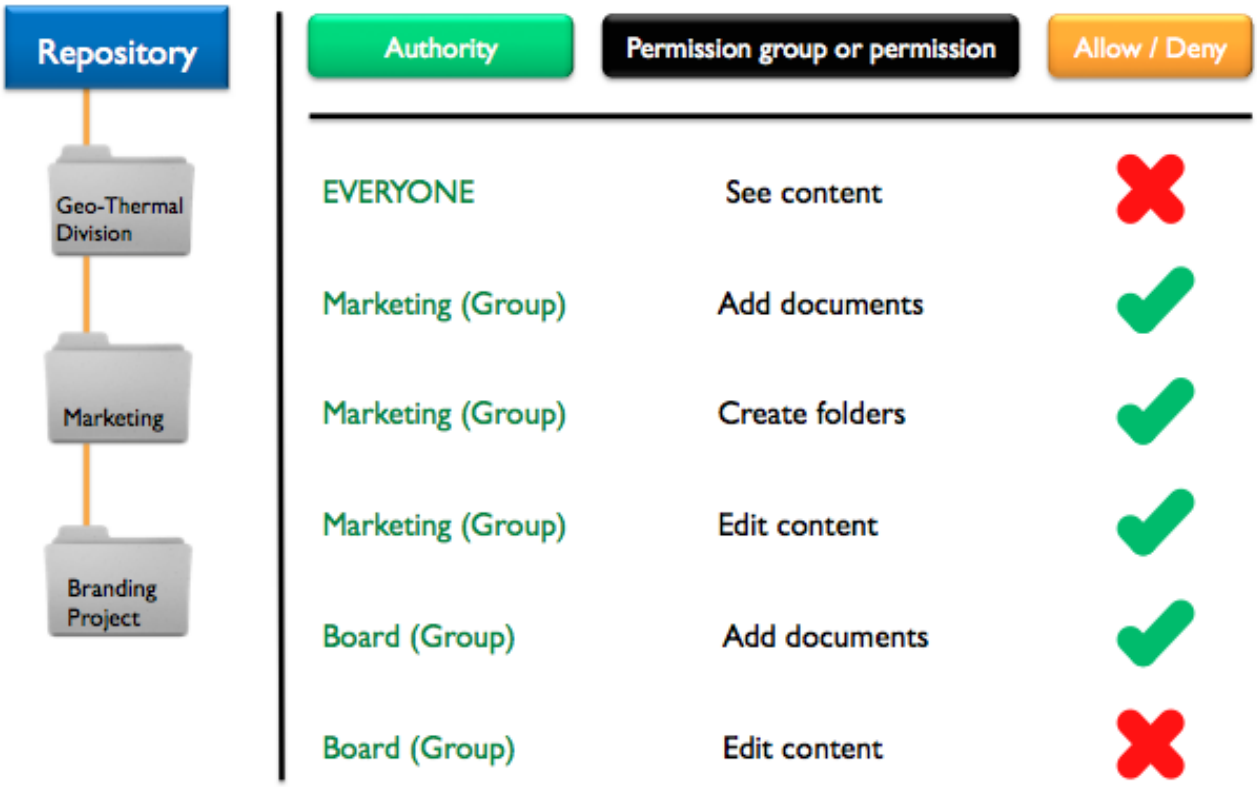
An ACE associates a single authority (a group, role or user) to a single permission group or permission, and states whether the permission is to be allowed or denied. All nodes have an

associated ACL. An ACL specifies if it should inherit ACEs from a parent ACL. When a new node is created it automatically inherits all ACEs defined on the parent within which it is created.

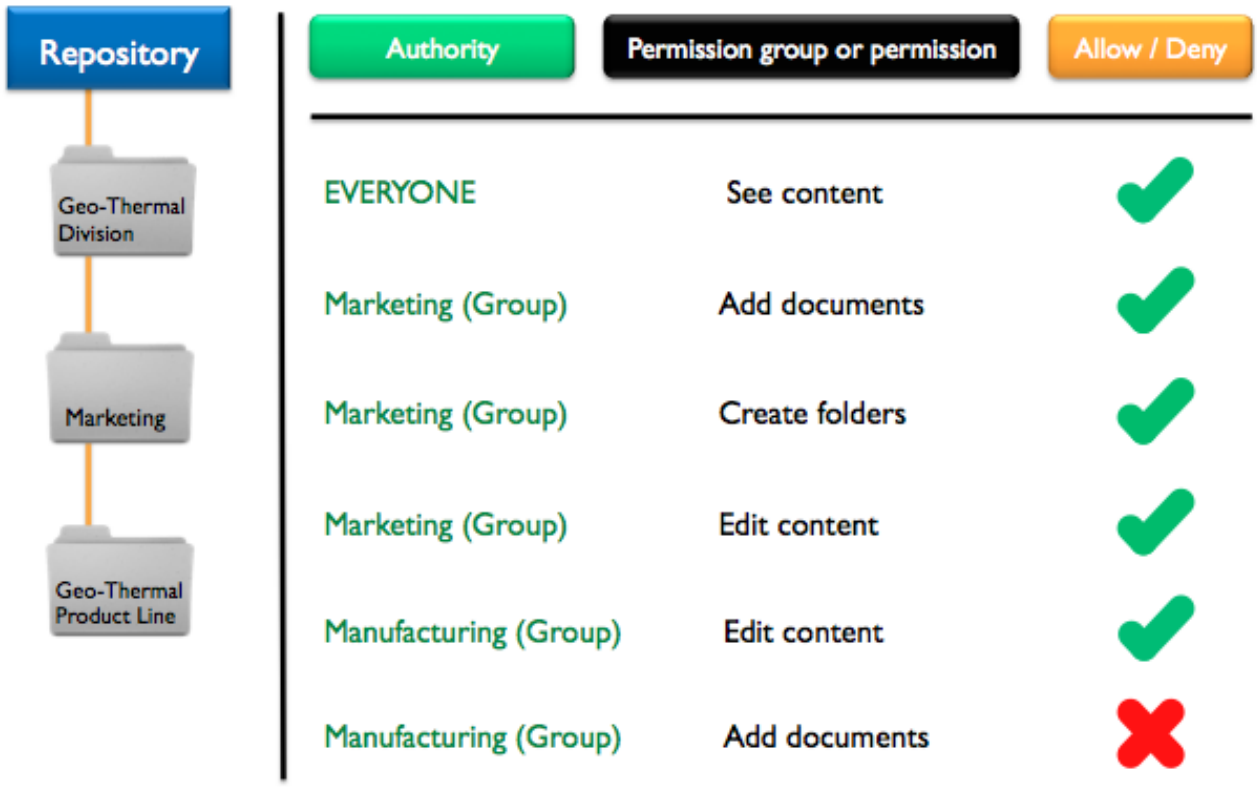
Green Energy scenario

In our business scenario we want to set up permissions in the Marketing area of our repository to match the on-going projects and business operations.

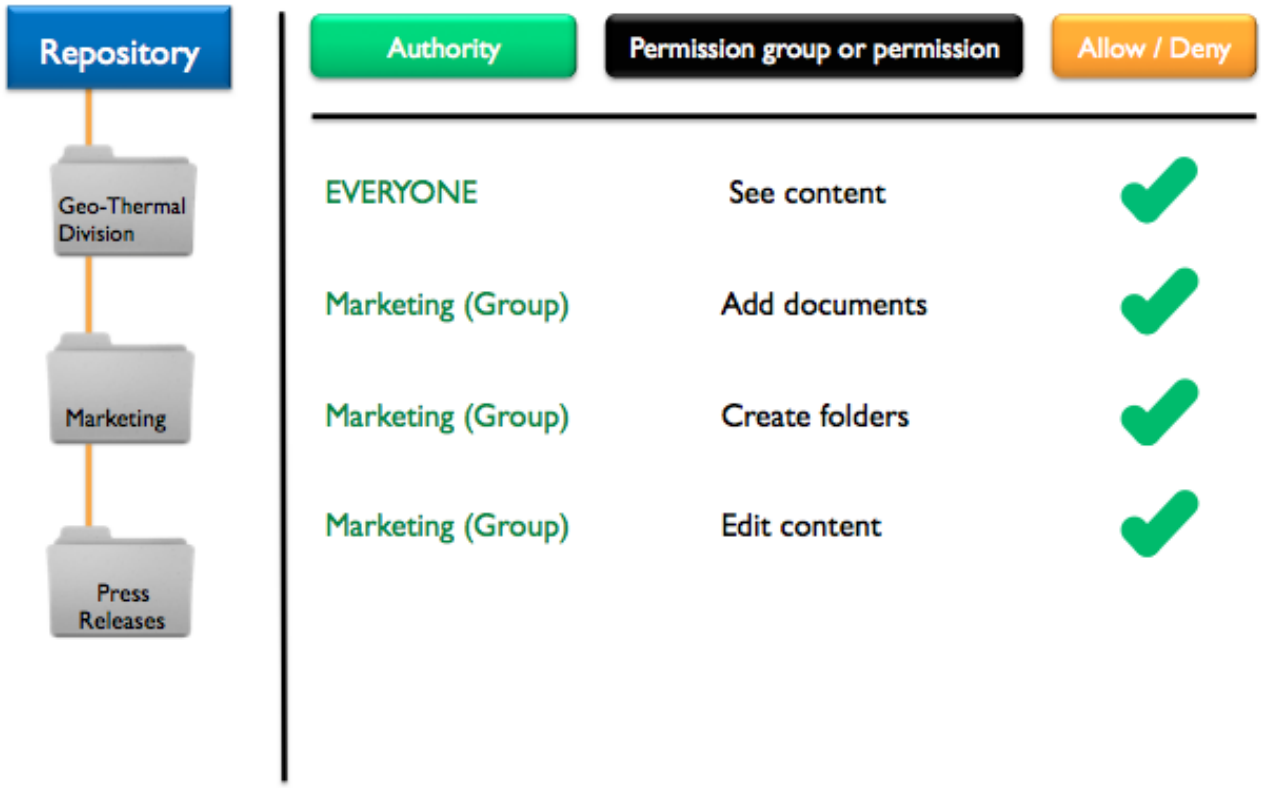
Green Energy is currently under-going a re-branding exercise led by the marketing department, with the active participation of the board. Other people should not see any of the work in here until the re-branding group is ready to release it. Marketing should be able to add documents create folders and edit any existing documents in the folder. The executive committee represented by the group; “board” should be able to add new documents, but not change any of the existing documents.



The Geo-Thermal Product Line folder is also managed by the marketing group. They add new documents here, but they want the manufacturing group to be able to correct and assist in the development of the marketing material for the product line, however they do not want manufacturing creating their own documents here. Everybody else in the organization should be able to see the contents of this folder.



Finally the folder Press Releases is viewable by everyone, but only marketing can add and change documents in this folder.



Permission groups

Demo: Permissions set up and users

In this video I will set the permissions for the three folders found in the repository - Geo-Thermal Division – Marketing folder, for the authorities Marketing, Board and EVERYONE as described in the previous slide.

Branding Project

For the Branding Project folder it was a requirement that no-one except the Marketing and Board groups can see this folder at this time. I will therefore turn off Inherit Permissions to remove the Consumer permission group from the EVERYONE authority. No one will be able to see this folder until I add new permissions. Marketing needed to be able to add documents, create folders and edit content. I'll add the Coordinator permission group to this authority. The Marketing group will have all possible permissions as if they were the owner of this folder. For the Board user group they needed to be able to add documents but not change existing documents. I will therefore assign the permission group Contributor to this authority.

Geo-Thermal Product Line

The Geo-Thermal Product Line folder is also managed by the Marketing group. I'll assign the Coordinator permission group. The Manufacturing group needed the ability to edit content but not create new content. I'll assign the Editor permission group to this authority. Everyone else should be able to see the contents of this folder so I will therefore leave the EVERYONE authority to the Consumer permission group.

Press Releases

For the final folder Press Releases Marketing will be Coordinators, EVERYONE will inherit the Consumer permission group.

Demo: Permissions testing

To show the effect of setting these permissions lets witness what specific users see when accessing the repository. Bill Dewi is Green Energy's Documentation Manager. He is neither a member of the Board, Marketing or Manufacturing. When he navigates to the repository - Geo-Thermal Division - Marketing folder he only sees the two folders Geo-Thermal Product Line and Press Releases as the EVERYONE authority has absolutely no permissions associated with the Branding Project folder. He can however enter either the Geo-Thermal Product Line or Press Releases folder and view or download content as a Consumer.

When Michael Ritter (Green Energy's Executive Chairman of the Board) logs in he is able to see the Branding Project folder as he is a member of the Board authority which has been assigned the Contributor permissions group. He is able to add content to the Branding Project folder, however he can only view existing content, not edit it. For the Geo-Thermal Product Line and Press Releases folders Michael presents as an EVERYONE authority and therefore assumes the Consumer permission group.

When Sebastian Koenig (Green Energy's Executive Vice President of Manufacturing) logs in he cannot see the Branding Project folder as only the Marketing and Board authorities have permissions against this folder. For the Geo-Thermal Product Line folder Sebastian can edit existing content as the Manufacturing authority has the Editor permission group assigned to this folder. He cannot however add content to this folder.

Finally Harriet Slim is Green Energy's Marketing Director. As such she is able to see all folders in this section of the repository. As a member of the Marketing authority she has the Coordinator permission group and therefore can perform all possible operations.

Access control lists additional

All objects in the Alfresco repository have a ACL. The ability to manage object permissions is available to the owner, administrator and those holding coordinator permissions. For all new objects created in the repository it will inherit an ACL from the folder where it is located. If the object is moved to a different folder it will inherit the ACL from the new parent folder. An object owner will always be able to see that object, it is not possible to hide this from its owner.

Lab - Create permissions

In this lab you will be establishing permissions to implement the business requirements of Green Energy, which govern their standard operating procedures.

1. 1. Login as the administrator and navigate within the repository to:
      - `Geo-Thermal Division > Manufacturing > Standard Operating Procedures`
      - `user = admin, password = admin`
   2. Ensure that the `Draft` folder is only visible to the `Manufacturing` group. (Manufacturing are responsible for creating new standard operating procedures hence this requirement.)
   3. Establish permissions for the In Review folder such when documents move into the folder they should be changeable by the following groups and visible to no one else.
      - `Manufacturing`
      - `Documentation`
      - `Quality Assurance`

      This requirement is necessary as we want the standard operating procedures to be reviewed and edited by these three groups.

4. Establish permissions for the `Effective` folder so that it is visible to everyone and its contents are editable only by the `Quality Assurance` group.
5. Establish permissions such that the `Superseded` folder is only visible to the `Quality Assurance` group.
6. Finally login with the following users to verify your settings:
    - `Matt Black – Member of Everyone (user: mblack, password: mblack)`
    - `Uschi Usha - Member of Manufacturing (user: uusha, password: uusha)`
    - `Bill Dewi – Member of Documentation (user: bdewi, password: bdewi)`
    - `Tom Klein – Member of Quality Assurance (user: tklein, password: tklein)`

Repository Configuration

Introduction

In this section we look at how repository configuration is undertaken, what can be configured and the different ways in which configuration changes can be made. The configuration bootstrapping (which controls and defines the Alfresco server startup) is carried out in a particular order and this ordering and its implications are explored. Finally we look at advanced configuration topics and best practices.

What can be configured

Almost anything which can be configured in Alfresco can be configured through properties files. Alfresco also holds configuration information in Spring configuration files, these are typically the province of developers rather than administrators. In some rare cases you may need to edit Spring XML files.

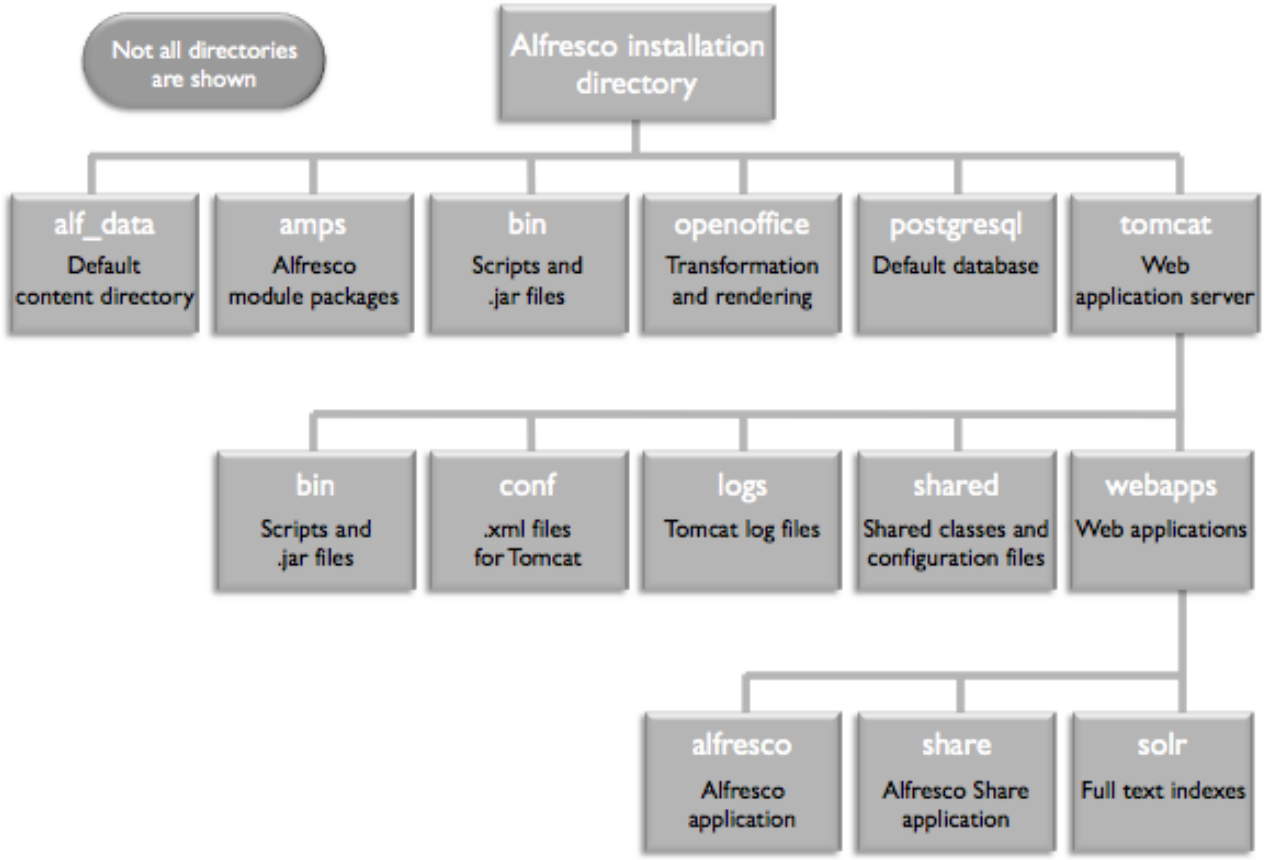
Version 4 of Alfresco introduced new functionality to the Share administrator's console, enabling a range of configuration settings to be made through that user interface.

Within the various property files much can be configured by simply editing the file, this includes: the details for connecting to the underlying database, the location of the content files, how authentication is to be performed, how Alfresco interacts with email, jobs and tasks which should be run at regular intervals and of course the content model of the repository.

The content model is a special case of configuration and is an area normally reserved for developers. This is something which is often customized within Alfresco so it is important to understand the process of deploying content models and how this configures the Alfresco repository.

When you install Alfresco using the installation wizard this information is configured for you based on the values and answers you provide during installation. However you may want to change some of the configured values at a later stage or add configuration parameters not supplied during installation.

Installation file structure



This diagram depicts an Alfresco installation. This is the file structure as created by the installer.

In production systems it is unlikely you would use the installer to deploy Alfresco, components are typically installed manually, for example directly to the web application server. Additionally remember that individual components can be split out to their own tier or server and therefore this is only a general representation.

Server configuration files

When the Alfresco server starts it first loads the web application file web.xml, this in turn loads Spring configuration and property files which launch the Alfresco server.

Key Spring configuration files which control the startup process include; `web-application-context.xml, application-context.xml, application-context-core.xml and core-services-context.xml`

When the Alfresco subsystems initiate they may load their own configuration properties.

The last configuration file to load is `alfresco-global.properties`

When choosing to configure Alfresco by editing property files, the Alfresco global properties file is your first port of call. As this is last configuration file to be loaded, it can be used to override any value that appears in any previously loaded property file.

