

Introduction

This page describes how to configure authentication in Alfresco versions 3.2 and greater. In version 3.2, this area underwent a major reorganization with the goals of:

- simplifying the configuration of the most common types of authentication
- allowing you to easily 'mix and match' different types of authentication
- clearly separating configuration from product
- consistently using a single simple configuration mechanism

What emerged were the *Authentication Subsystems* described on this page.

This page now replaces the following pages. Refer to these pages only if you are configuring authentication in Alfresco versions prior to 3.2:

- [Enterprise Security and Authentication Configuration](#)
- [CIFS Server Authentication](#)
- [3.0_Configuring_NTLM](#) - NTLM configuration for Alfresco 3.0-3.1
- [Configuring NTLM](#) - NTLM configuration for Alfresco versions prior to 3.0
- [Configuring the CIFS and web servers for Kerberos/AD integration](#)
- [SSO](#)

What is an Authentication Subsystem?

An authentication subsystem is a coordinated stack of compatible components responsible for providing authentication-related functionality to Alfresco. Alfresco offers multiple alternative implementations of authentication subsystem, each engineered to work with one of the different types of back-end authentication server that you may have available in your enterprise.

Authentication is just one of the many [Alfresco subsystems](#) introduced in version 3.2. The subsystem pattern lends itself particularly well to the world of authentication. In contrast with earlier releases:

- Subsystems for all supported authentication types are pre-wired and there is no need to edit template configuration.
- There is no danger of compatibility issues between sub-components, as these have all been pre-selected for you. For example, your CIFS authenticator and authentication filter is compatible with your authentication component.
- Common parameters are shared and specified in a single place. There is no need to specify the same parameters to different components in multiple configuration files.
- There is no need to edit web.xml. web.xml now uses generic filters that call into the authentication subsystem. The alfresco.war file is now a more portable unit of deployment.
- You can easily swap from one type of authentication to another by activating a different authentication subsystem.
- Your authentication configuration remains standard and will be a 'known quantity' for Alfresco support and other Alfresco experts.
- Authentication subsystems are easily chained.

What are the Authentication Subsystems?

The following table lists the Authentication subsystem types supplied with Alfresco and the optional features they support. Note that if you configure a single authentication subsystem of a type that doesn't support CIFS authentication (such as ldap), the CIFS server will automatically be disabled. If you want CIFS *and* LDAP, you add more than one subsystem to the authentication chain. Refer to [The Alfresco Authentication Chain](#) for more detail. For further information on each of these subsystem types and the configuration options they support, refer to the corresponding section in the [Authentication Subsystem Reference](#).

| Type | Description | Single Sign-On (SSO) ? | CIFS Authentication? | User Registry Export? |
|------------------------------|---|------------------------|----------------------|-----------------------|
| alfrescoNtlm | Native Alfresco authentication | Yes, NTLM | Yes | No |
| ldap | Authentication and user registry export via the LDAP protocol (e.g. OpenLDAP) | No | No | Yes |
| ldap-ad | Authentication and user registry export from Active Directory via the LDAP protocol | No | No | Yes |
| passthru | Authentication via a Windows domain server | Yes, NTLM (v1 only) | Yes | No |
| kerberos | Authentication via a Kerberos Realm | Yes, SPNEGO | Yes | No |
| external | Authentication via an external SSO mechanism | Yes | No | No |

What's in an Authentication Subsystem?

The main components of an authentication subsystem are:

An authentication component

Handles the specifics of talking to the back-end authentication system.

An authentication Data Access Object (DAO)

Decides what user management functions are allowed, if any (such as the ability to create a user).

An authentication service

Wraps the authentication component and DAO with higher-level functions. Refer to [The Authentication Service](#) for more detail.

A user registry export service (optional)

Exposes information about users and groups to the [Synchronization Subsystem](#)

Authentication Filters

Provide form or SSO-based login functions for

- Web Client
- Web DAV
- Web Scripts
- SharePoint Protocol

File server authenticators

Provide authentication functions for

- CIFS protocol (optional)
- FTP protocol

Location

The Alfresco Subsystems are located in `tomcat/webapps/alfresco/WEB-INF/classes/alfresco/subsystems`

The Alfresco Authentication Chain

It's very likely that at least one of the authentication subsystem types would allow you to integrate Alfresco with one of the authentication servers in use in your enterprise. However, integrating Alfresco with just one of these systems may not be enough; for various reasons you might want to 'mix and match' multiple authentication protocols against a collection of servers.

For this reason, Alfresco has a built-in *authentication chain*. In simple terms, this is a priority-ordered list of authentication subsystem instances. Alfresco composes together the functions of the subsystems in this list into a more powerful conglomerate.

Authentication Functions

The functions of the chain are composed in two different ways.

Chained Functions

Chained functions combine functions of more than one subsystem in the chain together.

For example, when a user logs in, Alfresco tries the user's credentials against each of the subsystems in the chain in the order specified.

- If a chain member accepts the credentials, the login succeeds.
- If no chain member accepts, the login fails.

User registry export is also chained. During a synchronize operation, users and groups are exported from each member of the chain supporting user registry export (such as those of type `ldap` or `ldap-ad`) and imported into Alfresco. Ordering in the chain is used to resolve 'collisions' between users and groups existing in the same directory. Refer to [The Synchronization Subsystem](#) for more detail.

Passthru functions

These functions cannot be chained and instead pass through to a single member of the chain, which handles them directly.