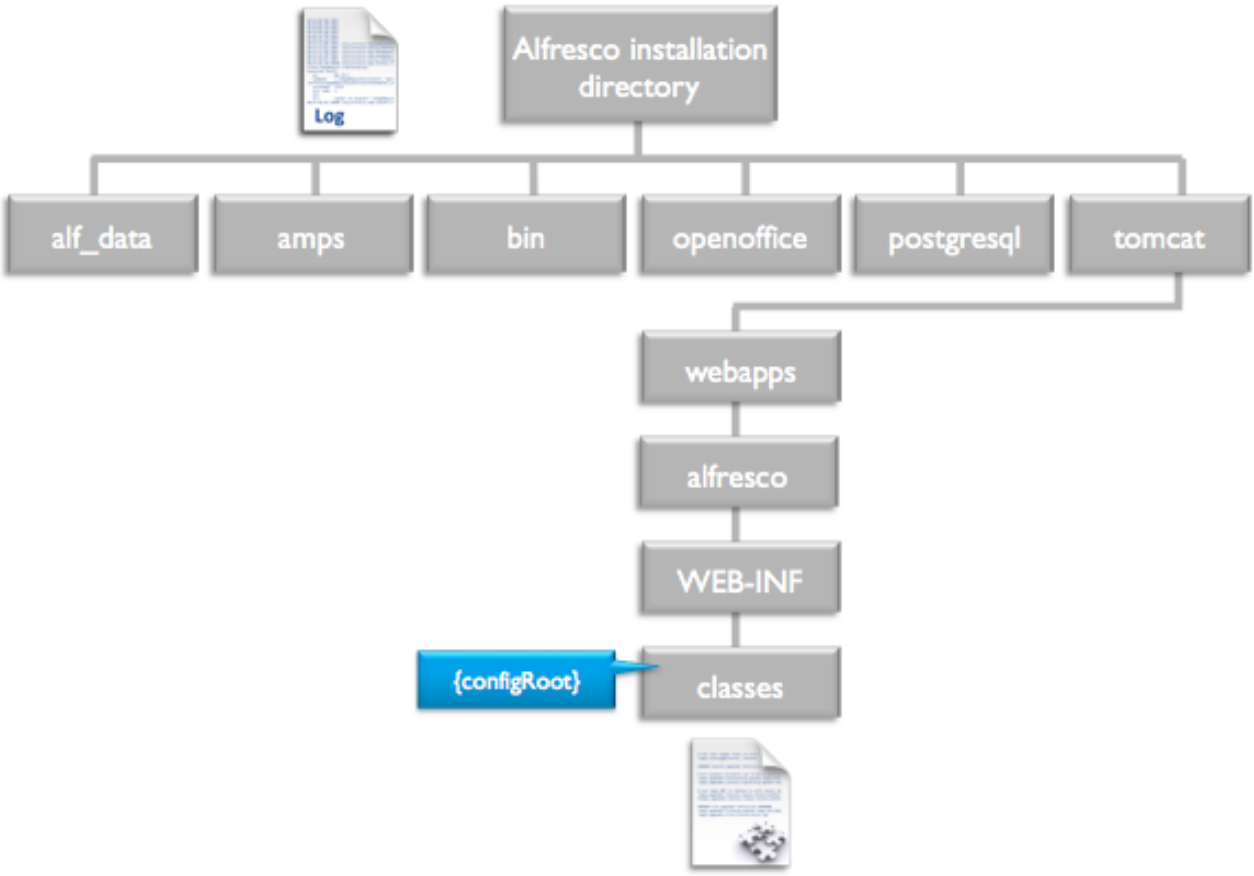
By default the files are located where indicated. In practice the system searches the `classPath` for the files discussed.

Do bear in mind, that due to the modular nature of the Alfresco architecture specific components of the solution may be deployed to individual tiers or severs and therefore the exact startup process may vary.

Logging

Logging within Alfresco is facilitated through the open source logging tool, Apache log4j. This allows the developer or the administrator, to control which log statements are output with arbitrary granularity. It is fully configurable at runtime using external property files. It is important to have some understanding of how this system works because at some point you will want to debug your application or troubleshoot your production Alfresco installation.

The file which controls the log4j logging is called `log4j.properties` and can be found in `<TOMCAT_HOME>/webapps/alfresco/WEB-INF/classes`, a directory known as **configRoot**.



In this file the logging is set up to produce a file called `alfresco.log` which is written to the `tomcat` directory. The properties file also specifies that each a new `alfresco.log` will be started and the previous log file will have the date appended.

If you would like to explore log4j further you can use these links.

`http://logging.apache.org/log4j/`

`http://en.wikipedia.org/wiki/Log4j`

Lab - Alfresco log file

1. In this lab you are going to change the location of the Alfresco log file.
    1. Find the `log4j.properties` file in `/opt/tomcat/webapps/alfresco/WEB-INF/classes`
    2. Copy the `log4j.properties` file to `/opt/tomcat/shared/classes/alfresco/extension`

3. Rename the `log4j.properties` file to `production-log4j.properties`
4. Edit the `production-log4j.properties` file

   Find the line starting `log4j.appender.File.File`

   and change it to

   `log4j.appender.File.File=/opt/alfresco/alf_data/logs/alfresco.log`

   Save the file.
5. Open the `Terminal` application (from the `Application > Accessories` menu) and enter the following commands to create a new logs directory and to set the correct permissions for that directory.

   ```
   cd /opt/alfresco/alf_data
   ```

   ```
   sudo mkdir logs
   ```

   (Enter `alfresco` when prompted for the password)

   ```
   sudo chown tomcat6.tomcat6 logs
   ```
6. Restart the Alfresco server.
7. Verify that Alfresco has created the `alfresco.log` file in the new location.

Repository Configuration

Demo: alfresco-global.properties

In this video the Alfresco global properties file will be examined. As has been described this is the last configuration file, which is loaded and this allows you to override previous settings made during the server startup.

The Alfresco global properties file is found in `{extensionRoot}`. This is `TOMCAT_HOME/shared/classes`

Naturally this path is the same on other platforms, such a Linux.

Let's take a look at the content of this file.

The first setting to note is `dir root`. This sets the location of the file system content and indexes, which by default is `alf_data`. Being a variable this enables you to change the `alf_data` location to an alternate directory or machine.

The database connection properties can be found next. This enables the database name, login name, password, host and port number to be set. In a production environment either you or the database administrator would with all certainty change the access settings and very likely the location of the Alfresco database.

The email settings can be found lower down the file.

For user authentication the Alfresco system operates by default. The Alfresco global properties file enables you to leverage an alternate authentication system.

Authentication is examined in detail in the Alfresco Authentication Element.

Demo: Share invites don’t fail – SMTP settings

The Share for administrators Alfresco Element demonstrated the process of a site manager inviting another to join the site. This invitation was sent to the individual through Share and displayed in their My Tasks dashlet.

Share is configured to also send the invitation by email. This is enabled by default and since the outbound SMTP server settings are not yet configured this generates errors, which can be seen

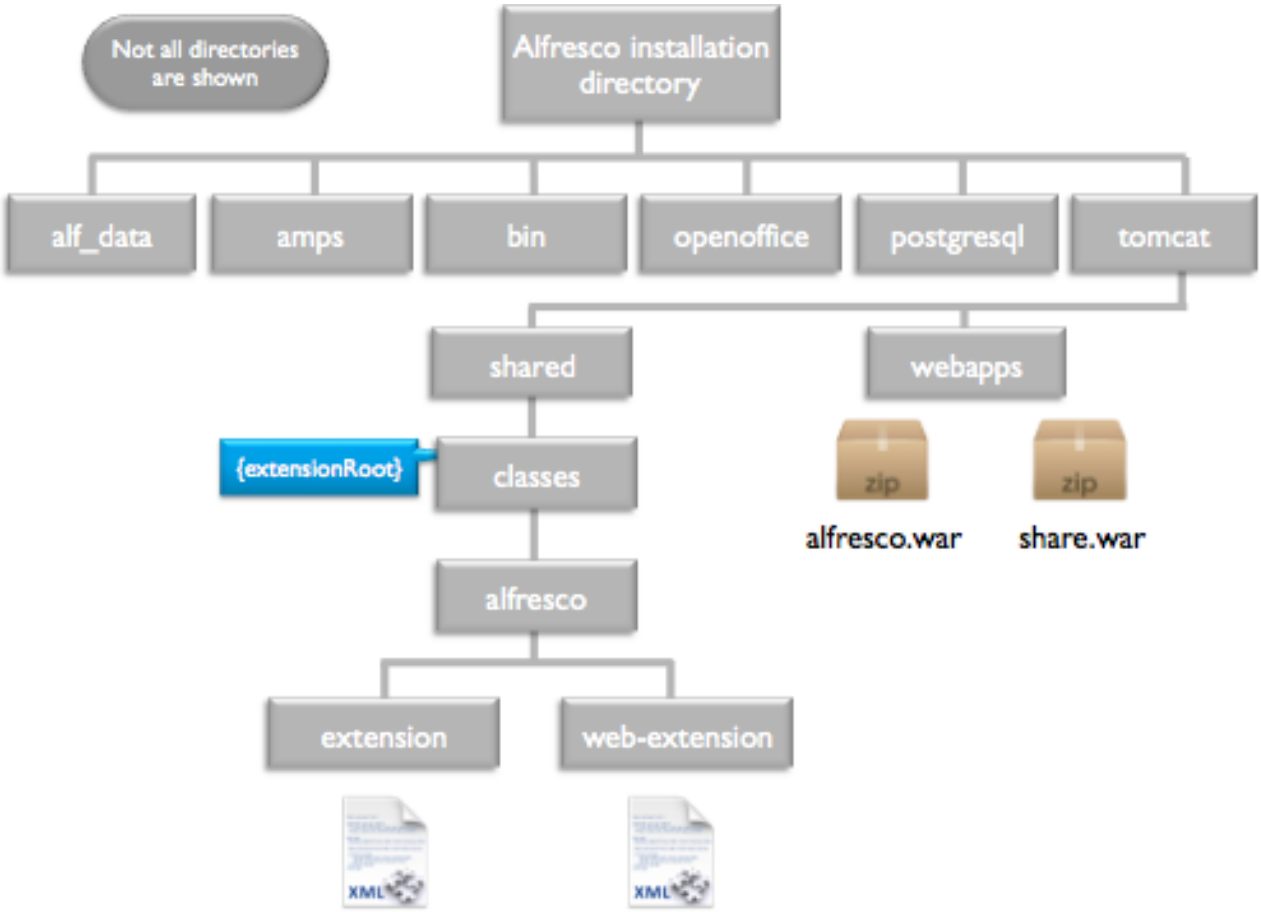
in the `alfresco.log` file. Once the outbound SMTP settings are made these errors will no longer be generated.

Should you wish to disable Share from sending email invitations the following line can be added to the Alfresco global properties file. This will come into effect upon the next server restart.

```
notification.email.siteinvite=false
```

Developing extensions

You can use several methods to extend Alfresco with new functionality and change the behavior of existing functionality beyond those exposed through properties files. The recommended way is to use the provided extension mechanism, which takes the form of the folders `{extensionRoot}/alfresco/extension and web-extension`

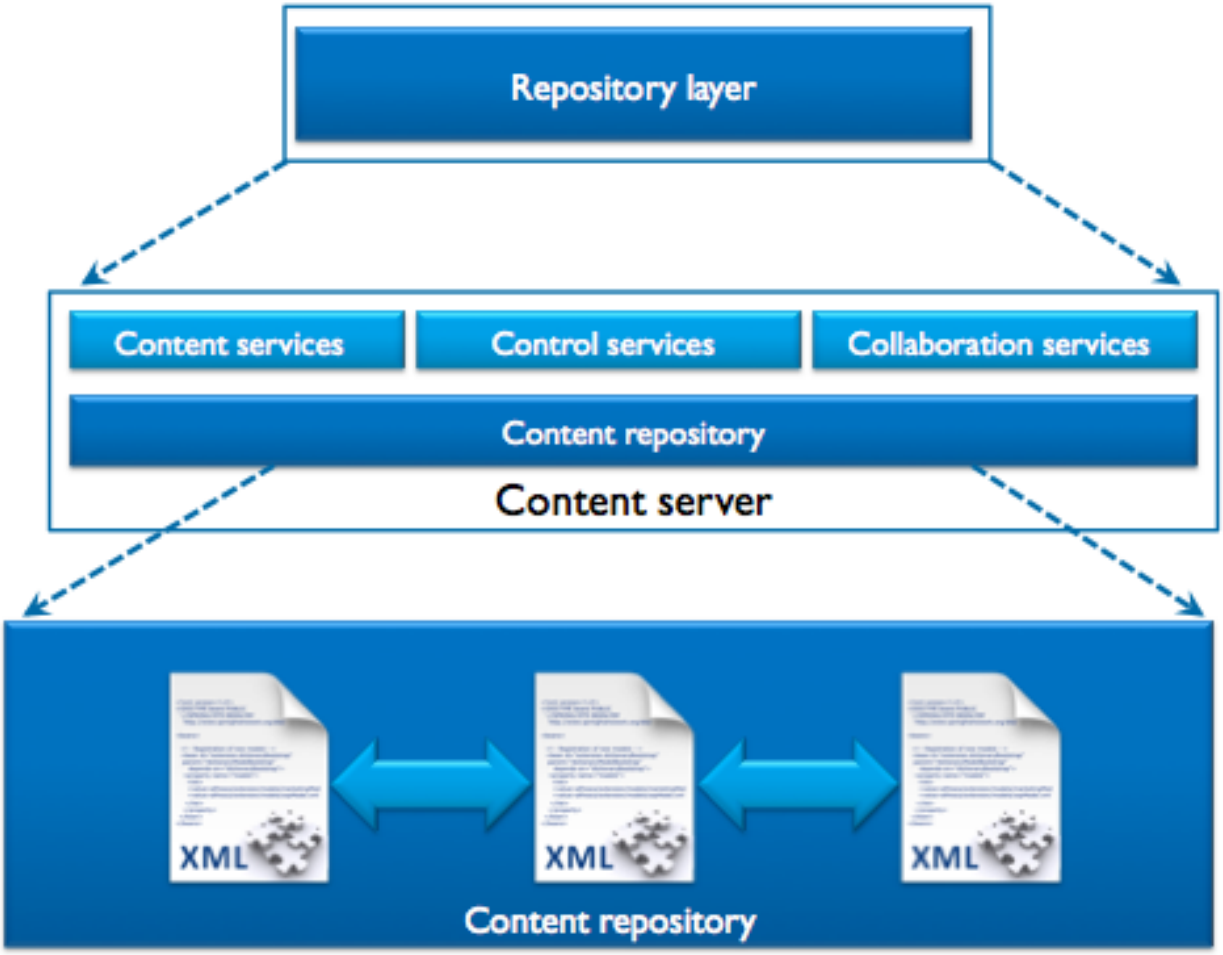


The advantage of this is that the extension files are outside the standard Alfresco web application, so the out-of-the-box Alfresco files do not need to be modified, making it upgrade friendly. This means that you can install newer `alfresco.war` and `share.war` files and the same extensions still function without changes or any further action from the administrator.

Deploying content models

Content models are a fundamental building block of the Alfresco content repository. A content model is defined in its entirety as a single XML document, each model contains a set of related and coherent definitions, and is deployed as a unit.

Several content models may be deployed to the content repository and definitions in one content model may depend on definitions in another content model, allowing for the sharing of definitions.



There are two approaches to deploying a content model to the Alfresco repository: bootstrap and dynamic.

The bootstrap approach involves creating your content model as an XML configuration file and placing this in the `extension` folder. When the Alfresco server starts it is read, validated and registered. It is recommended that you create a unique folder for your content model. This approach has long development cycles and is suited to finalized content models ready for production environments.

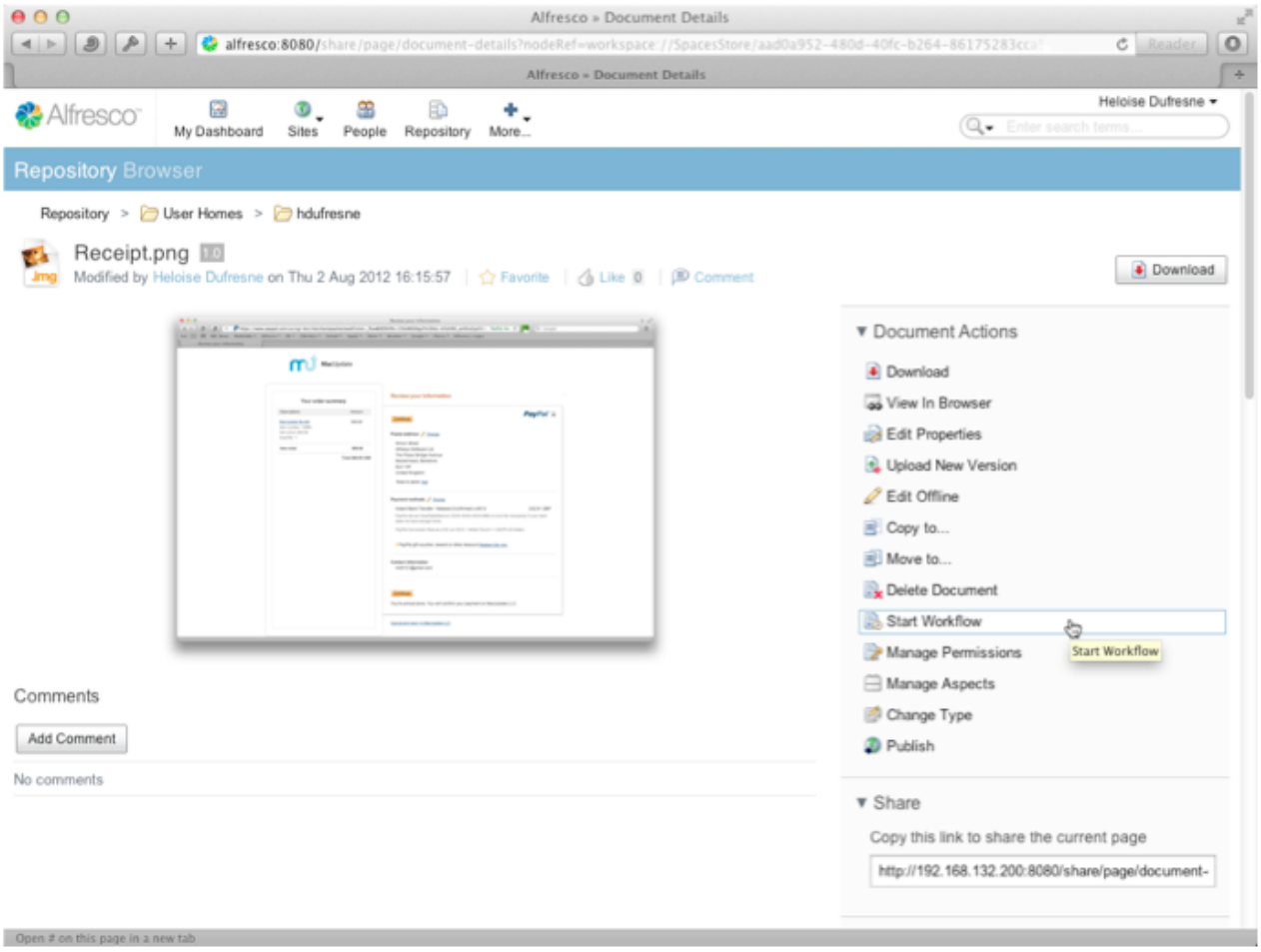
With dynamic deployment Alfresco Share is used to register and install a content model whilst the Alfresco server is running. This is helpful when developing a content model. There are some restrictions when using dynamic deployment, for example, only incremental changes to the model are possible and a content model cannot be deleted if a content item of this type exists in the repository.

Content models and their deployment are explored in detail in the Alfresco Elements Content Model Overview and Creating Content Models.

Alfresco Activiti workflow

Alfresco implements workflow through Activiti. This is an independent, open source, process management platform allowing business processes to be modeled and implemented within Alfresco.

A content item within the repository can be assigned to a workflow through Alfresco Share. The initiated workflow can have many steps and branches and is progressed through review and input from a user or group. Workflows are managed through the Activiti Workflow Console and are developed in XML or through the workflow modeller in Eclipse.



Workflows can be deployed using the bootstrap process (preferable for production systems) or dynamically using the Activiti Workflow Console.

The Training course Alfresco Activiti Workflow examines this topic in greater detail.

Advanced configuration

Alfresco can be configured in a number of advanced ways to support some sophisticated enterprise requirements, for example Alfresco can be configured to support content lifecycle management, that is moving content between different physical stores through its lifecycle, so that content can be stored on the most appropriate and cost effective medium.

For high availability Alfresco can be clustered and also supports replication (a topic covered in the Advanced Systems Administrations course).

Alfresco runs a number of administrative jobs on a scheduled basis and to support this implements the Quartz job scheduler, the frequency of these jobs can be modified and indeed new scheduled jobs can be added to the system as required.

JMX console

Within this Alfresco Element you have seen that the functionality and behavior of the Alfresco system can be manipulated and enhanced through XML (deployed either through the bootstrap process or dynamically) and by editing property files, which are read at startup.

There is a further way to enact changes within the system. This is achieved by editing property values directly through a Java Management Extension (JMX) console, whilst the Alfresco server is running. Such changes are dynamic and come into effect immediately. This functionality is explored with the Alfresco Element Managing the Repository.

Best practise

With all of these options available it is worth considering some best practices in configuration. First and foremost ensure that you have both a repository backup and a backup of any of the files you propose to change, by taking a separate backup of your configuration files before making changes.

Make sure that you have tested your changes on a non-production system before deploying to a live production environment. Changes to configuration files may potentially cause the Alfresco server not to start or to behave in unexpected ways. You may want to consider using a configuration management system such as Subversion, or Perforce, to store and version control changes to your environment. Either can be integrated with Eclipse for a better user experience.

Managing the repository

Introduction

Managing the repository consists of undertaking tasks to ensure you have a reliable and well performing repository, as well as managing ongoing tasks which may change the repository or apply adjustments.

The regular ongoing tasks include monitoring health and usage of the repository and preventative maintenance. At other times you may need to install applications and change the times and frequency of scheduled jobs.

Alfresco provides a very powerful way to add rules to a folder to provide creative solutions for automating and managing your content, we will look at what you might use such rules for and how you can set these rules up as administrator.

AMP files

For production applications the recommended way of deploying applications is to use the Alfresco module package (AMP). An AMP file is a collection of code, XML, images and CSS that collectively extend the functionality or data provided by the repository.

It may be fine to deploy a single small extension manually, however once you have more than one extension or a complex extension then installation will be challenging.

An AMP file is essentially a ZIP file with a pre-determined internal folder structure and, at its minimum, a single required configuration file. An AMP file is applied to a WAR file, which is then redeployed to the application server.

The `alfresco` and `share` WAR files contain the core Alfresco product and the Alfresco Share user interface. These are held in WAR files which are expanded and deployed during the server startup.

AMP files which are applied to these WAR files are naturally distributed at this time.

The extension will therefore appear as part of the Alfresco web application; for example, all the files are within `~tomcat/webapps/alfresco` or `~tomcat/webapps/share`. The repository also maintains a registry of the installed modules and their version enabling the modules to be upgraded independently of the WAR file.

There are some disadvantages though, since this means the files are all loaded by the application server's main classloader, you lose the ability to reload configuration files dynamically. In addition, since the AMP construction, WAR integration, and deployment are not conducive to a streamlined development cycle, you should only use AMP files for completed extensions.

Demo: AMP deployment

The Alfresco Records Management software is distributed as AMP files, let’s explore the installation to demonstrate the process of installing an AMP.

The software comes in two components;

`alfresco-rm-2.0.1-147.amp`

`alfresco-rm-share-2.0.1-147.amp`

The first file contains the additional functionality that is applied to the alfresco installation. The second file contains the software to enhance the Share user interface.

If you view the contents of these files with a zip utility you can see the folder structure.

I move the first file to the `amps` directory.

The second file I place in the `amps_share` directory.

I stop the Alfresco server.

Next I use the `apply_amps` command, which is found in the `bin` directory. This command applies the AMP files that are located in the `amps` and `amps_share` directories.

I start the Alfresco server, then login as the administrator.

I can now add the Records Managements functionality to the dashboard.

Returning to the original AMP file I again explore the folder structure. The `apply_amps` command has copied the content into the `alfresco.war` file. When the server starts the `alfresco.war` file is expanded and deployed into the `~tomcat/webapps` directory, maintaining the same folder structure.

The same is true for the Share components from the `alfresco-rm-share` AMP file, now found in the `share.war` file and similarly deployed.

This deployment method, commands and paths are the same on alternate platforms.

Scheduled jobs

Alfresco runs a number of inbuilt jobs which perform background processing, the open source Quartz Scheduler is used to perform the scheduling function. The scheduler runs a number of jobs from various content cleanup jobs, through full-text indexing and feed automation jobs. Some of the additional modules such as Web Content Services and Records Management also implement their own periodic jobs to perform specialized functions.

The frequency of jobs is optimized for performance in most situations but can be altered. This requires some thought and understanding to ensure that the system behaves as expected and is therefore considered an advanced system administration topic.

A healthy repository

Your regular preventative maintenance routine should start with repository monitoring. A healthy repository is characterized by a number of different factors; no errors in the log files, sufficient free disk space, a range of high quality backups, little or no fragmentation on disk volumes and a fast performing database. If all of these factors are correct then you should enjoy a stress-free experience which provides good performance for users. Performance is obviously subjective, but in a healthy repository response times should be good and you shouldn't have complaints from users.

This list provides you with guidance of the areas to look at on a regular basis.

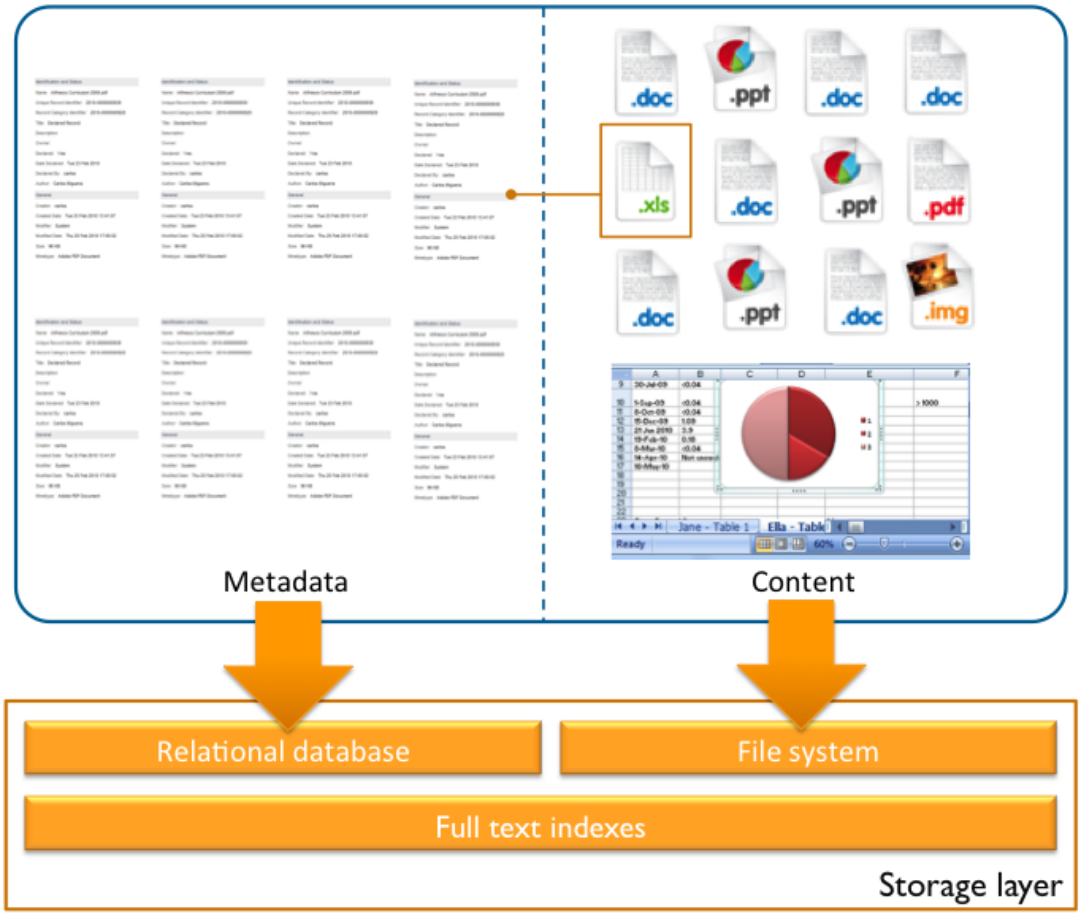
- Check log files
- Check disk space
- Check backups
- The state of the full text indexes
- Remove orphaned content
- Database optimization

How regularly you look at these depends on a number of factors such as the usage patterns of your repository, the service level objective you have for your system and the transaction throughput. If your production system is heavily utilized and you work in a well structured IT department you may use various tools to make this job easier, for instance you may integrate the Alfresco log file into a complete log monitoring service for the organization so that you are notified of any log files errors which occur through a notification service, like email, that means you don't have to physically inspect the log file.

File storage method

Within the Alfresco storage layer content is stored on the file system and content metadata is stored within the database. This is an efficient architecture which delivers optimal performance.

When a document is deleted from the system only its metadata is removed, the content file is left as orphaned content. To deal with this Alfresco implements a scheduled job to clean-up these files, part of you routine maintenance should ensure that this job is running at the right intervals and executing successfully.



Since Alfresco relies so heavily on the underlying database it is of utmost importance that this is performing at peak efficiency. All of the database systems which Alfresco works with have ways of tuning and optimizing performance for different scenarios. It is important that you involve a database administrator early to setup a regime of database maintenance and optimization which suits your usage profile.

Monitoring usage

Monitoring usage is best achieved through a Java Management Extension (JMX) console, (this must support JMX remoting). There are a number of JMX consoles available, for example Jmanage, MC4J and JConsole. We use JConsole in our examples as this ships automatically with Java standard edition 5 onwards. The JMX interface is only available with the Alfresco Enterprise edition.

Demo: JMX console

Let's explore the use of the JConsole Java Management Extension with Alfresco.

JConsole is invoked through the terminal, on the Windows platform it is usually found in the `bin` directory of the Java install.

As indicated the Java Management Extension must support remoting. I connect to the Alfresco repository using the remote process connection, using the following string.

service:jmx:rmi://ignored/jndi/rmi://127.0.0.1:50500/alfresco/jmxrmi

This string is attached to this presentation, so you can download and use it within an upcoming lab.

The default username is controlRole and password change_asap (for productions system you obviously need to change this !)

There is a large array of information that can be seen using a JConsole, for example the number of users and groups in the system, the number of connections defined and used, the total size of content currently stored, the size of the indexes, the transformers available and the patches applied to the system.

This information exceeds well over 100 discrete items of information. Should you ever be in contact with Alfresco Support they will often ask for a JMX dump as this provides a snapshot of the current system parameters and its status (this is achieved through the administrator's panel of Alfresco Share).

Not only is it possible to view system information, it is possible to alter and adjust the live system by editing parameters through JConsole.

For example you can enable or disable file servers such as CIFS.

Whenever you make a change to a subsystem value the subsystem will be stopped and you must restart it if you want the service to be enacted. Such a change is synchronously actioned and does not require you to restart the server.

Subsystems (such as authentication, replication, email, file servers) are a good use case for a JMX console, where you can build complex configurations without endless restarts.

The revert method will revert the property value to that held in the `alfresco-global.properties` file or in its absence the default value.

Most property edits are persisted in the database and will be remembered on server restarts. Some property changes, such as the logging level, take their value from their respective configuration file upon startup and are not persisted, these are principally the systems outside of the core alfresco system.

There may be cases in a production situation where you would wish to disable JMX entirely, you can do this by editing the core-service-context.xml file and commenting out the `RMIRegistryFactoryBean` section. This change will take effect at the next server restart.

JMX edits vs global properties

As discussed in the Repository configuration Alfresco Element the `alfresco-global.properties` file is loaded last. Any property value held in that file overrides the value from a previously loaded configuration file.

In the case where you edit system values with a JMX console these will persist through server restarts and settings in the `alfresco-global.properties` file will be overridden.

It is therefore good practise to copy any parameters edited through a JMX console to the `alfresco-global.properties` file.

JMX and clustering

Using a JMX console when you have a clustered environment means that property changes are made once and across the whole environment. If you were to access a specific machine and edit a property file directly then this could lead to consistency issues depending on your configuration.

JMX

Whilst the JMX console is exceptionally useful it only provides a real-time view of the system. If you are interested in historical trends and patterns, such as document growth, you will need to do some additional work such as configuring the audit trail to capture the information you are interested in.

You can learn more about the JMX interface from the Alfresco documentation online.

`http://docs.alfresco.com`

Lab - JMX console

1. In this lab you will use JConsole to access the Alfresco server and make property value changes. Your tasks are:
    1. Start JConsole and connect via the remote process.

       (The connection string and account details are attached to this presentation.)
    2. Disable inbound email.

       You will find this property in `Alfresco > Configuration > email > inbound > Attributes > email.inbound.enabled`

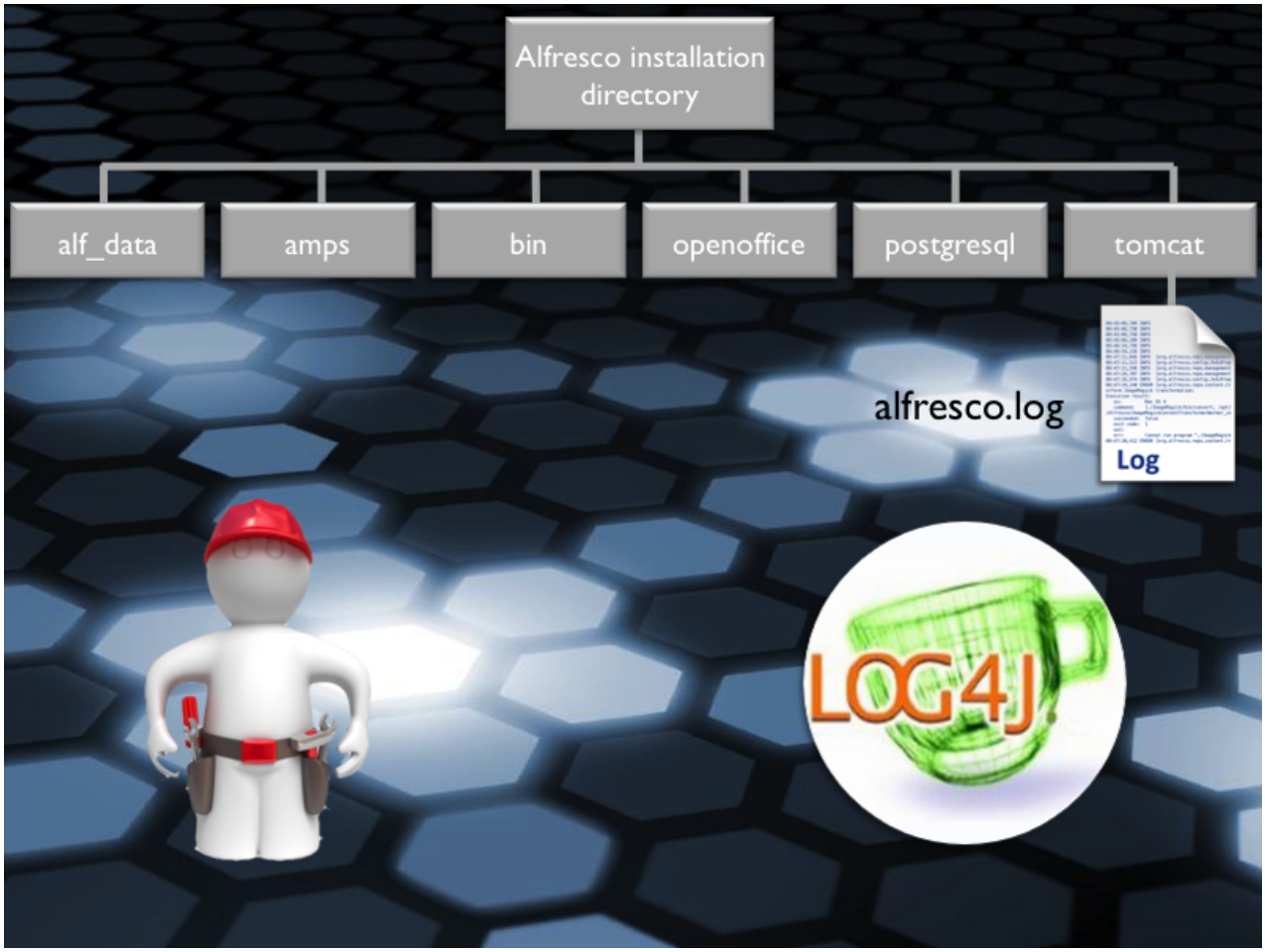
       Remember to restart this subsystem using the Operations.
    3. Restart the Alfresco server.
    4. Check that the change has persisted.
    5. Revert this change.

Managing the repository

Logging

Alfresco records server activity and errors within the `alfresco.log` file.

The logging system used is Log4j which is highly configurable and provides varying levels of error reporting allowing you to also use the log file for debugging and troubleshooting.



Although the alfresco.log file is your first port of call, Alfresco does not operate in isolation and other parts of the system may have an impact on Alfresco, hence there are other log files which you should also monitor for errors. The primary ones are the log file for your database and the log file for your application server. You will also want to visit the operating system logs as these may provide evidence of low level problems which affect higher level systems such as Alfresco.

The logs for the operating system, application server and database will vary from system to system, so you need to check the documentation for your particular stack. On some systems there may be an interactive monitoring tool for example the one shown here for MySQL. For Alfresco however the items you find in the `alfresco.log` file will come in a consistent pattern.

A good tool for sending you an email to periodically highlight errors that have occurred is `tail_n_mail: http://bucardo.org/wiki/Tail_n_mail`

This is a Perl script and is therefore supported on multiple platforms.

Storage space

Most of us use our hard disks like closets, stuffing in files and then forgetting about them. But no matter how big a disk you have, it's going to run out of free space one day, and running out of disk space when the CEO is trying to save his or her imminent presentation could hurt you badly. Keeping an eye on disk usage doesn't take much time or effort, here are some tips and tools. Alfresco does not provide a tool for monitoring disk space, there are many excellent tools already available to perform this task, you need to choose one that is appropriate to your operating system and infrastructure. The main rule when monitoring disk space is to ensure that the content location has at least 20% free. You may want to enable user quotas inside Alfresco, but this very much depends on your organization's policy and this can only ever be a blunt instrument in an enterprise content management system like Alfresco. Whilst you store any type of content inside Alfresco there are some content types which you should consider not allowing inside your repository, mainly these are executable files and databases, such as Access databases.

Demo: User quotas

Alfresco has the ability to track the cumulative size of content added by each user of the system. This is typically displayed in kilobytes, megabytes or gigabytes.

This feature is disabled by default in version 4 of Alfresco and can be enabled by setting the following value in the `alfresco-global.properties` file.

```
system.usages.enabled=true
```

This will come into effect on the next server restart.

The current stored content size is visible to the administrator when a user is examined in the User administration tool.

The administrator can optionally set a quota for a given user. This can be set when the user is created or imposed at a later time.

If a user tries to add or edit content, via any repository interface, so that usage will exceed their current quota then the user will see the error message "Quota Exceeded".

Solr

Alfresco version 4 introduced a change in architecture for indexing and user search; Apache Solr is now the default full text indexing technology. Lucene was used prior to this and can be enabled in favor of Solr if required.

Solr embeds and leverages Lucene, however Solr provides a functionally abstracted layer and can be installed on a dedicated server.

Solr delivers a number of advantages over Lucene including scalability, improved performance, enhanced language support and fine control over what gets indexed.

A principle issue for Lucene is transactional dependence; the indexes are updated as content is loaded or edited. This delivers fully accurate and synchronised indexes, but means the content server is locked during the transaction. This is a particular problem if the indexes ever have to be rebuilt from scratch, an operation that can take many hours with large content repositories.

The drawback and consideration when using Solr, is that it works asynchronously and is known as "eventually consistent". Content is indexed at small regular intervals and until the indexes are up to date.