Examples of passthru functions are:

- NTLM / SPNEGO - based Single Sign-On (SSO)
- CIFS Authentication

Such passthru functions are handled by the first member of the chain that supports that function and has it enabled. This means that only a subset of your user base may be able to use SSO and CIFS.

The Default Authentication Chain

The default product configuration has a simple chain with one member. This is an instance of the `alfrescoNtlmSubsystem` type with ID `alfrescoNtlm1`.

This is expressed in the built-in defaults (in `tomcat/webapps/alfresco/WEB-INF/classes/alfresco/repository.properties`) as:

```
authentication.chain=alfrescoNtlm1:alfrescoNtlm
```

To configure the properties of `alfrescoNtlm1`, you can use one of the methods described in [Configuring Subsystems](#).

This subsystem instance does not have SSO enabled by default.

To switch from password-based login to NTLM-based SSO, set the following property (in `alfresco-global.properties` or using JMX).

```
ntlm.authentication.sso.enabled=true
```

Note: You do not need to change `web.xml`, unlike previous versions of Alfresco.

When using `alfrescoNtlm`, the passwords are stored locally in the alfresco SQL database as MD4 hashes. To get `alfrescoNtlm` to work with SSO, the idea is to have your windows client authenticate with the same password as the one stored in SQL. This technique leads however to passwords synchronization issues. There is no simple way when using `alfrescoNtlm` to have SSO without having password synchronization issues; That's why when SSO is required, you usually have to choose another authentication subsystem, like 'passthru', 'kerberos' or 'external'. For direct authentication with a Windows domain server without the need to synchronize accounts in Alfresco and the domain, see e.g. the `passthruSubsystem` type.

Configuring the Authentication Chain

You can easily add to or completely replace the default authentication chain.

The chain is controlled by the special `authentication.chain`, which is found in the `Alfresco-global.properties` file.

This is a comma separated list of the form:

```
instance_name1:type1,...,instance_nameN:typen
```

So, for example, if you set this property to the following value, by navigating to the `Alfresco:Type=Configuration,Category=Authentication,id=manager` MBean in JConsole for the Enterprise version, or using one of the methods described in [Global Property Overrides](#)

```
alfrescoNtlm1:alfrescoNtlm,ldap1:ldap
```

then a new authentication subsystem instance called `ldap1` is created and added to the end of the authentication chain.

You can then configure values for the various properties of `alfrescoNtlm1` and `ldap1` as in [Configuring Subsystems](#).

Example 1: Advanced AD Chain

If you want to integrate Alfresco with Active Directory with the following requirements:

- Built-in Alfresco users and Windows users should be able to log in, with Alfresco taking precedence
- The Windows domain server should handle CIFS authentication directly
- LDAP should be used to synchronize user and group details

Configure the following authentication chain:

```
alfrescoNtlm1:alfrescoNtlm,passthru1:passthru,ldap1:ldap-ad
```

You then deactivate SSO in order to activate chained password-based login, target CIFS at `passthru1` and target synchronization (but not authentication) at `ldap1`. Set the properties using JMX as follows:

alfrescoNtlm1

```
ntlm.authentication.sso.enabled=false
alfresco.authentication.authenticateCIFS=false
```

passthru1

```
ntlm.authentication.sso.enabled=false
passthru.authentication.authenticateCIFS=true
```

ldap1

```
ldap.authentication.active=false
ldap.synchronization.active=true
```

Note 1: Although it is easy to chain authentication subsystems, not all protocols can be chained. For instance, in the requirement of the example, you authenticated CIFS against the AD server. Having a failover mechanism that tries first Alfresco database and then asks the AD server if that fails does not work. This is because CIFS authentication using NTLM protocol, which is a challenge-response protocol, and the client won't do two authentication attempts in a chain. In other words, you cannot have the following at the same time:

```
alfresco.authentication.authenticateCIFS=true
```

and

```
passthru.authentication.authenticateCIFS=true
```

Only one of the two can be set to 'true'.

Note 2: One can turn on SSO on one of the auth components for instance `passthru`, to get the following set of parameters values:

alfrescoNtlm1

```
ntlm.authentication.sso.enabled=false
alfresco.authentication.authenticateCIFS=false
```

passthru1

```
ntlm.authentication.sso.enabled=true
passthru.authentication.authenticateCIFS=true
```

ldap1

```
ldap.authentication.active=false
ldap.synchronization.active=true
```

That way, SSO is given to AD users. Local (AlfrescoNtlm) users will still be able to login, but they will need to login using a (backdoor)

URL: <http://localhost:8080/alfresco/faces/jsp/login.jsp> where the login form is displayed, even when SSO is turned on.

Example 2: Advanced LDAP Chain

You want to integrate Alfresco with two LDAP directories with the following requirements:

- User passwords should be validated directly against the LDAP servers for web, Sharepoint and FTP login
- The CIFS server should be deactivated, since neither server can handle CIFS-style authentication
- LDAP should be used to synchronize user and group details from both directories
- Users in the first directory, `ldap1`, should take precedence over those in the second directory, `ldap2`
- We want to use file-based configuration methods rather than JMX

For this, use the mechanism described in [Alfresco Subsystems](#) to configure different property sets for the two LDAP subsystem instances.