Database factors

It is anticipated you will have an experienced, certified database administrator on staff to support your Alfresco installation, however you should understand why certain performance problems may arise with databases and here we take a high level overview of the factors which affect database performance.

One of the primary factors affecting database performance relates to having the correct indexes defined, however you should never need to do this as Alfresco has already created the optimal indexes for best performance. If your DBA finds that an index appears to be missing or adding an index improves performance this should be reported to Alfresco support.

All databases can be tuned for performance in different ways for differing usage patterns, for example high throughput, long running queries, decision support or mixed usage. This is typically

done through a number of parameters which affect cache size, threads and connections, pooling of resources and so on. There are courses, books and documentation available which deal with database tuning and your resident DBA should be familiar with these.

All the databases which Alfresco run have query optimizers which rely on database statistics to be up-to-date for best performance, all the databases offer tools for gathering and updating these statistics and this is often an overlooked task. Note that in some database types index maintenance and optimization is an expensive, and in some cases blocking, operation that can severely impact Alfresco performance while in progress. Your certified DBA will have best practices for scheduling these operations in your database.

All databases use files on the file system to manage their tables and indexes. Over time the files and tables can become fragmented and this too can have a degrading effect on performance. You should ensure that your DBA has set in place procedures for dealing with this fragmentation in the Alfresco schema.

Of course all of these depend on the database which you are running and you need to look to the database specific documentation for more information than can be provided here.

State of the database

If you have access to the database there are a number of visual tools which can be run that will provide you at least an overview of the health of your database. Some databases come with their own very useful graphical tools, for example MySQL, equally there are generic tools out there which can monitor a variety of databases and provide useful information, for example Hyperic and Splunk.

Backup

Ultimately as with any system you have to plan for unexpected disasters and problems. Any system is prone to failures of varying kinds ranging from hardware failure through to simple human error. You must ensure that you have a good set of backups and that backups have been tested to work and your recovery procedures support the business.

This process is explored in the Alfresco Element, Backup & recovery.

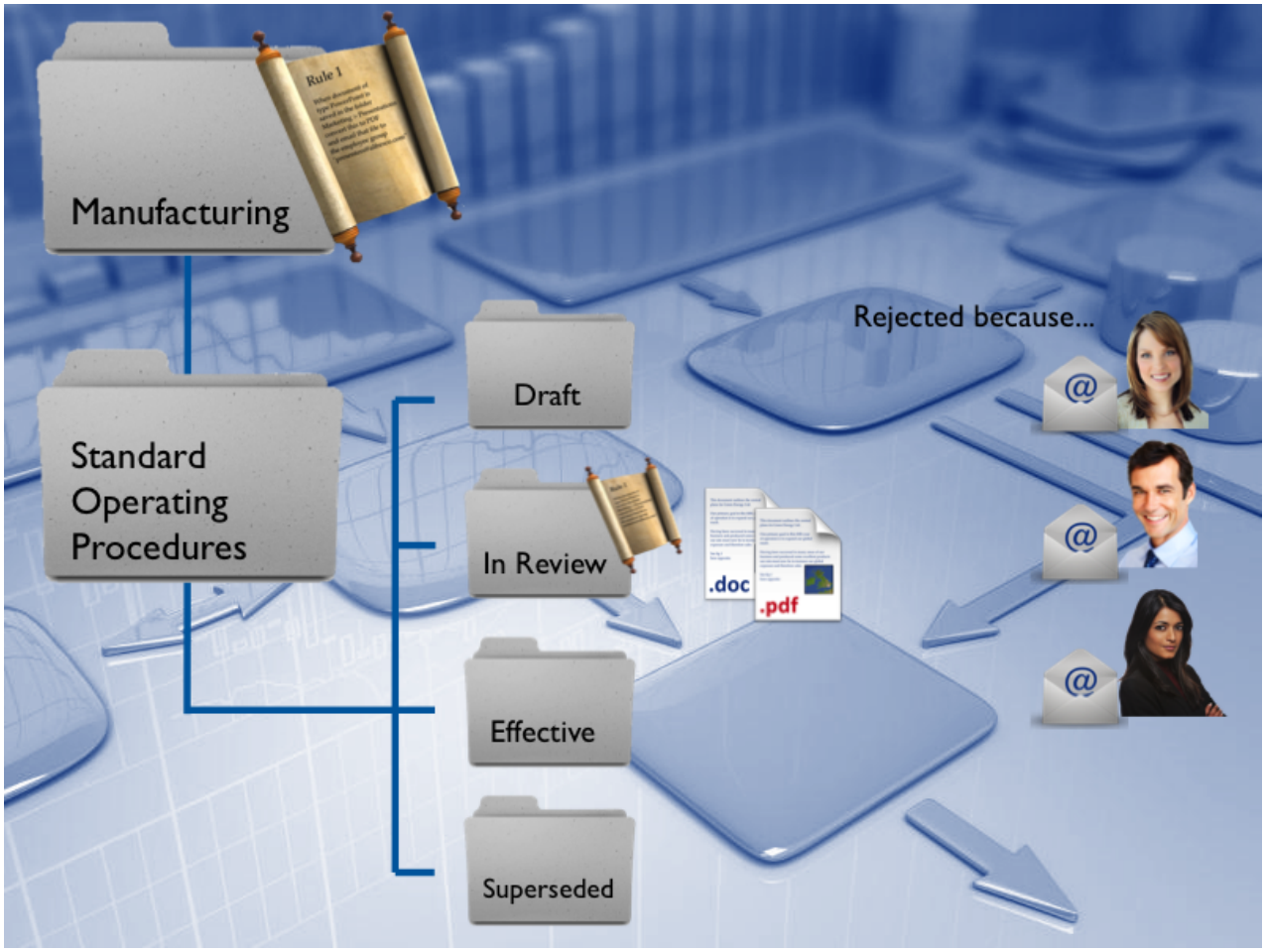
Holistic monitoring

We have talked about various ways of monitoring the health of your installation, you can of course use a number of widely available tools, which your organization may already use, for a holistic monitoring of all the components which make up an Alfresco installation, from the database to the application server.

Content rules

Content rules are a very powerful capability within Alfresco and let you accomplish a wide variety of content management tasks in your repository through a simple user interface. Typically these tasks would have previously required a developer to customize the system. Rules are limited by imagination rather than functionality, however some of the more typical uses of rules are for example: to perform some action on the document content; to perform some action on the document's properties; to change a document's status, notify somebody of a change in a document or put it into a workflow. Using rules you may also move, or copy, a document between folders.

Rules are defined on a folder and have a name and can be inherited, by default they just apply to the folder on which they are defined. Rules are designed to be run automatically when the rule condition is met and are by default run in the foreground synchronously.



Although each rule is simple and easy to define, by their nature they typically perform a single action. For more complex requirements you may need to combine two or more rules together to achieve your goals. When you have multiple rules they are triggered in a particular sequence which can be defined by you as an administrator.

Whilst a lot can be achieved with the rule actions which are standard, it is possible that the operation you want to achieve on your content is not there out of the box. In this case rules provide a catch-all action, execute script, action which allows a developer to write their own behavior, thus extending the range and power of content rules.

Demo: Content rules

In the following demonstration and lab we are going to use a folder structure which has already be created in the Green Energy repository. This defines the standard operating procedures for manufacturing.

In this example a rule has been added to the standard operating procedures folder which adds an aspect to any document added to this folder. The rule is such that it is inherited down to any child folders.

Another of the folders, In Review has its own rule defined Transform to PDF.

This is typically how you establish rules across folders.

In this demonstration I will establish a review and approval process using rules.

As the administrator I navigate within the repository to:

```
Geo-Thermal Division > Manufacturing > Standard Operating Procedures
```

And create a rule 'Transform to PDF for review'.

This operates on Word documents only and transforms content newly created or moved into the folder to a PDF file. The rule is created as a background process rather than the default foreground and is not applied to subfolders.

I will now add a second rule to this same folder where an approval action moves the PDF file to the Effective folder and a rejected action moves the file to Draft. The process is a foreground one and not applied to subfolders.

The ability to create and manage rules is defined by the authority held by the user. Within a Share site a user must hold the role of Manager in order to create and manage rules. Outside of a Share site and for folders within the repository the Coordinator permission must be held. The creator and owner of a folder will automatically have this ability. Security and permissions are examined in the Permissions Alfresco Element.

This is clearly a very simple rule and in the practice you would probably add further rules to add functionality such as notify an individual or group when a document had been placed in the Review folder. Add an aspect such as an effective date when the document was approved. You might wish to add an aspect to capture comments if a document were rejected. All of this and more is possible through the rules capability of Alfresco.

Now the rule is established let me demonstrate its functionality.

I move the Standard Operating Procedures document (which is in Microsoft Word format) from the Draft folder to the In Review folder. Immediately the rule creates a PDF version of this file.

Next I approve the PDF document and witness this being moved to the Effective folder.

The alternate action would be to reject the document and in this case it is moved to the Draft folder.

In practise you would have users undertake these move, approve or reject actions, however security and permissions have yet to be established for these folders so I undertook this demonstration as the administrator. security and permissions are examined Permissions Alfresco Element.

Lab - Content rules

1. In this lab you are going to create a rule to transform any PowerPoint file created or placed in a folder structure to PDF, and move the newly generated PDF file to a specific folder. You can log in as the administrator for this lab.
    1. Navigate to the folder `Geo-Thermal Division` in the repository.
    2. Add a rule to the Marketing folder to transform PowerPoint files to PDF and move the created PDF file to the folder:

        `Geo-Thermal Division > Marketing > Presentations`

        - The Mime type you should choose is `"Microsoft PowerPoint 2007 Presentation"`.
        - Make the rule inheritable.
        - Set the rule to run in the background.
    3. Test the rule by uploading the PowerPoint file: `Building a Brand Identity.pptx` to the folder:

        `Geo-Thermal Division > Marketing > Branding Project`

        The Building a Brand Identify file is found in the folder:

        `Desktop > Assets > Managing the repository`
    4. Check the Presentations folder for the newly created PDF file.

Managing the repository

Best practise

Rules by default are set to run in the foreground. This makes the user wait while the rules complete. Running a rule in the background however can raise additional difficulties and working in handling error conditions. Run all rules in the background unless you know that a rule is likely to generate errors often.

Alfresco lets you re-use rules from another folder. Consider using rules already defined which do what you need rather than redefining the same rule many times. Rules are simple so break-up complex business processes into multiple rules.

Another technique you can use is the concept of a drop zone or drop folder. This involves creating a folder which is used simply for uploading documents and has all the rules on it. The rules filter the incoming content and move the documents to the final destination folder.

Finally if you need to have a marker or status on a document the easiest way to do that is to create an aspect and then attached the aspect through a rule.

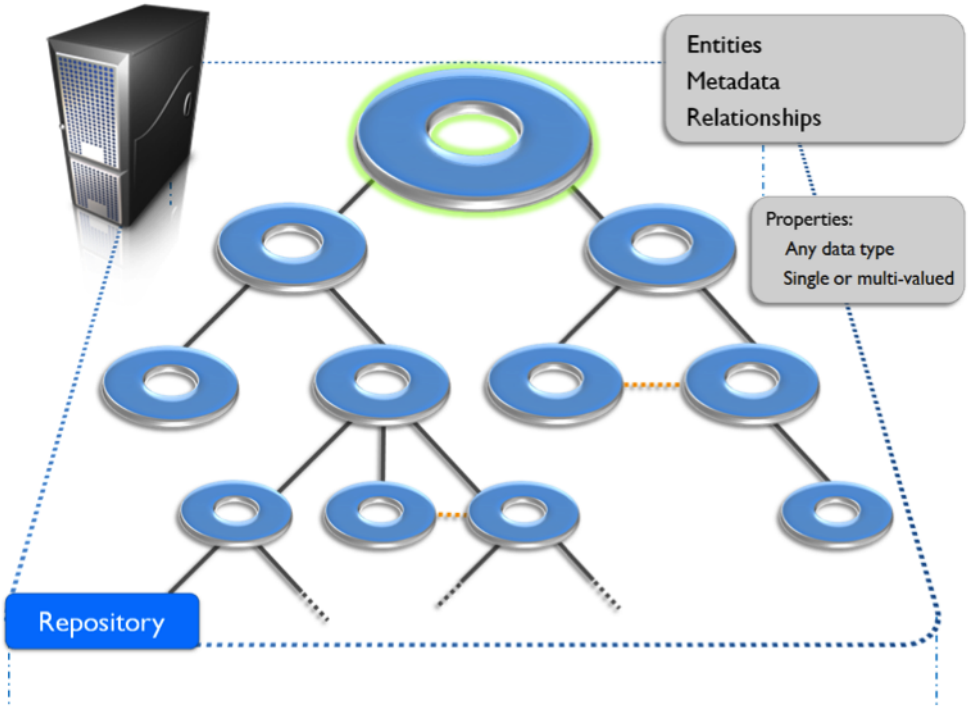
Summary

This section has given you a good grounding into the tasks which need to be done on a regular basis to keep an Alfresco repository healthy. We have also looked at the best way of installing applications and introduced the concept of scheduled jobs.

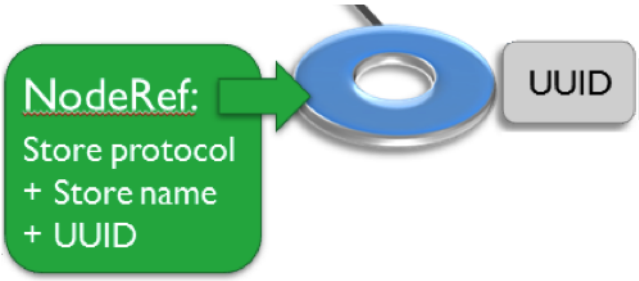
Content Model Overview

In this Alfresco Element content models are explored. You will understand what a content model is, understand the different ways content models can be implemented and how they are deployed through the bootstrapping process. You will also examine how content models are presented in the user interface.

Everything which Alfresco stores, is stored as a node. This is a simple, uncomplicated model for recording entities, metadata, and relationships. The repository is a tree of nodes where each node supports one or more properties whose values may be of any data type and either single or multi-valued. A node has at least one parent, except for the root node, and may contain one or more child nodes. Nodes may also be related through arbitrary peer relationships.



Any new node is assigned a Universally Unique Identifier, or UUID, which remains with it unchanged for the duration of its life. Any node can be referenced using a combination of store protocol, store name and UUID, this unique combination is known as the NodeRef.



Nodes constrained

- A node must be of a given kind.
- A node must carry a defined set of properties.

- A property must be of a given data type.
- A value must be within a defined set of values.
- A node must be related to other nodes in a particular way.

These constraints allow the definition of entities within the content management domain. For example, many ECM applications are built around the notion of folders and documents. It is content modeling that adds this meaning to the node data structure. The possible types of nodes, properties, associations and aspects, which a node supports are all defined by the content model.

In the example we can see that folders can appear within folders and folders may contain documents, which in turn may have renditions. In this case the associations between folders and documents, would have been created by the user as they create objects, or nodes, in the system using a user interface. Whereas the rendition relationship, would have been created automatically by the system, based on a user request or rule in the system.

Out of the box, Alfresco comes prepackaged with several content models for support of the common or standardized aspects of ECM, especially for Document and Records Management.

You can accomplish a lot with just these models but, if you want to break beyond the classic file system, you'll need to roll up your sleeves and get modeling to support the specific needs of your ECM requirement.

Content modeling is really all about metadata, that is, data describing data.

Three layer metadata model

To put content modeling in context, Alfresco talks in terms of the following three layer metadata model.

The first layer, M0, represents the nodes, properties, and relationships held in the Alfresco content repository. These entities are managed through the various content repository services, such as the file folder service or CMIS (Content Management Interoperability Services).

The next layer, M1, is a content model that defines a set of related definitions to constrain the nodes in layer M0. Many content models may be registered with the content repository.

A content model is itself described by the next layer, M2, the content metamodel.

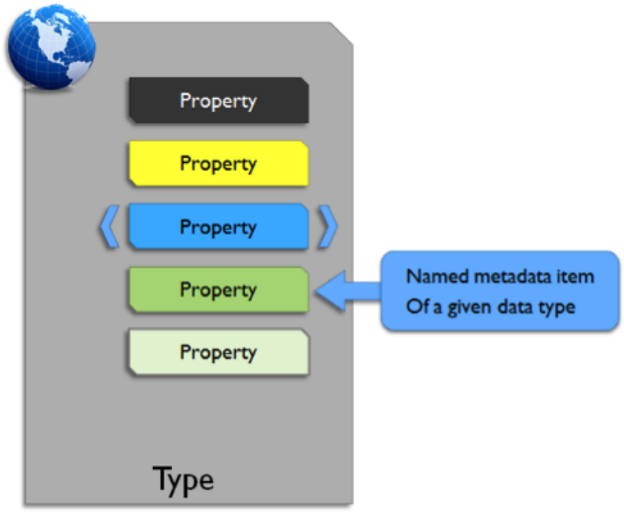
This is the section which we focus on, as this is how content models are expressed for extending Alfresco. If you do not understand the content metamodel, you cannot define a content model. There are already some standardized content metamodels: the CMIS data model and JSR-170 node type model.

In summary, we will be learning about the Alfresco content metamodel (M2), in order to define one or more content models (M1), to constrain the structure of your nodes (M0) held in your content repository.

Type, properties and constraints

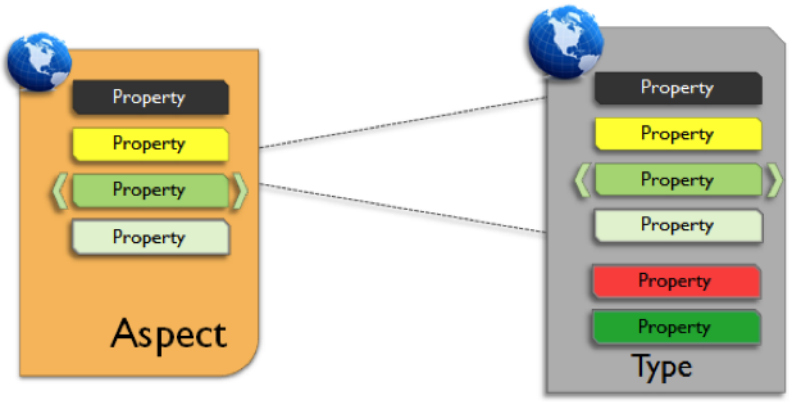
At the heart of the content metamodel is a type. If you are familiar with object-oriented models then a type is just like a type or class and should be familiar. It represents objects in the real world with support for properties and the ability to inherit the definition of a parent type.

Properties are named items of metadata associated with the type where each property is of a given data type. Constraints can be applied to restrict the values of a property.



Aspects

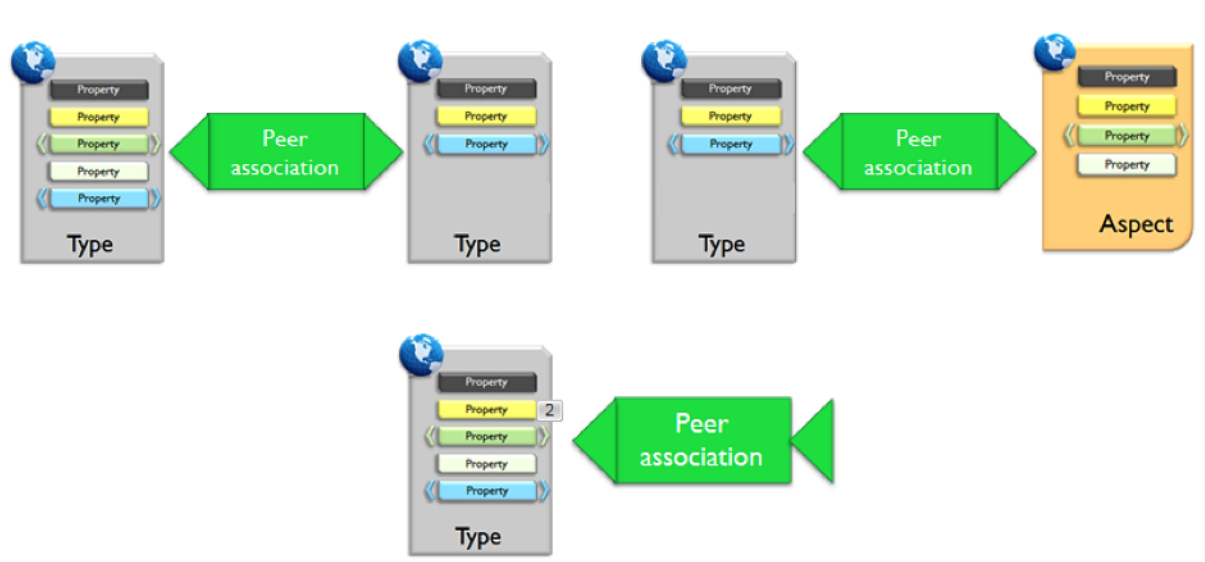
Alfresco supports aspects in its data model, these are extremely useful and flexible modeling tools. An aspect supports the same capabilities as a type, meaning it supports properties, and may be related to and may inherit the definition of a parent aspect. On the surface, aspects seem very similar to types, but there is one important distinguishing feature, which is that aspects may be shared across types. In other words, aspects allow cross-cutting of the content model, which is the sharing of property and association definitions by attaching them to multiple types. This is the content metamodel equivalent of multiple inheritance.



Associations

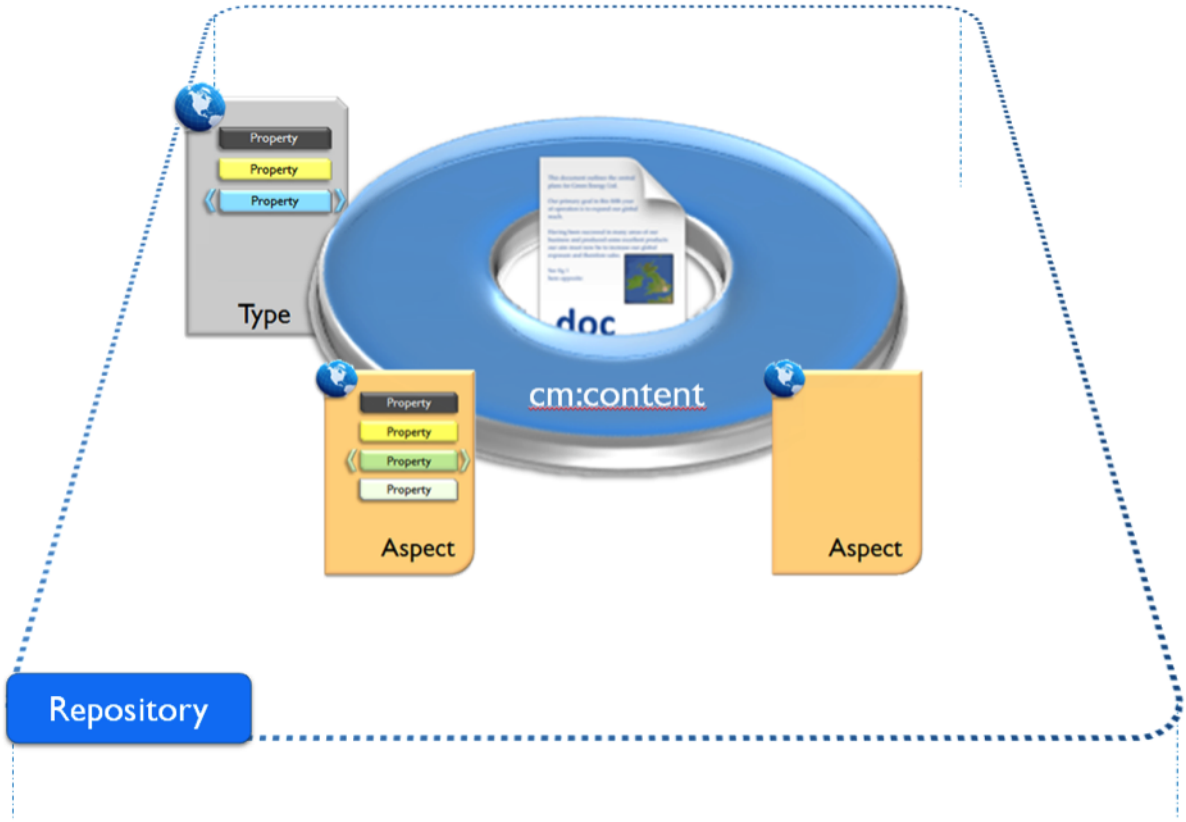
Relationships between types are modeled with associations, of which there are two kinds: child associations and peer associations. Child associations provide the ability to model compositions where the parent effectively owns the children and, as such, operations like delete will propagate through to the children, avoiding orphans. Peer associations, on the other hand, define a relationship between two objects where neither object is superior to the other.

Associations can be created between two different types, a type and aspect, or between a type and itself.



Content models

When a node is initially created in the system it must be of a given type from any of the content models. By default when a user creates a document the type will be a cm:content type. The type defines the properties which the node will have, a node may then have one or more aspects applied to it. These aspects may come with many properties of their own (or none at all), in this way a sophisticated range of metadata can be applied to a node.



Built-in models

In order to support the functionality which Alfresco provides as an ECM system some content models need to exist in the repository. Alfresco uses exactly the same techniques to define

content models as we describe on this course. The built-in content models are useful because they contain repository functionality and behavior and the Alfresco clients know how to deal with them, for example the display of a date datatype in Share. These built-in models can be used by you as a developer to extend the system, in some cases you need to use the content models, unless you want to do a lot of work re-writing functionality!

Although you should have little need to inspect the models directly you will find each model in a separate file called `<name>Model.xml` (e.g. **dictionaryModel.xml**) in the directory:

- `{tomcat_root}/webapps/alfresco/WEB-INF/classes/alfresco/model`

The primary built-in models are shown in the table below:

Abbreviation	Name	File	Description
app	application	application	application specific types and aspects, in this case folders, links and simple workflow.
cm	content model	contentModel.xml	A complex model which contains the major types of the system such as content, person, etc. based off the system model.
d	dictionary	dictionaryModel.xml	The data types used in other models.
sys	system	systemModel.xml	Contains the base types used in other models.
usr	user	in *alfresco.war*	Contains the definition of authorities. Note this file in embedded in the .war file and is not found in the models directory.

The following lists the built-in types and aspects which are available as a result of the built-in models. This is a reference list and may change over times as new types and aspects are added.

Built-In Types	Built-In Aspects	
cm:cmobject	cm:titled	cm:summarizable
cm:folder	cm:auditable	cm:countable
cm:content	cm:transformable	cm:copiedfrom
cm:dictionaryModel	cm:templatable	cm:workingcopy
cm:link	cm:webscriptable	cm:versionable
cm:savedquery	cm:complianceable	cm:lockable
cm:systemfolder	cm:ownable	cm:subscribable
cm:person	cm:author	cm:classifiable
cm:category_root	cm:dublincore	cm:generalclassifiable
cm:category	cm:basable	cm:attachable
cm:mlRoot	cm:referencing	cm:emailedcm:referencesnode
cm:mlContainer	cm:replaceable	cm:mlDocument
	cm:effectivity	cm:mlEmptyTranslation

Model deployment

Content models are simply XML files which correspond to a particular schema. A content model can be deployed in a number of ways into an Alfresco repository and once deployed the content model objects are then available for use.

However simply deploying a content model into the content server has no effect at all on the Alfresco clients. Additionally to make sure that your model can be used successfully in client applications you will also need to configure the Alfresco clients, such as Share, to be *model-aware*.

This concludes the overview of Alfresco content models.

Creating content models

In this Alfresco Element you will create and deploy a content model, and configure the Share interface to access this new definition. You will understand what can and cannot be done in a content model, examine best practice and explore advanced topics.

Throughout this section our examples are only shown as fragments of the content model for brevity and conciseness. These fragments build up throughout the section but will not work in isolation; they must exist in the context of a full content model. The full content model code can be found at the end of this chapter and within the **Assets > Creating content models > Final** folder found on the desktop of your virtual machine.

Alfresco dictionary schema

Alfresco content models must conform to the Alfresco dictionary schema. This is supplied as a `modelSchema.xsd` file which can be used to instruct your XML editor, if you use one, to validate against.

Content model file structure

Identification

Your model needs to be uniquely identified this is done by giving it a name and declaring a namespace with an associated prefix.

Each model, including the Alfresco models, has its own namespace which is mapped onto a prefix that is local for your model, this is often a unique string commonly prefixed with an HTTP address associated with the author. Namespaces are defined by a URI, this must be unique, but the location it refers to is not used, but should be under your control. The namespace is mapped onto a namespace prefix which is local to the file it is defined in.

```
<!-- Introduction of new namespaces defined by this model -->
<namespaces>
   <namespace uri="http://marketing.green-energy-demo.com" prefix="mar"/>
</namespaces>
```

To associate a name with a namespace, it is only necessary to prefix the name with the relevant namespace prefix. For example, if the namespace URI markerting.green-energy-demo.com and associated prefix mar are defined, then the name `mar:customtype` means that customtype is a name defined within the namespace marketing.green-energy-demo.com.

```
<model name="mar:marketingmodel" xmlns="http://www.alfresco.org/model/dictionary/1.0">

   <!-- Optional meta-data about the model -->
   <description>Green Energy Model for Marketing Documents</description>
   <author>Alfresco Training</author>
   <version>1.0</version>

</model>
```

Each content model must define at least one namespace for the names defined in that content model.

Imports

You can import one model into another allowing you to use definitions from the imported model. You will always need to do this for some of the Alfresco models in order to use data types for instance and the standard built-in types.

Imports are not physical file imports. The model is read from within the models the server has already loaded and identified by its URI.

```
<!-- Imports are required to allow references to definitions in other models -->
<imports>
   <!-- Import Alfresco Dictionary Definitions -->
   <import uri="http://www.alfresco.org/model/dictionary/1.0" prefix="d"/>
   <!-- Import Alfresco Content Domain Model Definitions -->
   <import uri="http://www.alfresco.org/model/content/1.0" prefix="cm"/>
</imports>
```

Types

A content model may define one or more types. Each type is uniquely named and is optionally labeled with a title and description. A type may declare any number of properties to represent metadata associated with the type and any number of associations to other types.

Although you could declare a type that is not a sub-type of an existing type this would require additional work to handle it in the system. Therefore we need to specify a parent on our type declaration, in most cases it will be `cm:content` or one of your own content types. This is very important as we want all the inbuilt behavior of the existing content type.

```
<types>
   <!-- Definition of new Content Type: Marketing Collateral-->
   <type name="mar:collateral">
      <title>Marketing Collateral</title>
      <parent>cm:content</parent>
   </type>
</types>
```

Obviously a type without any properties is not very useful, so we need to add some properties, let's see what types of properties we can add.

Properties

Each property must be uniquely named and is optionally labeled with a title and description. The only feature of a property that must be specified is its data type of which the content repository supports a wide variety. Each data type is named, with some commonly used data types provided out of the box.

Datatype	Description	
d:text	A text value, a character string	
d:int	An integer value	java.lang.Integer equivalent
d:long	A long value	java.lang.Long equivalent
d:float	A float value	java.lang.Float equivalent
d:double	A double value	java.lang.Double equivalent
d:date	A date value	java.lang.Date equivalent
d:datetime	A date and time value	java.lang.Date equivalent
d:boolean	A Boolean value	java.lang.Boolean equivalent
d:content	Arbitrarily long text or binary stream	
d:mltext	Multilingual text value where many localized representations of the text value may be held	
d:any	Any value regardless of type	

The following example shows a property called reviewDate, note the model prefix – `mar`. This is of type `date`, which can be found in the imported dictionary model.

```
<property name="mar:reviewDate">
    <type>d:date</type>
</property>
```

With the basics defined, it is possible to fine-tune the definition of a property. By default, a property supports a single value but this may be changed to support multiple values via the multiple element. Multiple values are rendered as lists in the various Alfresco APIs.

```
<property name="mar:reviewDate">
    <type>d:date</type>
    <multiple>true</multiple>
</property>
```

A value can also be mandated on creation of a node via the mandatory element.

```
<property name="mar:reviewDate">
    <type>d:date</type>
    <mandatory>true</mandatory>
</property>
```

That is, a transaction will not commit unless all mandatory values have been provided for nodes modified within that transaction.

There are actually two forms of mandatory: enforced and relaxed. Enforced is as described but relaxed gives finer control over when mandatory values must be provided.

A transaction will still commit if a relaxed mandatory value has not been specified; however, the node with the missing value will be marked as incomplete (via the `sys:incomplete` aspect). This is important, as not all content creation processes (for example, via many of the protocols supported by Alfresco) provide the ability to set property values. Custom solutions can then be configured or built that trigger a business process for collecting the missing values (for example, via workflow user-assigned tasks).

In conjunction with mandating a value, a property definition can also specify a default value that is set automatically by the content repository if the value has not been set at transaction commit time.

Designating a property to be protected renders it read-only to the user and updatable only by a server side process or code.

Content model deployment

A content model is defined in its entirety as a single XML document, which must comply with the content metamodel XSD schema provided by the Alfresco content repository. Each model contains a set of related and coherent definitions, and is deployed as a unit.

Several content models may be deployed to the content repository and definitions in one content model may depend on definitions in another content model, allowing for the sharing of definitions.

There are two approaches to deploying a content model into the content repository: bootstrap and dynamic. The bootstrap approach involves modifying Alfresco content repository XML configuration files in order to register the content model such that on startup of the content repository, the content model is read, validated, and registered. With the bootstrap approach, changes to model definitions through the content model XML file are only registered after restarting the content repository. This can lead to long development and test cycles, so the bootstrap approach is more suited to final production deployment.

Using the bootstrap approach additionally provides an increased level of validation, ensuring that content models that have a secondary dependency are not dynamically deleted or altered which can prevent a repository from starting successfully.

For this reason, an alternate dynamic approach to deploying a content model is provided, allowing the registration and updates to content models without the need to restart the content repository to pick up the changes. Instead of placing content model XML files into the classpath, they are placed in the content repository itself.

Although dynamic deployment is helpful because you can makes changes to a model without restarting the server there are some disadvantages and restrictions on what changes can be made to a content model XML file and when a content model XML file can be deleted. For example only incremental additions, changes that do not require modifications to existing data in

the content repository, are allowed and the content model can be deleted only if it is not used by any data in the content repository.

Demo: Content model deployment

In this demonstration I will show dynamic content model deployment. I will load the marketing collateral content model into the repository. This can be achieved with Alfresco Share, which provides full access to the data dictionary.

No content of this model type currently exists. In a live installation should that not be true, only incremental changes would be possible, that is no modification to existing data.

I head to the `Repository > Data Dictionary` and then `Models` folder.

I can either upload my XML file or Create the Content. I choose `Create Content > XML` and then enter the content model name.

Heading back to the Marketing Model XML file I copy the content and then paste this, finally I choose `Create`.

When creating or uploading a content model, by default, this will be inactive.

To enable the model I edit the metadata, click `Model Active` and then choose `save`. The model will be validated and made active.

Full details of any parsing errors will be found in the Alfresco log file.

A content model can be deleted by simply deleting the XML file from the data dictionary. This is only possible if the model is not in use by any repository content.

To deactivate and model, simply edit the metadata and deselect the `Model Active` checkbox, and then `Save`, I'll cancel since this is our work in progress

Demo: Content model usage and Share configuration

Now the content model has been successfully loaded to the repository it is available in the system. To make this visible in the Share interface some configuration must be undertaken and this has been completed (an overview will follow).

Harriet Slim logs in, she is Green Energy's Marketing Director.

She navigates to the `Repository, Geo-Thermal Division > Marketing > Branding Project` folder.

Selecting the `Developing Brand Identity and Personality` PDF she chooses the `Change Type` function and selects `Marketing Collateral`, which is the new custom content type.

Now this type change is confirmed, you can see that the reviewed date and review period metadata items are visible.

Editing the metadata the review period value can be entered and adjusted as desired. The reviewed date is intended to be set automatically, prompted by a workflow.

To enable the content model to be displayed in the Share interface, the `share-config-custom.xml` file must be edited. This resides in `TOMCAT_HOME/shared/classes/alfresco/web-extension`

To enable the Change Type menu to display the content model name I created the file `artwork-client-context.xml`, saved in the same directory and `artworkModel.properties` file saved in the messages folder.

As examples these files can be found as attachments to this slide.

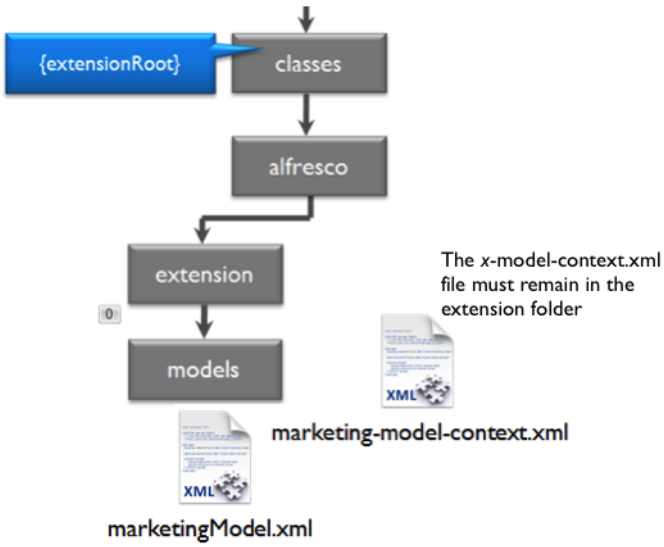
This topic is explored in greater detail on the Alfresco Share Configuration course.

Dictionary bootstrap

A repository component called the Dictionary Bootstrap is responsible for loading a specified list of content models and registering them with the content repository. To register new content models, it is necessary either to modify the content model list of an existing Dictionary Bootstrap component or define a new Dictionary Bootstrap component. For encapsulated modular extensions, it is recommended to define a new Dictionary Bootstrap component. This is achieved by creating the file, your model name-model-context.xml. Your model can then be registered in the bean called `extension.dictionaryBootstrap`.

```
<bean id="marketingmodel.extension.dictionaryBootstrap" parent="dictionaryModelBootstrap"
    depends-on="dictionaryBootstrap">
  <property name="models">
    <list>
      <value>alfresco/extension/marketingModel.xml</value>
    </list>
  </property>
</bean>
```

It is recommended when creating your bootstrapped content model to create a new directory for this as shown below.



Content model deployment practice

Best practice would see you use dynamic deployment whilst developing your content model. During this period you should be prepared to delete content and even trash your entire repository.

When the content model is finalized you should use the bootstrap method to install this, remembering to register it.

Additionally you should create a directory folder for your content models.

Finally you will need to configure the client user interface to expose your content model.

Lab - Creating a content model 1

In this lab you are going to create a content model for Green Energy's standard operating procedures, (the SOP) and deploy this dynamically.

In your virtual lab environment you will find a specification for the content model in the: `Desktop > Assets > Creating content models` folder. The document is called **Green Energy - SOP Specification.pdf**. This can also be found in the appendix of this document.

This specification defines the types you are required to create in this lab, and the constraints, associations and aspects you will create later in this Alfresco Element. Please review this document now.

You will now define the content model type properties. You should open the file `sopModel Start here.xml` and save this as `sopModel.xml` as you make changes. Use the **gedit**b> editor. This can be found in the `Applications > Accessories` menu, or by right mouse clicking over a file. This program understands xml and colors the syntax appropriately.

Create the content model

1. Create the following properties (as defined in the **Green Energy – SOP Specification.pdf**):
    - nextReviewDate
    - proposedEffectiveDate
    - reviewDate
    - reviewPeriod
2. Deploy the content model dynamically as administrator and make it active.
3. Use the `Change Type` function to apply the new type to a document in the repository and witness the additional information now displayed in the Share interface.

The configuration of the Share client in your virtual lab environment has already been done for you. You must therefore use the naming convention given.

Within the `Final` folder you will find the file `sopModel.xml`. This is the completed content model and is the full answer for all the labs in this Alfresco Element. See how far you can get without referring to it.