Include the following in `alfresco-global.properties`:

```
authentication.chain=ldap1:ldap,ldap2:ldap
```

Then copy `$TOMCAT_HOME/webapps/alfresco/WEB-INF/classes/alfresco/subsystems/Authentication/ldap/ldap-authentication.properties` to

```
$TOMCAT_HOME/shared/classes/alfresco/extension/subsystems/Authentication/ldap/ldap1/ldap-authentication.properties
```

and

```
$TOMCAT_HOME/shared/classes/alfresco/extension/subsystems/Authentication/ldap/ldap2/ldap-authentication.properties
```

You can then edit the properties for `ldap1` and `ldap2` as required to complete the configuration. Use the [LDAP Configuration](#) section as a reference for each of the properties.

Example 3: Configure Authentication subsystem in "shared" folder

To configure an Authentication subsystem (e.g.: `myldap`), in "shared" folder (so upgrading Alfresco you will not lost personalizations) :

- copy this folder

```
$TOMCAT_HOME/webapps/alfresco/WEB-INF/classes/alfresco/subsystems/Authentication/ldap
```

to this folder

```
$TOMCAT_HOME/shared/classes/alfresco/subsystems/Authentication
```

(create the folder tree if not exist)

- rename folder

```
$TOMCAT_HOME/shared/classes/alfresco/subsystems/Authentication/ldap
```

to

```
$TOMCAT_HOME/shared/classes/alfresco/subsystems/Authentication/ldap/myldap
```

- Edit file

```
$TOMCAT_HOME/shared/classes/alfresco/subsystems/Authentication//ldap/myldap/ldap-authentication.properties
```

for standard configuration.

- Edit file

```
$TOMCAT_HOME/shared/classes/alfresco/subsystems/Authentication/ldap/myldap/ldap-authentication-context.xml
```

replace :

```
<import resource="../../common-ldap-context.xml" />
```

with :

```
<import resource="classpath*:alfresco/subsystems/Authentication/common-ldap-context.xml" />
```

- Modify **authentication.chain** adding **myldap:myldap** (e.g.:*authentication.chain=myldap:myldap,alfrescoNtlm:alfrescoNtlm*)
- Restart Alfresco

Authentication Subsystem Reference

The following sections provide a detailed reference guide to configuring and using each of the authentication subsystems. Thanks to the many people who have contributed and continue to contribute to the body of wisdom in these sections.

AlfrescoNtlm

`alfrescoNtlm` is the subsystem configured by default in the Alfresco Authentication Chain. It performs authentication based on user and password information stored in the Alfresco repository. It is capable of supporting both form-based login and NTLM-based Single Sign-On (SSO) as well as providing authentication for the CIFS server.

Note that by default the NTLM SSO functions are disabled by default, thus not making any assumptions about the availability of a Windows Domain. However, activating SSO only requires the editing of a single property (see [Configuration](#)) and no longer requires any changes to `web.xml` or further file server configuration as in earlier Alfresco versions.

Note that this subsystem has *nothing* to do with Active Directory or LDAP! A better way to think of it is simply as the Alfresco native authentication subsystem that happens to support CIFS (via NTLM).

NTLM

The `alfrescoNtlm` subsystem supports optional NTLM Single Sign-On (SSO) functions for WebDAV and the Alfresco Explorer client. Note that since Alfresco 3.0, NTLM v2 is supported which is more secure than the NTLM v1 previously supported. NTLMv2 will automatically downgrade to NTLMv1 if the client does not support it.

By using NTLM authentication to access Alfresco Explorer and Alfresco WebDAV sites the web browser can automatically log in.

Internet Explorer will use your Windows logon credentials when requested by the web server when SSO is enabled. Firefox and Mozilla also support the use of NTLM but you need to add the URI to the Alfresco site that you want to access to `network.automatic-ntlm-auth.trusted-uris` option (available through writing `about:config` in the URL field) to allow the browser to use your current credentials for login purposes.

The Opera web browser does not currently support NTLM authentication, the browser is detected and will be sent to the usual Alfresco logon page.

Note in this configuration, Alfresco must still store its own copy of your MD4 password hash. In order to remove this need and authenticate directly with a Windows domain controller, consider the [Passthru](#) subsystem.

Configuration

The `alfrescoNtlm` subsystem supports the following properties. See [Configuring Subsystems](#) for how to configure these.

`ntlm.authentication.sso.enabled`

A Boolean that when `true` enables NTLM based Single Sign On (SSO) functionality in the Web clients. When `false` and no other members of the authentication chain support SSO, password-based login will be used.

`ntlm.authentication.mapUnknownUserToGuest`

Should unknown users automatically be logged on as the Alfresco guest user during Single Sign-On (SSO)?

`ntlm.authentication.brower.ticketLogons` (new in v3.3 SP3)

Can a ticket parameter in the request URL be used to authenticate with the Alfresco Explorer application? Default is `true`. Note that WebDAV URLs always accept ticket parameters.

`alfresco.authentication.authenticateCIFS`

A Boolean that when `true` enables Alfresco-internal authentication for the CIFS server. When `false` and no other members of the authentication chain support CIFS authentication, the CIFS server will be disabled.

`alfresco.authentication.allowGuestLogin`

Should we allow guest access to Alfresco?

Alfresco Share SSO using NTLM

The Alfresco Share application exists as an entirely separate web application to the main Alfresco Repository/Explorer WAR file. It can run in the same app-server instance on the same machine to the main Alfresco web application or can run on a completely separate app-server instance on a completely different machine altogether. The Share application uses HTTP(S) to communicate with the configured Alfresco Repository. Therefore, to use NTLM with Share, you must first enable SSO for the `alfrescoNtlm` or `passthru` subsystem. Then you need to tell Share how to connect to alfresco (connector endpoints). On older versions, you also need to modify the `Share web.xml` file.