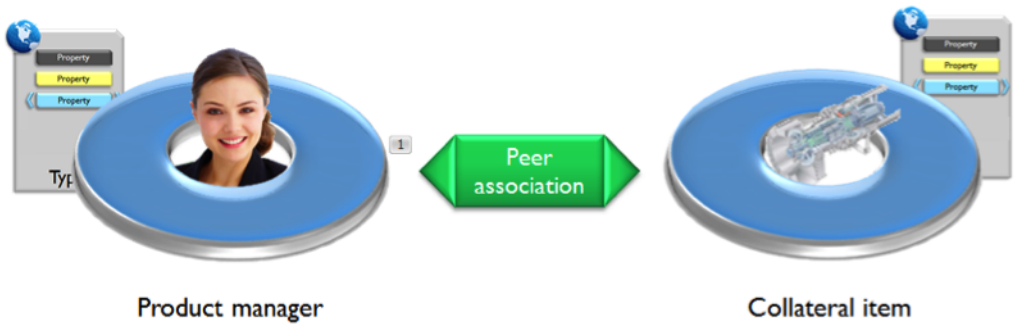
Creating content models

Associations

Associations allow relationships to be created between two different types - a source type and a target type - where the source type is inherently the type that defines association. The only feature of an association that must be specified is its target type via the class element on the target end.

```
<types>
   <type name="mar:collateral">
      <title>Marketing Collateral</title>
      <parent>cm:content</parent>
      <properties>
         <property name="mar:nextReviewDate">
            <type>d:date</type>
         </property>
      </properties>
      <!-- Associated Objects -->
      <associations>
         <association name="mar:productManager">
            <title>Marketing Owner</title>
            <target>
               <class>cm:person</class>
            </target>
         </association>
      </associations>
   </type>
</types>
```

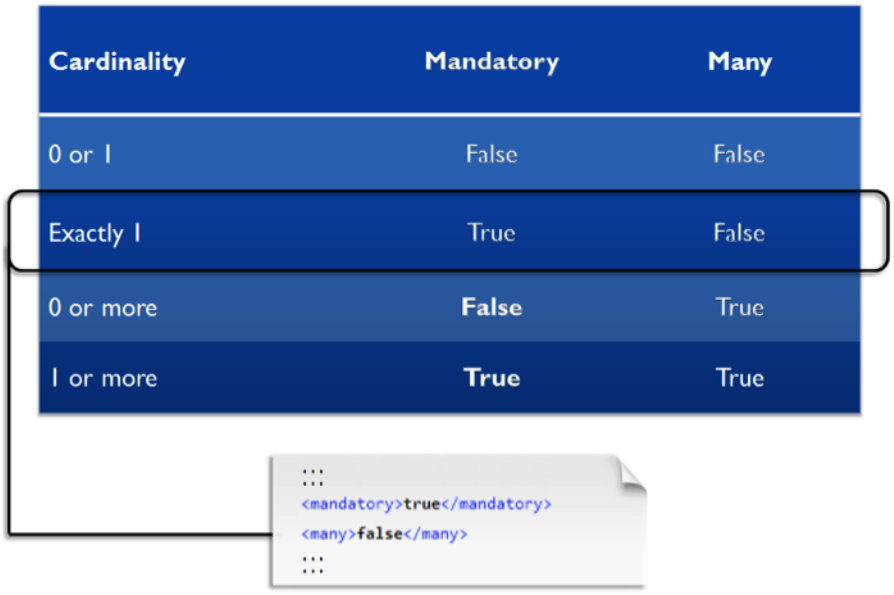
As you know peer and child associations exist. In the instance of child associations be aware there is no way in the standard user interface for cascade delete to work on your own types unless you programmatically listen for deletes as a policy in the server. This is because delete only cascades along the single child association used to create the child node, known as the primary child association. If you create child associations between pre-existing nodes, then delete will not cascade along them. Since when a node is created, through Share for example, the primary association will always be between the object and the folder.



To develop the Green Energy Marketing model lets define an association between a collateral item and a product manager. It will be a requirement for a collateral item to have a product manager, but only one product manager is permitted. To achieve this, the mandatory element is used, set to true. Next, the many element is introduced, set to false.

Cardinality

Cardinality between associations can be specified through the mandatory and many elements, as shown in the following table:



The Green Energy production manager to collateral association was defined as exactly one.

Alfresco folder - Content model definition

The Alfresco content model definition of a folder and content is shown here. Note the, 0 or more, child association; permitting folders with no content. The class `sys:base` contains `cm:content` permitting content of any format, which also permits folders within folders. The `duplicate` element defines whether content can have the same name within a folder and `propagateTimestamps` will update a parent time stamp if the child content is modified or updated.

```
<type name="cm:folder">
   <title>Folder</title>
   <parent>cm:cmobject</parent>
   <archive>true</archive>
   <associations>
      <child-association name="cm:contains">
         <source>
            <mandatory>false</mandatory>
            <many>true</many>
         </source>
         <target>
            <class>sys:base</class>
            <mandatory>false</mandatory>
            <many>true</many>
         </target>
         <duplicate>false</duplicate>
         <propagateTimestamps>true</propagateTimestamps>
      </child-association>
   </associations>
</type>
```

Demo: Associations

Progressing the development of the Green Energy marketing model I am going to create an association to define an owner for the review process.

I open the marketing model xml file and paste in the new association. A review owner cardinality is exactly one. (mandatory = true and many is set to false).The target class is `cm:person`

There is some additional xml needed here, which is the source, many element set to true. This allows a product manager (`cm:person`) to be associated with more than one product, if this were not included a single `cm:person` would only ever be able to be associated with a single `mar:collateral` item.

I save the file, then `Edit` menu, `select all, copy`. As this model is still in development I will continue to deploy this dynamically.

As administrator I navigate to the Data Dictionary > Models folder and open the `marketingModel.xml` file.

I paste in the new code.

Then choose `Save`.

Since this is an incremental addition to the content model it is permitted, even though content exists of this type.

When a user (such as Harriet) select a document of the marketing collateral type, the newly created association presents itself as the review process manager. She selects herself and saves. The association is now visible.

Lab - Creating a content model 2

Associations

In this lab you are going to add two specified associations to your Green Energy SOP content model.

1. Review the **Green Energy - SOP Specification.pdf** document. You will find the `owner` and `supersededSops` associations are to be created.
2. Edit the `sopModel.xml` you created previously and add the two specified associations:
   - `owner` (cardinality: Exactly one. The class is `cm:person`)
   - `supersededSops` (cardinality: Zero or more. The class is `sop:sop`)
3. Deploy the content model dynamically as the administrator.
4. Test your model.

As before the Share configuration has been completed, this is why the names must be as specified so that the Share customization and model will tie together.

Creating content models

Aspects

Aspects form their own hierarchy apart from the type hierarchy and each content model may define one or more aspects. Aspects support all the same features as types and, as such, are defined in the same way as types. An aspect may be attached to one or more types, this means that a node created of that type automatically inherits the attached aspects upon creation.

```
<aspects>
    <!-- This aspect is used to track marketing
         specific information about the document -->
    <aspect name="mar:marketing">
        <title>Marketing Details</title>
     <properties>
        <property name="mar:region">
           <type>d:text</type>
           <default>Global</default>
        </property>
     </properties>
    </aspect>
</aspects>
```

Aspects allow property and association definitions to be shared across many types of nodes. This means a cross-cutting feature of an ECM domain model may be encapsulated and applied throughout the rigid part of the model represented by types. As aspects can be applied at any time this effectively provides “late-binding” for properties to objects. Effectively this allows properties to be present only where applicable rather than be always present 'just in case‘.

A node in the content repository must be of a single type, but may have one or more aspects attached. The aspects are either inherited from its type (as defined in the content model), or

can be attached or detached at runtime, allowing a node to dynamically inherit features and capabilities. An example of this is the result of a user action or system workflow.

Aspects can be interpreted by the repository, to change behaviour, for example the presence of an aspect can be an indicator (much like a flag), for example with Records Management a vital record is indicated by an aspect with no properties. This is an example of how the system itself provides functionality through aspects, others system uses include versioning and categorization. Further aspects can be applied by the user or developer, such as the metadata set Dublin Core.

Lab - Creating a content model 3

Aspects

In this lab you will add an aspect to your developing content model.

1. Review the content model specification.
2. Create the auto review aspect, name `autoReview`
    - The property name is `autoReviewStart`, define the type and default value.
3. Deploy the content model dynamically.
4. Use `Manage Aspects` to apply the aspect and test by editing the metadata.

Stretch Goal

1. Create the aspects; `effectivity` and `sign off`
2. The effectivity aspect name is `effective`, implemented using an association.
3. The property name is `effective_date`
4. The sign off aspect name is `signoff`.
5. The association name is `signatory`
6. The target class is `cm:person`, and its cardinality is zero or one.

Remember to test your model again, **Note:** The aspects `effectivity` and `signoff` are intended to be completed automatically as part of an automatic workflow (not explored in this Alfresco Element).

Creating content models

Constraints

Constraints can be applied to properties to restrict their values. Constraints appear at this point in the content model because this is where the XSD schema for the content model expects them to be.

Constraints can either be defined standalone or inline. The example shown is a standalone constraint, allowing for reuse of the constraint across many properties. Within the type element, the constraint is referenced.

```
<constraints>
     <constraint name="mar:mediumList" type="LIST">
          <parameter name="allowedValues">
               <list>
                    <value>web</value>
                    <value>print</value>
                    <value>radio</value>
                    <value>tv</value>
               </list>
          </parameter>
     </constraint>
</constraints>
```

Inline constraints are defined within and for a single property.

```
<property name="mar:propertyexample">
    <type>d:text</type>
        <protected>true</protected>
        <default></default>

        <constraints>
            <constraint type="LENGTH">
            <parameter name="minLength"><value>0</value></parameter>
            <parameter name="maxLength"><value>128</value></parameter>
        </constraint>
    </constraints>

</property>
```

A standalone constraint must specify a unique name and a type. There are several constraint types provided out of the box where the commonly used types are:

- **LENGTH -** Text property value length must reside within minimum and maximum length limits.
- **MINMAX -** Numeric property value must reside within minimum and maximum range limits.
- **LIST -** Property value must be one of the specified list of values.
- **REGEX -**Property value matches regular expression.

Custom constraint types may be developed and registered with the content repository, these can be written in Java and can do complex things such as query other systems for example.

Length

The LENGTH constraint type limits the minimum and maximum string length of a property value, using the minLength and maxLength parameters. These parameters can be used independently, they do not have to be used together in a constraint.

```
<constraint name="ge:phonenumber" type="LENGTH">
  <parameter name="minLength"><value>10</value></parameter>
  <parameter name="maxLength"><value>12</value></parameter>
</constraint>
```

Minmax

The MINMAX constraint type limits the minimum and maximum numerical value of a property, using the minValue and maxValue parameters. As for LENGTH the minValue and maxValue parameters can be used independently.

```
<constraint name="ge:severity" type="MINMAX">
    <parameter name="minValue">
        <value>1</value>
    </parameter>
    <parameter name="maxValue">
        <value>10</value>
    </parameter>
</constraint>
```

List

Where the multiple element is used in a type, and set to true, the LIST constraint type allows for the definition of values in that list. These will be presented as a drop-down menu within the user interface.

```
<constraint name="ge:status" type="LIST">
    <parameter name="allowedValues">
        <list>
            <value>New</value>
            <value>In process</value>
            <value>Closed</value>
        </list>
    </parameter>
</constraint>
```

Regular expression

Regular expressions are a powerful mechanism for restricting and validating property values, however there is a steep learning curve associated with these. The constraint below will restrict the input to: numbers only, three banks of three digits separated by a hyphen.

```
<constraint name="ge:carNumberConstraint" type="REGEX">
    <parameter name="expression">
      <value><![CDATA[[0-9]\{3\}-[0-9]\{3\}-[0-9]\{3\}]]></value>
    </parameter>
</constraint>
```

When coding the regular expression in your content model this should be embedded between the CD data block as shown here using the exact syntax:

```
<![CDATA[ regular expression entered here ]]>
```

This second example of a regular expression shows how to validate for an email address.

```
<constraint name="mar:enquiryCheck" type="REGEX">
    <parameter name="expression">
        <value><![CDATA[^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$]]></value>
    </parameter>
</constraint>
```

In constructing your regular expression a good way to test is to use a command line tool, for example grep or sed. Other tools available are:

RegexBuddy found at this URL: http://www.regexbuddy.com/

plug-ins for Eclipse, for example QuickREx: http://sourceforge.net/projects/quickrex

A regular expression tutorial can be found at this URL: http://www.regular-expressions.info/tutorial.html

Lab - Creating a content model 4

In this lab you are going to add a standalone constraint to your model.

Constraints

1. Create the type property `visibility` and the constraint ref: `sop:visibilityList`
2. Create the constraint `visibilityList`of type LIST
3. Deploy the content model.
4. Test the constraint.

Creating content models

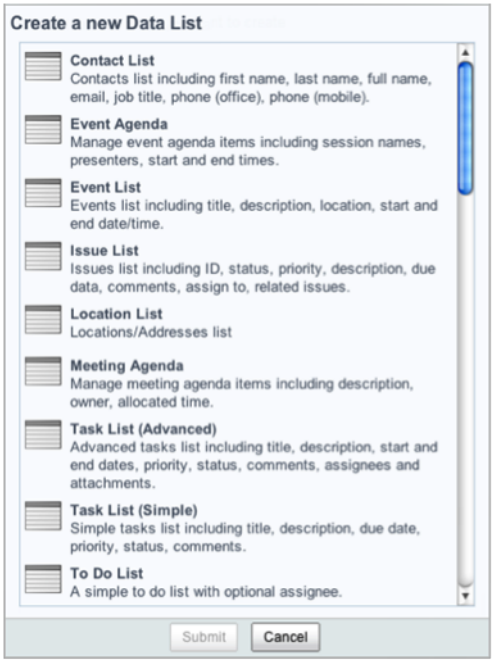
Advanced topics

Data lists

Data lists are grouped by Site and there maybe multiple lists types per site. The structure of these lists are driven by their type and aspects applied. Data lists offer great flexibility because they are simply an extension of the standard modeling and forms configuration and as such allow you to leverage the skills and techniques you have already learned about on this course.

When installed there are some predefined list types such as Tasks, both simple and advanced, Issues, Contacts and Events. This list may change over time as we extend the range of functionality offered. These can be used as they are, but in some cases you may find that none of the vanilla data lists quite match your specific requirement. In such cases, your knowledge of content models should enable you to extend an existing list or create a new list type.

Each data list has its own type, for example `dl:todoList`, and must be a subtype of `dl:dataListItem`. Data lists are stored by site, there is no concept of a global data list.



Data lists are displayed in the Share user interface through a data grid. This knows about the target data types, for example person, date, content, folders and objects. An assigned person will link to that person's profile, attached content items will link to that item in the repository. The client-side JavaScript used to render the different types is found in;

`<TOMCAT_HOME>/webapps/share/components/data-lists/datagrid.js`

Demo: Data lists

In this demo a custom data list will be created for Green Energy. To support a product launch the marketing department must complete a suite of collateral. Some of this content is produced by external agencies. The data list will provide tracking to the development and release of this collateral.

The custom data list will contain the following fields:

- Collateral piece name
- Agency developing the collateral
- Expected completion date
- Date approved
- Priority
- Production status
- Notes
- Collateral owner
- Collateral item
- Other attachments

Let's extend the existing marketing content model to implement this data list.

I open the content model file.

First I add the import for the data list model definition, prefix `dl`.

Next I add the constraints. These are standalone and will be referenced by an aspect soon to be defined.

The constraints are;

- `Region list`.
- `Language list`.
- `Medium list`.
- `Enquiry check`, to be used to validate email addresses.
- `Task status`.
- `Task priority`.

The collateral content type is updated with the following two properties:

- `nextReveiwDate`, used to initiate automatic reivews for collateral items.
- `proposedPublicationDate`, a suggested publication date.

The tracking data list will now be added. The individual fields which comprise the data list are:

- `Collateral piece name`.
- `Agency developing the collateral`.
- `Expected completion date`.
- `Date approved`.
- `Priority`, which references the constraint task priority.
- `Production status`, which referecnes the task status constraint.
- `Notes`.

The data list comprises the following two assocations:

- `Collateral owner`, note the target class and source definition.
- `Collateral item`, again consider the nature of the target and source elements.

An number of aspects are next added.

- `autoReivew`, defines the automatic review start date. If this aspect is absent the nextRevewDate property value of the collateral type will be used.
- `released`, a date initially based on the proposed publication date property.
- `signoff`, this would be added as part of a workflow, sent to the person completing the task.

The following aspects are used to track specific marketing information about the collateral item:

- `region`, references the regsionList constraint, allowing a collateral item to be designated to a geographical areas.
- `language`, references the languageList constraint.
- `medium`, uses the mediumList constraint, for example print, web, TV, radio.
- `enquiryEmail`, references the enquiryCheck constraint to validate email addresses.

Now the content model is fully updated I will copy and paste this to the Data Dictionary to dynamically deploy it.

User experience

Harriet logs in, and navigates to the Marketing Share site.

The Document Library is used to store the collateral items which support a product launch.

She select the Green Energy Turbine Web image, this is of the Marketing Collateral content type and therefore has review and publication date information as well as a review manager.

She chooses Manage Aspects to add the dynamic aspects which enable further marketing information to be applied to the repository item.

Editing the metadata you witness the constraints in action as they populate the menus of these new aspects.

Moving to the site Data List the server presents a list of new possible data list to create, including the Collateral Tracking list created in the updated content model. She selects this and enters a title.

This data list is shown on the left hand side in the Lists item, which she selects.

You can see a number of the standard title fields presented in the data list are hidden in the Share user interface. This has been achieved through customization of the share-config-custom.xml file. The fields shown are the ones created in the custom data list, defined in the content model.

Next she chooses New Item and enters the detail for the in progress web version graphic for this product.

Choosing Collateral Materials she links this data list item to the first draft which was created internally, this is going to be sent to the agency for further work.

Harriet creates further lists items for this product to track the creation of the other collateral items which the agency will create to support this product launch.

As it is possible to create multiple data lists Harriet creates these to support the different products soon to be launched.

Lab - Creating a content model 5

In this lab you are going to create a custom data list to enable Green Energy to track the status of product certification. When a new product is brought to market it must undergo various tests and pass regulatory standards. For example all wind turbines must undergo acoustic noise tests. The product managers wants to check the current state of compliance to ensure certification is completed in a timely manner prior to a product launch.

1. Update your SOP model to create this data list, ensure you stick to the naming convention given.
    - Review the SOP specification PDF, where you will find the details for the data list to be created.
    - You will require the constraint: taskStatus of type LIST.
        - `taskStatus`: allowedValues = Not Started, In Progress, Pass, Fail
2. Set the data list name to `sop:trackingList`, created with the following properties:
    - `testAgency`
    - `testScheduledDate`
    - `testConductedDate`
    - `testStatus` default=Not Started, references constraint taskStatus
    - `testNotes`
3. The data list should contain two associations
    - `testOwner` source: 0 or more, target exactly 1, class = cm:person

- `testAttachments` source: 0 or 1, taget: 0 or more, class = cm:content

4. Include the following aspect:

```
<mandatory-aspects>
  <aspect>cm:titled</aspect>
</mandatory-aspects>
```

5. Deploy the content model dynamically as administrator.
6. Create a certification Share site where you can create the data list.
7. Create a certification data list and populate this with a number of data list items. Experiment with the possible options and values.
8. If you require assistance the full correct answer for this and all the labs undertaken so far is found in the file: `sopModel.xml` found in the `Desktop > Assets > Creating content models > Final` folder.

Creating content models

Indexing

Each content model property is indexed, alongside content and metadata. By default each property is indexed atomically in the foreground, that is synchronized with committed repository content.

This default indexing can be controlled. The effective property indexing defaults are shown here.

```
<type name="cm:content">
   <title>Content</title>
   <parent>cm:cmobject</parent>
   <archive>true</archive>
   <properties>
      <property name="cm:content">
         <type>d:content</type>
         <mandatory>false</mandatory>

         <index enabled="true">
            <atomic>true</atomic>
            <stored>false</stored>
            <tokenised>true</tokenised>
         </index>

      </property>
   </properties>
</type>
```

The index enabled element toggles indexing on (true) or off (false) for that specific property.

The atomic element dictates foreground indexing in the transaction (true) or background indexing (false).

For the tokenised element, true ensures the string value of the property is tokenised before indexing. False will index the single string as is, both permits the two forms in the index.

Type archive. The final feature of a type definition is to allow control over whether nodes of that type are archived when deleted. Archived nodes may be restored just like the recycle bin of many

operating systems. The reason you don’t often set this is because you will typically be inheriting from cm:content and hence it is already set for you.

The eLearning course 'Introductory Systems Administration' looks at content recycling, storage and indexing in greater detail.