Connector endpoints

You need to make a configuration change to the Share application. The Share web application supports overriding of application config files in a very similar way to the Alfresco Explorer web client. To use NTLM with Share, find the `.sample` configuration override file for your version. That sample file name depends on the version of alfresco.

Alfresco versions 4.0 (from 4.0.2) and 3.4 (from 3.4.9)

Alfresco 4.0.2 introduces header based external authentication, see [ALF-10763](#) . A new parameter has been introduced in order to allow setting the value of the header used or external authentication. In NTLM SSO, you should not need to modify the header parameter. You will need to modify this header only if you use the 'external' authentication subsystem and you use the header based version of external authentication.

To use NTLM with Share, find the `.sample` configuration override file.

```
\\tomcat\\shared\\classes\\alfresco\\web-extension\\share-config-custom.xml.sample
```

Copy and rename the file to:

```
\\tomcat\\shared\\classes\\alfresco\\web-extension\\share-config-custom.xml
```

Then edit the file and uncomment this section as appropriate:

```
<!-- example port config used to access remote Alfresco server (default is 8080) -->
<config evaluator="string-compare" condition="Remote">
  <remote>
```

```

<endpoint>
  <id>alfresco-noauth</id>
  <name>Alfresco - unauthenticated access</name>
  <description>Access to Alfresco Repository WebScripts that do not require authentication</description>
  <connector-id>alfresco</connector-id>
  <endpoint-url>http://localhost:8080/alfresco/s</endpoint-url>
  <identity>none</identity>
</endpoint>

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

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

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

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

NOTE: For external SSO switch the endpoint connector to "AlfrescoHeader" and set
the userHeader to the name of the HTTP header that the external SSO
uses to provide the authenticated user name.
-->
<config evaluator="string-compare" condition="Remote">
  <remote>
    <keystore>
      <path>alfresco/web-extension/alfresco-system.p12</path>
      <type>pkcs12</type>
      <password>alfresco-system</password>
    </keystore>

    <connector>
      <id>alfrescoCookie</id>
      <name>Alfresco Connector</name>
      <description>Connects to an Alfresco instance using cookie-based authentication</description>
      <class>org.alfresco.web.site.servlet.SlingshotAlfrescoConnector</class>
    </connector>

    <connector>
      <id>alfrescoHeader</id>
      <name>Alfresco Connector</name>
      <description>Connects to an Alfresco instance using header and cookie-based authentication</description>
      <class>org.alfresco.web.site.servlet.SlingshotAlfrescoConnector</class>
      <userHeader>SsoUserHeader</userHeader>
    </connector>

    <endpoint>
      <id>alfresco</id>
      <name>Alfresco - user access</name>
      <description>Access to Alfresco Repository WebScripts that require user authentication</description>
      <connector-id>alfrescoCookie</connector-id>
      <endpoint-url>http://localhost:8080/alfresco/wcs</endpoint-url>
      <identity>user</identity>
      <external-auth>true</external-auth>
    </endpoint>
  </remote>
</config>

```

Note: Change the endpoint-url value to point to your Alfresco Server location.

Alfresco versions 4.0 (up to 4.0.1), 3.4 (before 3.4.9) and 3.3

The connector sample file is called:

```

\tomcat\shared\classes\alfresco\web-extension\share-config-custom.xml.sample

```

Copy and rename the file to:

```

\tomcat\shared\classes\alfresco\web-extension\share-config-custom.xml

```

Then edit the file and uncomment this section as appropriate:

```

<!--
SSO authentication config for Share
NOTE: change localhost:8080 below to appropriate alfresco server location if required
-->
<config evaluator="string-compare" condition="Remote">
  <remote>
    <connector>
      <id>alfrescoCookie</id>
      <name>Alfresco Connector</name>
      <description>Connects to an Alfresco instance using cookie-based authentication</description>
      <class>org.springframework.extensions.webscripts.connector.AlfrescoConnector</class>
    </connector>

    <endpoint>
      <id>alfresco</id>
      <name>Alfresco - user access</name>
      <description>Access to Alfresco Repository WebScripts that require user authentication</description>
      <connector-id>alfrescoCookie</connector-id>
      <endpoint-url>http://localhost:8080/alfresco/wcs</endpoint-url>
      <identity>user</identity>
      <external-auth>true</external-auth>
    </endpoint>
  </remote>
</config>

```

Note: Change the endpoint-url value to point to your Alfresco Server location.

Alfresco version 3.2

In 3.2 the connector sample file is at:

```
\tomcat\shared\classes\alfresco\web-extension\webscript-framework-config-custom.xml.sample
```

Copy and rename the file to:

```
\tomcat\shared\classes\alfresco\web-extension\webscript-framework-config-custom.xml
```

Then edit the file and uncomment this section as appropriate for your version:

```
<!-- Overriding endpoints to reference an Alfresco server with NTLM filter enabled -->
<!-- NOTE: the NTLM Authentication Filter must be enabled for both repository and web-tier web.xml -->
<!-- NOTE: if utilising a load balancer between web-tier and repository cluster, the "sticky -->
sessions" feature of your load balancer must be used when NTLM filter is active -->
<config evaluator="string-compare" condition="Remote">
  <remote>
    <connector>
      <id>alfrescoCookie</id>
      <name>Alfresco Connector</name>
      <description>Connects to an Alfresco instance using cookie-based authentication</description>
      <class>org.alfresco.connector.AlfrescoConnector</class>
    </connector>

    <endpoint>
      <id>alfresco</id>
      <name>Alfresco - user access</name>
      <description>Access to Alfresco Repository WebScripts that require user authentication</description>
      <connector-id>alfrescoCookie</connector-id>
      <endpoint-url>http://localhost:8080/alfresco/wcs</endpoint-url>
      <identity>user</identity>
      <external-auth>true</external-auth>
    </endpoint>
  </remote>
</config>
```

Note: Change the endpoint-url value to point to your Alfresco Server location.