Type inheritance

A type may inherit its definition from another type. All features of the parent type are inherited including property, association, and constraint definitions — except for the parent type name, title, and description.

A type is said to inherit from another type when its parent element is populated with the name of the parent type to inherit. The inheriting type is often referred to as the subtype, while its parent is often referred to as the super-type. Inheritance may be nested, so it is possible to inherit from a type which itself inherits from another type.

Property overrides

Subtypes have the freedom to specify further property, association, and constraint definitions in addition to those inherited from its parent. However, in some cases, it is useful to refine a definition inherited from its parent. For example, a parent type may support an optional property, which the subtype wishes to lock down by mandating its value.

It is not possible to refine all inherited definitions, as it would be very easy to define an incoherent content model. As such, the content metamodel provides a fixed set of refinements known as property overrides.

Each property override is given the same name as the property it wishes to override from its parent type. The following property features may be overridden:

- **mandatory:** A subtype may enforce a property to become mandatory but it cannot relax an existing parent mandatory constraint.
- **default:** A subtype may introduce a default value or change an existing parent default value.
- **constraints:** Additional constraints may be applied to a parent property but existing constraints cannot be modified.

Localization

Every type, aspect, property, association, constraint, and data type defined within a model has a title and description. Both of these values are provided in the model XML file but only one language may be supported: the language of the values specified in the XML file.

To support localization of a model, it is possible to augment the model XML values with locale specific values. This is achieved by registering a standard Java resource bundle for each language variant of a model.

You may be asking why you need to localize content model values. Often, it is required to render user interfaces that are driven from the content model, such as a property sheet that displays a grid of property name and value.

The content models provided out of the box are all augmented with a default (for US English) Java resource bundle.

Best practice

Best practice for content model development can be characterised by following a few simple steps;

1. De-normalize and share aspects
    - There is no limit to depth, but consider 3 levels as a rule of thumb.

- For example, implement and abstract, real and then specialized type.

2. Consider if you need a corporate super-type. It is hard to add retrospectively.
3. Prototyping can be problematic, so invest time in developing your content model well.
4. If you plan to localize follow this development path from the start.

With the material covered you should be ready to start developing your own models in their entirety. Remember that your user interface requires customization to expose your content model, otherwise it is effectively hidden.

Creating content models - Individual lab solutions

Solutions

These are the model solutions for the content modelling exercises, which are also found in the `Desktop > Assets > Creating content models` folder of your virtual lab environment.

Please note that only the part of the content model relevant for each lab is shown below, to see the full solution skip ahead to the SOP Model chapter which shows the content model in its entirety. This is also found in the file **sopModel Final.xml** in the **Labs Solutions** folder.

Lab 1

```
<!--     T Y P E   D E F I N I T I O N S     -->
<types>
    <!-- Definition of new Content Type: Standard Operating Procedure -->
    <type name="sop:sop">
        <title>Standard Operating Procedure</title>
        <parent>cm:content</parent>
        <properties>
            <property name="sop:nextReviewDate">
                <description>Used to initiate automatic reviews for SOPs</description>
                <type>d:date</type>
            </property>

            <property name="sop:reviewDate">
                <type>d:date</type>
            </property>

            <property name="sop:proposedEffectiveDate">
                <description>Entered by the user, to suggest when this SOP should become
                        effective</description>
                <type>d:date</type>
            </property>

            <property name="sop:reviewPeriod">
                <!-- Requires a constraint -->
                <description>Used to initiate automatic reviews for SOPs. For example 365,
                        would mean an SOP is reviewed every year</description>
                <type>d:int</type>
                <default>365</default>
            </property>
        </properties>
    </type>
</types>
```

Lab 2

```
<!--    T Y P E   D E F I N I T I O N S    -->
<types>
    <!-- Definition of new Content Type: Standard Operating Procedure -->
    <type name="sop:sop">
        <title>Standard Operating Procedure</title>
        <parent>cm:content</parent>
        <properties>
            <property name="sop:nextReviewDate">
                <description>Used to initiate automatic reviews for SOPs</description>
                <type>d:date</type>
            </property>

            <property name="sop:reviewDate">
                <type>d:date</type>
            </property>

            <property name="sop:proposedEffectiveDate">
                <description>Entered by the user, to suggest when this SOP should become
                        effective</description>
                <type>d:date</type>
            </property>

            <property name="sop:reviewPeriod">
                <!-- Requires a constraint -->
                <description>Used to initiate automatic reviews for SOPs. For example 365,
                        would mean an SOP is reviewed every year</description>
                <type>d:int</type>
                <default>365</default>
            </property>
        </properties>

        <!-- Associated Objects -->
        <associations>
            <association name="sop:supersededSops">
                <title>Superseded SOPs</title>
                <source>
                    <mandatory>false</mandatory>
                    <many>true</many>
                </source>
                <target>
                    <class>sop:sop</class>
                    <mandatory>false</mandatory>
                    <many>true</many>
                </target>
            </association>

            <association name="sop:owner">
                <title>SOP Owner</title>
                <description>Alias value used in the review process to decide which person
                        initially gets the document.</description>
                <source>
                    <mandatory>false</mandatory>
                    <many>true</many>
                </source>
                <target>
                    <class>cm:person</class>
                    <mandatory>true</mandatory>
                    <many>false</many>
                </target>
            </association>
        </associations>
    </type>
</types>
```

Lab 3

```
<!--      A S P E C T   D E F I N I T I O N S   -->
<aspects>
<!-- This indicates how many days before the next review date this document
     should have an automatic review start. If this aspect is not present it is
     assumed that review starts on the next_review_date. This would be used
     in conjunction with a scheduled job to start a review workflow -->
    <aspect name="sop:autoReview">
        <title>Auto Review</title>
        <properties>
            <property name="sop:autoReviewStart">
                <!-- Requires a constraint -->
                <description>This indicates how many days before the next review
                             date this document should have an automatic review start.
                             If this aspect is not present it is assumed that review
                             starts on the next_review_date</description>
                <type>d:int</type>
                <default>90</default>
            </property>
        </properties>
    </aspect>

    <!-- Typically this would be based on the date that the user initially
         entered in proposed_effective_date on the SOP type,
         and would be automatically set by the workflow process  -->
    <aspect name="sop:effective">
        <title>Effective Date</title>
        <properties>
            <property name="sop:effective_date">
                <!-- Requires a constraint -->
                <description>Typically this would be based on the date that the user initially
                             entered in proposed_effective_date on the SOP type.</description>
                <type>d:date</type>
            </property>
        </properties>
    </aspect>

    <!--  This aspect is added as part of a workflow, during the sign-off task.
          The value will be set to the person completign the task -->
    <aspect name="sop:signoff">
        <title>Signoff</title>
        <associations>
            <association name="sop:signatory">
                <title>Signatory</title>
                <description>This should be added as part of a workflow action,
                             hence it is automatic.</description>
                <target>
                    <class>cm:person</class>
                    <mandatory>false</mandatory>
                    <many>false</many>
                </target>
            </association>
        </associations>
    </aspect>
</aspects>
```

Lab 4

```
<!--       C O N S T R A I N T S       -->
<constraints>
    <constraint name="sop:visibilityList" type="LIST">
        <parameter name="allowedValues">
            <list>
                <value>Internal</value>
                <value>Public</value>
            </list>
        </parameter>
    </constraint>
</constraints>

<!--      T Y P E   D E F I N I T I O N S     -->
<types>
    <!-- Definition of new Content Type: Standard Operating Procedure -->
    <type name="sop:sop">
        <title>Standard Operating Procedure</title>
        <parent>cm:content</parent>

        <!-- ** Existing detail of this type hidden for clarity ** -->

        <properties>
            <property name="sop:visibility">
                <description>Used to indicate if this is SOP is internal use
                             only or for public consumption.</description>
                <type>d:text</type>
                <mandatory>true</mandatory>
                <constraints>
                    <constraint ref="sop:visibilityList" />
                </constraints>
            </property>
        </properties>

        <!-- ** Existing detail of this type hidden for clarity ** -->

  </type>
</types>
```

Lab 5

```
<types>

    <!--    D A T A   L I S T   D E F I N I T I O N S   -->
    <!--    Data list defintions for this model go here -->
    <type name="sop:trackingList">
        <title>Certification Tracking</title>
        <parent>dl:dataListItem</parent>
        <properties>
            <property name="sop:testAgency">
                <title>Agency</title>
                <description>The agency that conducts the test.</description>
                <type>d:text</type>
                <mandatory>true</mandatory>
            </property>

            <property name="sop:testScheduledDate">
                <title>Scheduled Date</title>
                <type>d:datetime</type>
                <mandatory>false</mandatory>
            </property>

            <property name="sop:testConductedDate">
                <title>Conducted Date</title>
                <description>Although user entered to start with, this will in the
                        longer term this may be updated automatically by a workflow.</description>
                <type>d:datetime</type>
                <mandatory>false</mandatory>
            </property>

            <property name="sop:testStatus">
                <title>Status</title>
                <type>d:text</type>
                <default>Not Started</default>
                <constraints>
                    <constraint ref="sop:taskStatus" />
                </constraints>
            </property>

            <property name="sop:testNotes">
                <title>Notes</title>
                <type>d:text</type>
                <mandatory>false</mandatory>
            </property>
        </properties>
```

Continued on following page...

Lab 5 continued

```
        <associations>
            <association name="sop:testOwner">
                <title>Owner</title>
                <source>
                    <mandatory>false</mandatory>
                    <many>true</many>
                </source>
                <target>
                    <class>cm:person</class>
                    <mandatory>true</mandatory>
                    <many>false</many>
                </target>
            </association>

            <association name="sop:testAttachments">
                <title>Test Results</title>
                <source>
                    <mandatory>false</mandatory>
                    <many>false</many>
                </source>
                <target>
                    <class>cm:content</class>
                    <mandatory>false</mandatory>
                    <many>true</many>
                </target>
            </association>
        </associations>
        <!-- the titled aspect is added to provide the name and description
             of the data item -->
        <mandatory-aspects>
            <aspect>cm:titled</aspect>
        </mandatory-aspects>
    </type>
</types>
```

Green Energy - SOP Specification

Standard Operating Proceudre

Types

Object	Standard Operating Procedure
Type	sop *(use this as the type name)*
SuperType	cm :content *(this is the parent type)*
Description	This is a document type specifically for the representation of SOPs

Attribute	Type	Use and Domain	Behaviour and Description	Data Source
nextReviewDate	Date	Any valid date	Used to initiate automatic reviews for SOPs	Automatic/User
proposedEffectiveDate	Date	A future date	Entered by the user on first creation of the SOP. Mandatory.	User
reviewDate	Date	Any valid date	Indicates when the SOP was last reviewed.	Automatic
reviewPeriod	Integer	>0 and <=720 Number of days for review Defaults to 365	Used to initiate automatic reviews. For example 365, would mean an SOP is reviewed every year.	User
visibility	Text	Mandatory	Used to indicate if this is SOP is internal use only or for public consumption.. (Drop down List, implemented via a constraint).	User
owner	Association	Exactly 1 Corresponds to an alias	Alias value used in the review process to decide which person initially gets the document.	User
supersededSops	Association	>= 0 References other valid effective SOPs	This should be a drop down list of SOPs.	User

Constraints

Visibility

Object	Visibility
Constraint	visibilityList
Description	This constraint implements a list menu containing two items.

Attribute	Type	Use and Domain	Behaviour and Description	Data Source
allowedValues	LIST	Values = Internal, Public	Values available on list item.	User

Aspects

Auto Review Start

Object	Auto Review Start
Aspect	autoReview
Description	This aspect is used to initiate automatic reviews. This indicates how many days before the next review date this document should have an automatic review start. If this aspect is not present it is assumed that review starts on the nextReviewDate.

Attribute	Type	Use and Domain	Behaviour and Description	Data Source
autoReviewStart	Integer	>0 <90 number of days defaults to 90	A number of days, should be constrained.	User

Effectivity

Object	Effectivity
Aspect	effective
Description	This aspect is used to show the date when this SOP becomes effective, if not present then SOP not effective.

Attribute	Type	Use and Domain	Behaviour and Description	Data Source
effective_date	date	Date the SOP became effective		User

Sign off

Object	Sign off
Aspect	signoff
Description	This aspect is used to indicate that this SOP has been signed off and who completed the action.

Attribute	Type	Use and Domain	Behaviour and Description	Data Source
signatory	user	A user from the system 0 or 1	This should be added as part of a workflow action, hence it is automatic.	Automatic

Data Lists

Test list

Object	Test Tracking List
Type	trackingList
Parent	dl:dataListItem
Description	This type is a new data list type. When a new product is brought to market it has to undergo various tests and certifications before the regulatory authorities will allow its sale. For example all wind turbine products must undergo a system acoustic noise test and come in under the acceptable limits. The product manager wants to be able to track the current state of compliance in order that he can ensure that all the tests and certificates can be acquired in a timely manner before the product is released. This is important because of the tests must be passed in order for the product to be able to ship.

Attribute	Type	Use and Domain	Behaviour and Description	Data Source
title	Aspect	Add cm:titled	Mandatory. The titled aspect is added to provide the name and description of the data item, and is the test/certification name	User
testAgency	Text	Free text	Mandatory	User
testScheduledDate	Date/Time	A date		User
testConductedDate	Date/Time	A date	Can only be updated/edited not added on creation	User
testStatus	Text	Possible values are: Not Started, In Progress, Passed, Failed	Defaults to Not Started	User
testNotes	Text	Free text		User
testOwner	Association	A user from the system. Corresponds to cm:person	Mandatory	User
testAttachments	Association	The documents being tracked. Corresponds to cm:content	One or more document attachments, for example full test report, certificates, etc.	User

Green Energy - SOP Full XML solution

```
<?xml version="1.0" encoding="UTF-8"?>

   <!--  ___________________________________________ -->
   <!--  Alfresco Training - All Courses             -->
   <!--  Final Solution - Content Modeling           -->
   <!--  (set tab width to 4 to view this file)      -->
   <!--                                              -->
   <!--  This is the full and final solution to      -->
   <!--  the content modeling exercise containing    -->
   <!--  the following parts:                        -->
   <!--  - Types: a type sub-classed off             -->
   <!--    cm:content and one a sub-type of          -->
```

```
   <!--    dl:dataListItem                          -->
   <!--  - Aspects: various aspects which can be    -->
   <!--  attached to the types                      -->
   <!--  - Constraints: one for a type and one      -->
   <!--    for a datalist                           -->
   <!-- ___________________________________________ -->
   <!--  Alfresco Training                          -->
   <!--  27 September 2012                          -->
   <!--  Created: 2010-08-06                        -->
   <!--  Revised: 2012-10-01, 2011-06-24            -->
   <!--  Copyright Alfresco 2010-2012               -->
   <!-- ========================================== -->

<!-- The important part here is the name - Note: the use of the sop: namespace
 which is defined further on in the document -->
<model  name="sop:sopmodel" xmlns="http://www.alfresco.org/model/
dictionary/1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.alfresco.org/model/dictionary/1.0/
modelSchema.xsd">

 <!-- Meta-data about the model -->
 <description>Green Energy Model for Standard Operating Procedures</
description>
 <author>Alfresco Training</author>
 <version>2.1</version>

 <!-- Imports are required to allow references to definitions in other models
 -->
 <imports>
  <!-- Import Alfresco Dictionary Definitions -->
  <import uri="http://www.alfresco.org/model/dictionary/1.0" prefix="d"/>
  <!-- Import Alfresco Content Domain Model Definitions -->
  <import uri="http://www.alfresco.org/model/content/1.0" prefix="cm"/>
  <!-- Import Alfresco Data List Model Definitions -->
  <import uri="http://www.alfresco.org/model/datalist/1.0" prefix="dl"/>
 </imports>

 <!-- Introduction of new namespaces defined by this model -->
 <!-- NOTE: The following namespace "sop.green-energy-demo.com" should be
 changed to reflect your own namespace -->
 <namespaces>
  <namespace uri="sop.green-energy-demo.com" prefix="sop"/>
 </namespaces>

 <!--  C O N S T R A I N T S  -->
 <constraints>
  <constraint name="sop:visibilityList" type="LIST">
   <parameter name="allowedValues">
    <list>
     <value>Internal</value>
     <value>Public</value>
    </list>
   </parameter>
  </constraint>

  <!--  Used in datalists  -->
  <constraint name="sop:taskStatus" type="LIST">
   <parameter name="allowedValues">
    <list>
     <value>Not Started</value>
     <value>In Progress</value>
     <value>Pass</value>
     <value>Fail</value>
    </list>
   </parameter>
  </constraint>

  <!--  Requested in the datamodel Specifications -->
```

```
 <constraint name="sop:reviewPeriodRange" type="MINMAX">
  <parameter name="minValue">
   <value>1</value>
  </parameter>
  <parameter name="maxValue">
   <value>720</value>
  </parameter>
 </constraint>

 <constraint name="sop:autoReviewStartRange" type="MINMAX">
  <parameter name="minValue">
   <value>1</value>
  </parameter>
  <parameter name="maxValue">
   <value>90</value>
  </parameter>
 </constraint>
</constraints>


<!--  T Y P E   D E F I N I T I O N S  -->
<types>

 <!-- Definition of new Content Type: Standard Operating Procedure -->
 <type name="sop:sop">
  <title>Standard Operating Procedure</title>
  <parent>cm:content</parent>
  <properties>
   <property name="sop:nextReviewDate">
    <description>Used to initiate automatic reviews for SOPs</description>
    <type>d:date</type>
   </property>

   <property name="sop:reviewDate">
    <type>d:date</type>
   </property>

   <property name="sop:proposedEffectiveDate">
    <description>Entered by the user, to suggest when this SOP should become
effective</description>
    <type>d:date</type>
   </property>

   <property name="sop:reviewPeriod">
   <!-- Requires a constraint -->
    <description>Used to initiate automatic reviews for SOPs. For example 365,
would mean an SOP is reviewed every year</description>
    <type>d:int</type>
    <default>365</default>
    <constraints>
     <constraint ref="sop:reviewPeriodRange"/>
    </constraints>
   </property>

   <property name="sop:visibility">
    <description>Used to indicate if this is SOP is internal use only or for
public consumption.</description>
    <type>d:text</type>
    <mandatory>true</mandatory>
    <constraints>
     <constraint ref="sop:visibilityList"/>
    </constraints>
   </property>
  </properties>

  <!-- Associated Objects -->
  <associations>
   <association name="sop:supersededSops">
```

```
    <title>Superseded SOPs</title>
    <source>
     <mandatory>false</mandatory>
     <many>true</many>
    </source>
    <target>
     <class>sop:sop</class>
     <mandatory>false</mandatory>
     <many>true</many>
    </target>
   </association>

   <association name="sop:owner">
    <title>SOP Owner</title>
    <description>Alias value used in the review process to decide which person
initially gets the document.</description>
    <source>
     <mandatory>false</mandatory>
     <many>true</many>
    </source>
    <target>
     <class>cm:person</class>
     <mandatory>true</mandatory>
     <many>false</many>
    </target>
   </association>
 </associations>
</type>


<!-- D A T A   L I S T   D E F I N I T I O N S -->
<!-- Data list defintions for this model go here -->
<type name="sop:trackingList">
 <title>Certification Tracking</title>
 <parent>dl:dataListItem</parent>

 <properties>
  <property name="sop:testAgency">
   <title>Agency</title>
   <description>The agency that conducts the test.</description>
   <type>d:text</type>
   <mandatory>true</mandatory>
  </property>

  <property name="sop:testScheduledDate">
   <title>Scheduled Date</title>
   <type>d:datetime</type>
   <mandatory>false</mandatory>
  </property>

  <property name="sop:testConductedDate">
   <title>Conducted Date</title>
   <description>Although user entered to start with, this will in the longer
term this may be updated automatically by a workflow.</description>
   <type>d:datetime</type>
   <mandatory>false</mandatory>
  </property>

  <property name="sop:testStatus">
   <title>Status</title>
   <type>d:text</type>
   <default>Not Started</default>
   <constraints>
    <constraint ref="sop:taskStatus" />
   </constraints>
  </property>

  <property name="sop:testNotes">
```

```
   <title>Notes</title>
   <type>d:text</type>
   <mandatory>false</mandatory>
  </property>
 </properties>

 <associations>
  <association name="sop:testOwner">
   <title>Owner</title>
   <source>
    <mandatory>false</mandatory>
    <many>true</many>
   </source>
   <target>
    <class>cm:person</class>
    <mandatory>true</mandatory>
    <many>false</many>
   </target>
   </association>
    <association name="sop:testAttachments">
     <title>Test Results</title>
     <source>
      <mandatory>false</mandatory>
      <many>false</many>
     </source>
     <target>
      <class>cm:content</class>
      <mandatory>false</mandatory>
      <many>true</many>
     </target>
    </association>
   </associations>

<!-- the titled aspect is added to provide the name and description of the
data item -->
  <mandatory-aspects>
   <aspect>cm:titled</aspect>
  </mandatory-aspects>
 </type>
</types>

<!--   A S P E C T   D E F I N I T I O N S  -->
<aspects>
 <!-- This indicates how many days before the next review date this document
should have an automatic review start.
      If this aspect is not present it is assumed that review starts on the
next_review_date. This would be used
      in conjunction with a scheduled job to start a review workflow -->
 <aspect name="sop:autoReview">
  <title>Auto Review</title>
  <properties>
   <property name="sop:autoReviewStart">
    <!-- Requires a constraint -->
    <description>This indicates how many days before the next review date this
document should have an automatic review start. If this aspect is not present
it is assumed that review starts on the next_review_date</description>
    <type>d:int</type>
    <default>90</default>
    <constraints>
     <constraint ref="sop:autoReviewStartRange"/>
    </constraints>
   </property>
  </properties>
 </aspect>

 <!--  Typically this would be based on the date that the user initially
entered in proposed_effective_date on the SOP type, and would be automatically
set by the workflow process -->
```

```
  <aspect name="sop:effective">
   <title>Effective Date</title>
   <properties>
    <property name="sop:effective_date">
    <!-- Requires a constraint -->
     <description>Typically this would be based on the date that the user
 initially entered in proposed_effective_date on the SOP type.</description>
     <type>d:date</type>
    </property>
   </properties>
  </aspect>

  <!-- This aspect is added as part of a workflow, during the sign-off task.
 The value will be set to the person completign the task -->
  <aspect name="sop:signoff">
  <title>Signoff</title>
  <associations>
   <association name="sop:signatory">
    <title>Signatory</title>
    <description>This should be added as part of a workflow action, hence it is
 automatic.</description>
    <target>
     <class>cm:person</class>
     <mandatory>false</mandatory>
     <many>false</many>
    </target>
   </association>
  </associations>
 </aspect>
</aspects>
</model>
```

Developing applications

Introduction

The Alfresco product provides several out-of-the-box applications that interact with the core Alfresco repository through well-defined interfaces and services. These applications provide interaction with content and business logic from the repository to deliver solutions for Enterprise Content Management, such as Document Management, Records Management, and Web Content Services.

The product suite leverages open standards and well-understood service interfaces to provide valuable options for customization and extension. In this section we focus on the extension points and on how they are commonly used to build custom integrations and solutions.

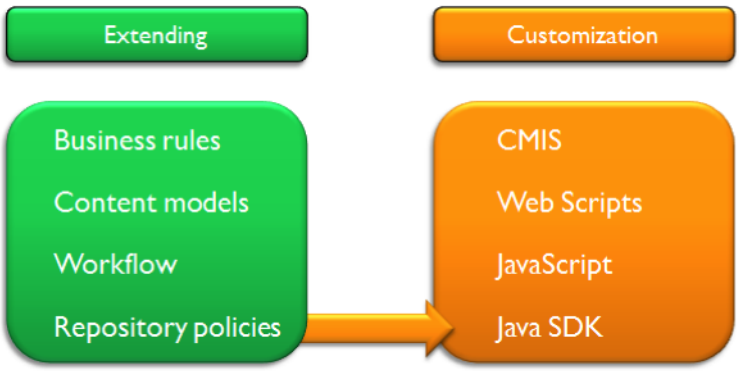
As such this acts as a precursor to the Alfresco development courses where you will learn more about these topics.

This Alfresco Element will explore the different approaches to application development, help you to decide on a good application development approach, examine the different APIs available and access and leverage the source code.

Extending and customizing

Alfresco offers a range of ways to extend or customize the system, when you get to repository policies you are looking at moving towards a customization approach, indeed even implementing content models requires some additional effort for the user interface which may arguably called customization.

Ultimately you can of course go directly to the source code of an open source product like Alfresco. But this is not recommended unless you never want to upgrade.



Development environment

The choice of development languages to use is yours.

You can use various frontends to access Alfresco; such as Liferay and Drupal and do your development work in those systems if this is your preferred platform.

There are many development tools which you can use in conjunction with Alfresco the most commonly used one being the Eclipse IDE.

Whatever your choice there will be some way to integrate. Some interfaces are more well developed and supported than others, for instance support for PHP is available but rudimentary. Alfresco is written in Java and there is excellent support for both Java and JavaScript, with this being the choice of many Alfresco developers.

Alfresco also makes extensive use of frameworks such as Spring, Spring Surf, YUI Library (the Yahoo User Interface) as well as other existing Open Source technologies and projects such as Solr and Quartz.

The architecture of Alfresco was explored in the architecture and technologies Alfresco Element. Lets know explore the integration points and widely used interfaces.

Interfaces for Share

As Share is used as the primary user interface by such a large number of people we focus on this first.

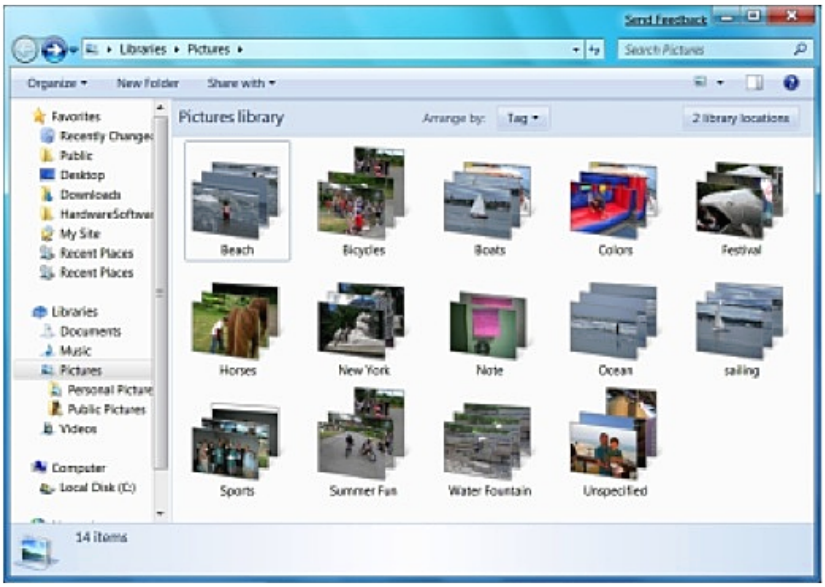
Share is built on the Spring Surf framework and this framework is available to you. The framework in turn provides connectors for accessing the repository through a Web Scripts based REST style interface.

Recently the CMIS standard has been ratified and Alfresco provides a compliant CMIS interface which can be used as Web Scripts or through SOAP. The extended superset of CMIS, OpenCMIS, (part of the Apache Chemistry project) is available through client side Java APIs.

User interface development

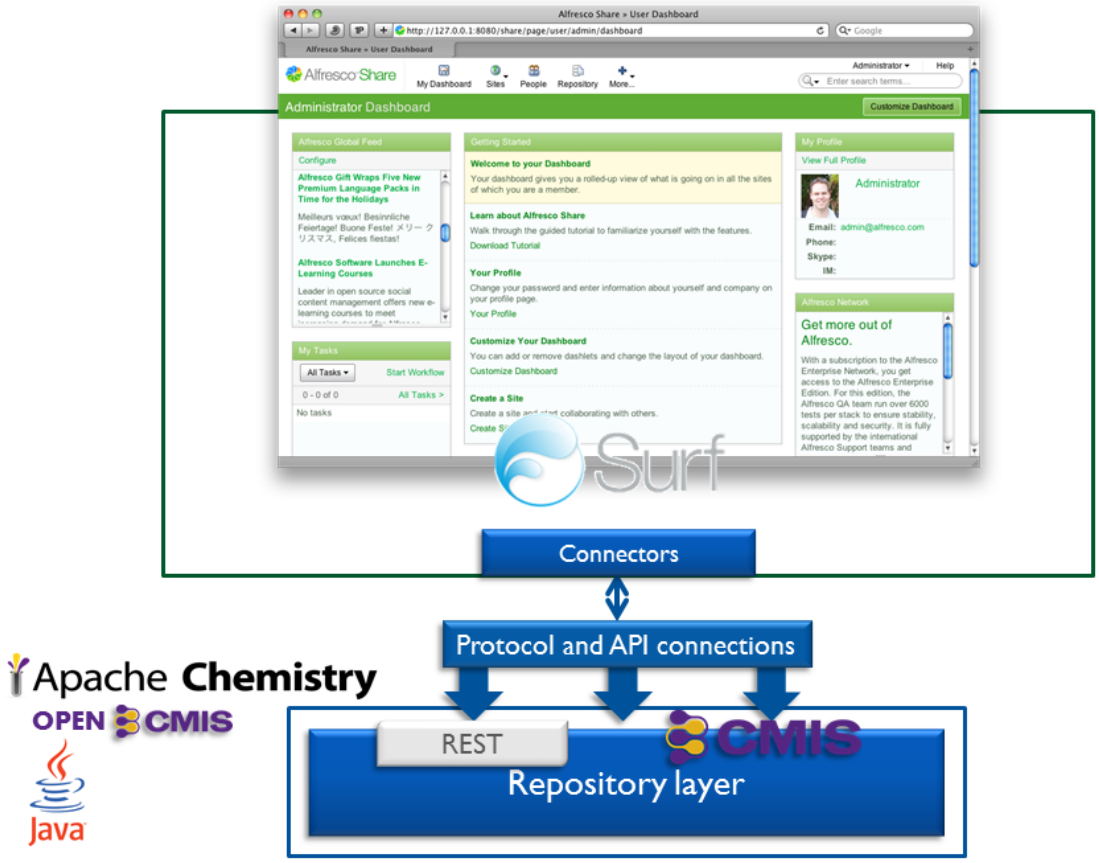
When you need to undertake user interface development you have a number of options.

You can write your own interface to Alfresco in your preferred development environment and then use the appropriate connector for that environment. This approach gives you the most flexibility allowing you to build a user interface which completely matches your needs in whatever technology you want. Using this approach, people have developed user interface applications for Alfresco based on Adobe Air and for the Apple iPhone to name but two. This is also the approach you would use if you wanted to integrate in another different front-end for example a portal such as Liferay or Drupal. With this approach you can develop both thin and thick client applications.



The second approach uses the Spring Surf framework, this framework was created by Alfresco for developing model-view-controller (MVC) type web applications and works very well with Alfresco. This provides some structure for your application and eases the programming development burden.

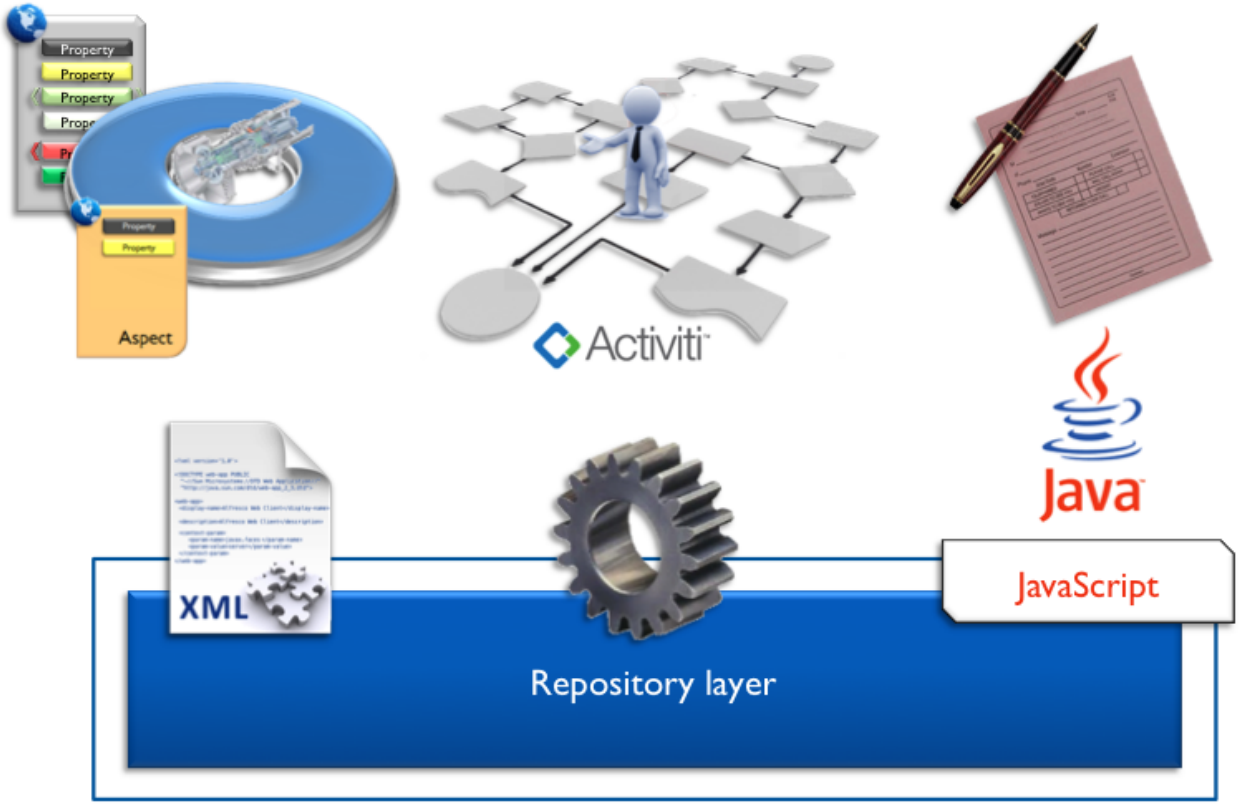
Finally you can take the Share user interface, which itself is built on Surf, and customize this interface to your requirements. Using this approach can get you further more quickly because you can use the work which Alfresco has already put into the user interface. You can take what we provide and extend and tailor to your own requirements. Whilst there is a lot you can do using this approach you are ultimately restricted to the overall look and feel and structure which Share imposes.



Server side development

On the server side you also have some development options, the one most commonly used, is the addition of content models into the server. The second way of extending the server is by adding process or workflow definitions using Activiti.

The server also has a mechanism whereby you can extend its behavior through policies. This is done by using Java and a JavaScript API is also available server side which can be used within workflows, for example.



Alfresco Public API

The Alfresco public API allows developers to create custom applications (desktop, mobile, or cloud) that persists content to Alfresco in the Cloud. The API includes CMIS and some Alfresco REST calls where CMIS does not provide the functionality.

Any kind of application can be written which also affords integration with applications in the cloud.

This requires registration at this address: http://developer.alfresco.com

Once registered you can add applications to your profile. Each application has a unique authentication key and a secret. OAuth2 is used to handle authentication.

Presently this is restricted to Alfresco in the cloud, however integration to on-premise Alfresco Enterprise will come soon.

Source code

Whilst it is only the Community Edition source code which is available to you, this code is updated with bug fixes from the Enterprise Edition. The Community Edition runs ahead of the Enterprise Edition in terms of new features and functionality.

The Alfresco Community source code is stored in a Subversion repository. For more information see these two links.

- http://wiki.alfresco.com/wiki/Source_Code
- http://wiki.alfresco.com/wiki/Development_Environment

Additional Training References

For additional information on customizing Alfresco to meet your specific needs see the links below.

Alfresco Training

- *Learning Resources:* http://university.alfresco.com/resources.html

Alfresco Platform

- *Documentation:* http://www.alfresco.com/resources/documentation
- *How to Contribute:* http://wiki.alfresco.com/wiki/How_to_Contribute
- *Alfresco Developers Guide:* http://wiki.alfresco.com/wiki/Developer_Guide
- *Node References:* http://wiki.alfresco.com/wiki/Node_References_and_Store_Protocols

Alfresco Cookbooks

- *Node Ref:* http://wiki.alfresco.com/wiki/NodeRef_cookbook
- *JavaScript API:* http://wiki.alfresco.com/wiki/JavaScript_API_Cookbook
- *Web Scripts:* http://wiki.alfresco.com/wiki/Web_Scripts
- *FreeMarker:* http://wiki.alfresco.com/wiki/FreeMarker_Template_Cookbook
- *Content Modeling:* http://wiki.alfresco.com/wiki/Data_Dictionary_Guide
- *Constraints:* http://wiki.alfresco.com/wiki/Constraints

Jeff Potts Tutorials

- *Content Models article (2nd edition):* http://ecmarchitect.com/images/articles/alfresco-content/content-article-2ed.pdf
- *Developing custom actions(2nd Edition):* http://ecmarchitect.com/images/articles/alfresco-actions/actions-article-2ed.pdf
- *Implementing Custom behaviour:* http://ecmarchitect.com/images/articles/alfresco-behavior/behavior-article.pdf
- *Intro to the Web Script Framework:* http://ecmarchitect.com/images/articles/alfresco-webscripts/web-script-article.pdf
- *Advanced Workflow (2nd Edition):* http://ecmarchitect.com/images/articles/alfresco-workflow/advanced-workflow-article-2ed.pdf
- *Getting Started with CMIS:* http://ecmarchitect.com/images/articles/cmis/cmis-article.pdf

Alfresco Add-ons and Extras

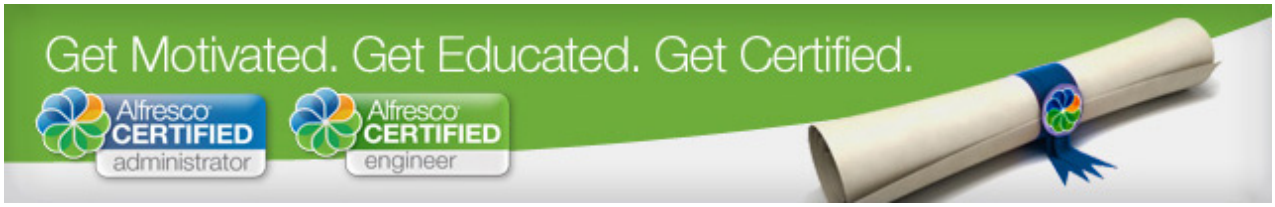
- *Share Extras:* http://code.google.com/p/share-extras/
- *Alfresco Add-Ons:* http://addons.alfresco.com/
- *Alfresco Model Designer:* http://addons.alfresco.com/addons/alfresco-model-designer
- *Form Model Management:* http://addons.alfresco.com/addons/alfresco-form-model-management
- *Change Share Locale:* http://addons.alfresco.com/addons/change-share-locale

VM Ware

- *Common Issues:*http://communities.vmware.com/docs/DOC-8978

- *BIOS Error - This virtual machine is configured for 64-bit guest operating systems. However, 64-bit operation is not possible!:*http://blog.creativeitp.com/posts-and-articles/bios/bios-relevant-vmware-error-this-virtual-machine-is-configured-for-64-bit-guest-operating-systems-however-64-bit-operation-is-not-possible

Alfresco Certification Program



We now have a new program in place to formally recognize the growing number of Alfresco engineers and administrators in the workplace. Alfresco Certification is a new standard which indicates a proven level of technical proficiency on the Alfresco platform.

Exams

The currently available open-enrolment exam certifications are:

- Alfresco Certified Engineer (ACE)
- Alfresco Certified Administrator (ACA)

Which one you aim for will depend on your background, job function and career goals. To certify, you will need to pass the relevant exam with a score of at least 70%. Find out more about what is assessed in each exam by reviewing the appropriate blueprint.

You can book in for the exam with our accreditation partner, Pearson Vue. They have over 5,000 test centres in 165 countries and can tell you the nearest centre to you, when you contact them to arrange an appointment.

If you don't feel ready quite yet, you may want to take some training. We have lots of courses that will help you learn more Alfresco and free webinars where you can see exactly how others have built brilliant content centric applications on top of the Alfresco platform.

Certifying as an Alfresco Administrator or an Engineer is a great career move. It bolsters your resume, increases your credibility, and will help you build and manage better content management stores.

Schedule an Exam

Exams are conducted at Pearson VUE testing facilities.

For further information on certification or to schedule an exam please visit: `http://university.alfresco.com/certification.html`