Now restart the Share web application. If you have configured alfrescoNtlm or passthru in your Alfresco authentication chain and enabled SSO, NTLM will now be the active authentication mechanism.

Share web.xml

In older versions of alfresco, you need to modify the Share web.xml file in order to apply different authentication filters.

versions 3.4 and above

There is no need to modify the Share web.xml file.

versions 3.2 and 3.3

for 3.3, 3.2 Edit the Share application **web.xml** file in the WEB-INF folder and change the servlet filter that is used. Enable the following servlet filter:

```
<filter>
  <filter-name>Authentication Filter</filter-name>
  <filter-class>org.alfresco.web.site.servlet.NTLMAuthenticationFilter</filter-class>
  <init-param>
    <param-name>endpoint</param-name>
    <param-value>alfresco</param-value>
  </init-param>
</filter>
```

Then add the following filter mappings:

```
<filter-mapping>
  <filter-name>Authentication Filter</filter-name>
  <url-pattern>/page/*</url-pattern>
</filter-mapping>

<filter-mapping>
  <filter-name>Authentication Filter</filter-name>
  <url-pattern>/p/*</url-pattern>
</filter-mapping>

<filter-mapping>
  <filter-name>Authentication Filter</filter-name>
  <url-pattern>/s/*</url-pattern>
</filter-mapping>
```

The NTLM settings should already be in the web.xml file in a commented out section. Use the settings that are present in your web.xml as they may differ slightly from those above depending on the Alfresco version you have installed.

WebDAV (for 3.4)

The WebDAV interface is configured via the web.xml file in the WEB-INF folder. Use the following entries:

```
<context-param>
  <param-name>store</param-name>
  <param-value>workspace://SpacesStore</param-value>
</context-param>
<context-param>
  <param-name>rootPath</param-name>
  <param-value>/app:company_home</param-value>
</context-param>

<servlet>
  <servlet-name>WebDAV</servlet-name>
  <servlet-class>org.alfresco.repo.webdav.WebDAVServlet</servlet-class>
  <load-on-startup>5</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>WebDAV</servlet-name>
  <url-pattern>/webdav/*</url-pattern>
</servlet-mapping>

<filter>
  <filter-name>WebDAV Authentication Filter</filter-name>
  <filter-class>org.alfresco.repo.web.filter.beans.BeanProxyFilter</filter-class>
  <init-param>
```

```

    <param-name>beanName</param-name>
    <param-value>webDavAuthenticationFilter</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>WebDAV Authentication Filter</filter-name>
  <url-pattern>/webdav/*</url-pattern>
</filter-mapping>

```

Edit `alfresco-global.properties` to use ntlm authentication, e.g.:

```

ntlm.authentication.sso.enabled=true
ntlm.authentication.mapUnknownUserToGuest=false
alfresco.authentication.allowGuestLogin=false
ntlm.authentication.brower.ticketLogons=true
alfresco.authentication.authenticateCIFS=false

```

To set up a `webDavAuthenticationFilter` bean you need to include `alfresco-authentication-context.xml` file to your configuration (you may probably find it in `...\\alfresco\subsystems\Authentication\alfrescoNtlm` dir). And the entries for `webDavAuthenticationFilter` are:

```

<bean id="webDavAuthenticationFilter" class="org.alfresco.repo.webdav.auth.NTLMAuthenticationFilter">
  <property name="active">
    <value>${ntlm.authentication.sso.enabled}</value>
  </property>
  <property name="ticketLogons">
    <value>true</value>
  </property>
  <property name="serverConfiguration">
    <ref bean="fileServerConfiguration"/>
  </property>
  <property name="authenticationService">
    <ref bean="AuthenticationService"/>
  </property>
  <property name="authenticationComponent">
    <ref bean="AuthenticationComponent"/>
  </property>
  <property name="personService">
    <ref bean="personService"/>
  </property>
  <property name="nodeService">
    <ref bean="NodeService"/>
  </property>
  <property name="transactionService">
    <ref bean="TransactionService"/>
  </property>
  <property name="mapUnknownUserToGuest">
    <value>${ntlm.authentication.mapUnknownUserToGuest}</value>
  </property>
</bean>

```

Note (for windows users): you may use `org.alfresco.repo.webdav.auth.AuthenticationFilter` for the filter (and apparently do not use ntlm authentication), but in this case you won't be able to connect a network drive in explorer, but access to WebDAV through a browser will remain available (I'm sure you can get around that in linux systems).

Note: it is said that this configs are for 3.4 version. I haven't tried 3.2-3.3 versions, but suppose they're similar.

LDAP

An LDAP subsystem supports two main functions:

user authentication

checking a user's ID and password using an LDAP bind operation

user registry export

exposing information about users and groups to the [Synchronization Subsystem](#)

These functions can be used in isolation or in combination. When LDAP authentication is used without user registry export, default Alfresco person objects are created automatically for all those users who successfully log in. However, they will not be populated with attributes without user registry export enabled. LDAP user registry export is most likely to be used without LDAP authentication when chained with other authentication subsystems – for example, Kerberos against Active Directory, passthru against Active Directory and possibly Samba on top of OpenLDAP.

The user registry export function assumes that groups are stored in LDAP as an object that has a repeating attribute which defines the distinguished names of other groups, or users. This is supported in the standard LDAP schema using the `groupOfNames` type. See [the example LDIF file](#).

Configuration

Both the **ldap** and **ldap-ad** subsystem types support the following configurable properties. See [Configuring Subsystems](#) for how to configure these. Defaults for **ldap** are typical for Open LDAP and defaults for **ldap-ad** are typical for Active Directory.

ldap.authentication.active

This Boolean flag, when `true`, enables use of this LDAP subsystem for authentication. It may be that this subsystem should only be used for user registry export, in which case this flag should be set to `false` and you would have to chain an additional subsystem such as `passthru` or `kerberos` to provide authentication functions.

ldap.authentication.java.naming.security.authentication

The mechanism used to validate passwords with the LDAP server. Should be one of the standard values documented [here](#) or one of the values supported by the LDAP provider. Sun's LDAP provider supports the SASL mechanisms documented [here](#). Recommended values are:

simple

the basic LDAP authentication mechanism requiring the username and password to be passed over the wire unencrypted. You may be able to add SSL for secure access; otherwise, only use this for testing.

DIGEST-MD5

More secure [RFC 2831](#) Digest Authentication. Note that with Active Directory, this requires your user accounts to be set up with reversible encryption, not the default setting.

ldap.authentication.userNameFormat

How to map the user id entered by the user to that passed through to LDAP. If set to the empty string (the default for the `ldap` subsystem), an LDAP query involving `ldap.synchronization.personQuery` and `ldap.synchronization.userIdAttributeName` will be performed to resolve the DN from the user ID dynamically. This allows directories to be structured and doesn't require the user ID to appear in the DN. If set to a non-empty value, the substring `%s` in this value will be replaced with the entered user ID to produce the ID passed to LDAP. This restricts LDAP user names to a fixed format. The recommended format of this value depends on your LDAP server.

Active Directory

You have two alternatives:

User Principal Name (UPN)

These are generally in the format of <sAMAccountName>@<UPN Suffix>. Official documentation can be found [here](#). If you are unsure of the correct suffix to use, use an LDAP browser, such as Softerra, to browse to a user account and find its `userPrincipalName` attribute. For example:

```
%s@domain
```

DN

This requires the user to authenticate with part of their DN, so may require use of their common name (CN) rather than their login ID. It also may not work with structured directory layouts containing multiple organization units (OUs). For example:

```
cn=%s,ou=xyz,dc=domain
```

OpenLDAP

The format used depends on the value chosen for `ldap.authentication.java.naming.security.authentication`.

simple

This must be a DN and would be something like

```
uid=%s,ou=People,dc=company,dc=com
```

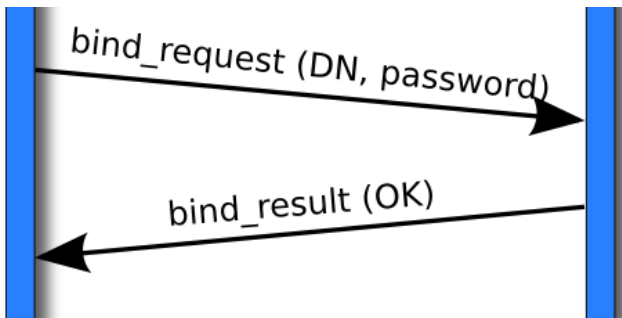
DIGEST-MD5

Use this value to pass through the entered value as-is:

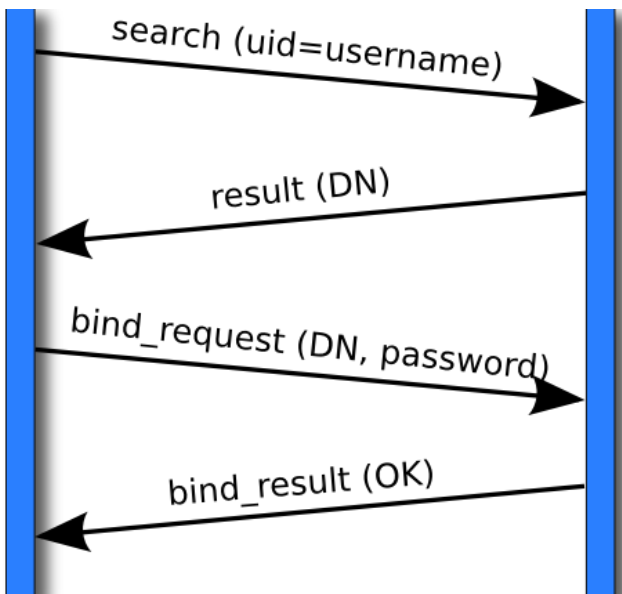
```
%s
```

BLANK

Note that the `ldap.authentication.userNameFormat` can also be blank (see [\[ALF-5155\]](#) and [\[ETHEREOH-1806\]](#)); In that case a search will first be made to identify a unique DN, then a bind



The above picture illustrates the LDAP authentication using the 'simple' mechanism and a non blank `userNameFormat`.



The above picture illustrates the LDAP authentication using the 'simple' mechanism and a *blank* `userNameFormat`.

`ldap.authentication.allowGuestLogin`

Should we allow unauthenticated users to log in to Alfresco as 'guest'?

`ldap.authentication.java.naming.factory.initial`

The LDAP context factory to use. There is no need to change this unless you do not want to use the default Sun LDAP context factory.

`ldap.authentication.java.naming.provider.url`

The URL to connect to the LDAP server, containing its name and port. The standard ports for LDAP are 389 (and 636 for SSL). For example:

```
ldap://openldap.domain.com:389
```

`ldap.authentication.escapeCommasInBind`

Escape commas in the entered user ID when authenticating with the LDAP server? Useful when using simple authentication and the CN is part of the DN and contains commas.

`ldap.authentication.escapeCommasInUid`

Escape commas in the entered user ID when deriving an Alfresco internal user ID? Useful when using simple authentication and the CN is part of the DN and contains commas, and the escaped `\,` is pulled in as part of `asynchronize` operation. If this option is set to true it will break the default home folder provider as space names cannot contain `\`.

`ldap.authentication.defaultAdministratorUserNames`

A comma separated list of user names who should be considered administrators by default

`ldap.synchronization.active`

This flag enables use of the LDAP subsystem for user registry export functions and decides whether the subsystem will contribute data to the [Synchronization Subsystem](#). It may be that this subsystem should only be used for authentication, in which case this flag should be set to false.

ldap.synchronization.java.naming.security.authentication

The authentication mechanism used to connect to the LDAP server when performing user registry exports. Note this property is new in v3.4 and in previous versions it was the same `asldap.authentication.java.naming.security.authentication`. Should be one of the standard values documented [here](#) or one of the values supported by the LDAP provider. Sun's LDAP provider supports the SASL mechanisms documented [here](#). Recommended values are:

none

your LDAP server may support connection without a password. This is known as **anonymous bind**.

simple

the basic LDAP authentication mechanism requiring the username and password to be passed over the wire unencrypted. You may be able to add SSL for secure access; otherwise, only use this for testing.

DIGEST-MD5

More secure [RFC 2831](#) Digest Authentication. Note that with Active Directory, this requires your user accounts to be set up with reversible encryption, not the default setting.

ldap.synchronization.java.naming.security.principal

The LDAP user to connect as to do the export operation, if one is required by the above authentication mechanism.

ldap.synchronization.java.naming.security.credentials

The password for this user, if required.

ldap.synchronization.queryBatchSize

If set to a positive integer, this property indicates that [RFC 2696](#) paged results should be used to split query results into batches of the specified size. This overcomes any size limits imposed by the LDAP server. The default value of 1000 matches the default result limitation imposed by Active Directory. If set to zero or less, paged results will not be used. **Note:** when using openldap, `ldap.synchronization.queryBatchSize` needs to be zero, as many versions of openldap do not support this type of paging.

See <http://support.microsoft.com/kb/555536> In practice, using a non zero `ldap.synchronization.queryBatchSize` with openldap could result in empty groups.

ldap.synchronization.attributeBatchSize

If set to a positive integer, this property indicates that [range retrieval](#) should be used to fetch multi-valued attributes (such as `member`) in batches of the specified size. Overcomes any size limits imposed by Active Directory.

Note: if you set this parameter to a non zero value for non MS Active Directory LDAP directories, this will result in **empty groups!**

ldap.synchronization.groupQuery

The query to select all objects that represent the groups to export. This query is used in 'full sync mode', which by default is scheduled every 24 hours. See [The Synchronization Subsystem](#).

ldap.synchronization.groupDifferentialQuery

The query to select objects that represent the groups to export that have changed since a certain time. Should use the placeholder `{0}` in place of a timestamp in the format specified by `ldap.synchronization.timestampFormat`. The timestamp substituted will be the maximum value of the attribute named by `ldap.synchronization.modifyTimestampAttributeName` the last time groups were queried. This query is used in 'differential sync mode', which by default is triggered whenever a user is successfully authenticated that does not yet exist in Alfresco. See [The Synchronization Subsystem](#).

ldap.synchronization.personQuery

The query to select all objects that represent the users to export. This query is used in 'full sync mode', which by default is scheduled every 24 hours. See [The Synchronization Subsystem](#).

ldap.synchronization.personDifferentialQuery

The query to select objects that represent the users to export that have changed since a certain time. Should use the placeholder `{0}` in place of a timestamp in the format specified by `ldap.synchronization.timestampFormat`. The timestamp substituted will be the maximum value of the attribute named by `ldap.synchronization.modifyTimestampAttributeName` the last time users were queried. This query is used in 'differential sync mode', which by default is triggered whenever a user is successfully authenticated that does not yet exist in Alfresco. See [The Synchronization Subsystem](#).

ldap.synchronization.groupSearchBase

The DN below which to run the group queries.

ldap.synchronization.userSearchBase

The DN below which to run the user queries.

ldap.synchronization.modifyTimestampAttributeName

The name of the operational attribute recording the last update time for a group or user.

ldap.synchronization.timestampFormat

The timestamp format. Unfortunately, this varies between directory servers.

Active Directory

yyyyMMddHHmmss'.0Z'

OpenLDAP

yyyyMMddHHmmss'Z'

ldap.synchronization.userIdAttributeName

The attribute name on people objects found in LDAP to use as the uid in Alfresco.

ldap.synchronization.userFirstNameAttributeName

The attribute on person objects in LDAP to map to the first name property in Alfresco.

ldap.synchronization.userLastNameAttributeName

The attribute on person objects in LDAP to map to the last name property in Alfresco.

ldap.synchronization.userEmailAttributeName

The attribute on person objects in LDAP to map to the email property in Alfresco.

ldap.synchronization.userOrganizationalIdAttributeName

The attribute on person objects in LDAP to map to the email property in Alfresco.

ldap.synchronization.defaultHomeFolderProvider

The default home folder provider to use for people created via LDAP import. See [Security_and_Authentication#Creating_home_spaces_-_from_1.4_onwards](#).

ldap.synchronization.groupIdAttributeName

The attribute on LDAP group objects to map to the authority name in Alfresco.

ldap.synchronization.groupDisplayNameAttributeName

The attribute on LDAP group objects to map to the authority display name property in Alfresco (v3.3+)

ldap.synchronization.groupType

The group type in LDAP.

ldap.synchronization.personType

The person type in LDAP

ldap.synchronization.groupMemberAttributeName

The name of the multi-valued attribute on an LDAP group object that lists its members. If the value of this attribute parses as a Distinguished Name (DN) then the exporter will resolve the member name and type by looking up that DN, determining its object class (user or group) and getting the appropriate name attribute. This strategy will work with the `groupOfNames` class, for example. Otherwise, the attribute value is assumed to contain a user ID. This strategy will work with the `posixGroup` class, for example.

How do I find out what Simple Authentication and Security Layer (SASL) authentication mechanisms my LDAP server supports?

Using an LDAP browser, such as the one from Softerra, check the values of the supportedSASLMechanisms attributes on the root node of your LDAP server. Note: the "simple" authentication method will not be reported as it is not a SASL mechanism.

If you use OpenLDAP, you can also query using ldapsearch:

```
ldapsearch -h localhost -p 389 -x -b "" -s base -LLL supportedSASLMechanisms
dn:
supportedSASLMechanisms: DIGEST-MD5
supportedSASLMechanisms: NTLM
supportedSASLMechanisms: CRAM-MD5
```

OpenLDAP Tips

Sample Configuration

Here is a sample configuration file.

There are a number of things to note:

- The maximum number of results returned has been increased from the default of 500 that even applies to paged results. See the OpenLDAP documentation on [limits](#). If you have more than 500 users or groups this would be an issue.
- Digest authentication has been configured to map from a user ID to the corresponding distinguished name. See the example data.
- Passwords are in clear text (so that any authentication mechanism can be used). It is possible they can be in the correct hashed form for the MD5 digest to work.

```
#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
include      /usr/local/etc/openldap/schema/core.schema
include      /usr/local/etc/openldap/schema/cosine.schema
include      /usr/local/etc/openldap/schema/inetorgperson.schema

# Define global ACLs to disable default read access.

# Do not enable referrals until AFTER you have a working directory
# service AND an understanding of referrals.
#referral    ldap://root.openldap.org

pidfile      /usr/local/var/run/slapd.pid
argsfile     /usr/local/var/run/slapd.args

# Load dynamic backend modules:
# modulepath /usr/local/libexec/openldap
# moduleload back_bdb.la
# moduleload back_ldap.la
# moduleload back_ldbm.la
# moduleload back_passwd.la
# moduleload back_shell.la

# Sample security restrictions
# Require integrity protection (prevent hijacking)
# Require 112-bit (3DES or better) encryption for updates
# Require 63-bit encryption for simple bind
# security ssf=1 update_ssf=112 simple_bind=64

# Sample access control policy:
# Root DSE: allow anyone to read it
# Subschema (sub)entry DSE: allow anyone to read it
# Other DSEs:
#     Allow self write access
#     Allow authenticated users read access
#     Allow anonymous users to authenticate
# Directives needed to implement policy:
# access to dn.base="" by * read
# access to dn.base="cn=Subschema" by * read
# access to *
#     by self write
#     by users read
#     by anonymous auth
#
# if no access controls are present, the default policy
# allows anyone and everyone to read anything but restricts
# updates to rootdn. (e.g., "access to * by * read")
#
# rootdn can always read and write EVERYTHING!

#####
# BDB database definitions
#####

database     bdb
suffix       "dc=company,dc=com"
rootdn       "cn=Manager,dc=company,dc=com"
# Cleartext passwords, especially for the rootdn, should
# be avoided. See slapd.conf(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
# This is secret ...
rootpw       {SSHA}u9AUUyOSVX6id1XcwYyOAG6G84oHfPvG
# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
directory    /usr/local/var/openldap-data
# Indices to maintain
```

```

index    objectClass    eq

# Clear text to allow hashing
password-hash    {CLEARTEXT}

# SASL mappings for md5 digest authentication
# Extract the user id and use as the search key

authz-regexp
uid=([^\,]*) ,cn=digest-md5,cn=auth
ldap:///dc=company,dc=com?one?(uid=$1)

authz-regexp
uid=([^\,]*) ,cn=company.com,cn=digest-md5,cn=auth
ldap:///dc=company,dc=com?one?(uid=$1)

# Tweaks to increase the result set size and max query time

sizelimit 50000
timelimit 3600

```

Sample Data

Here is a very simple example LDIF file that defines `People` and `Groups` Organizational units and some example users and groups.

```

# Initial directory contents
dn: dc=company,dc=com
dc: company
objectClass: top
objectClass: domain

dn: ou=People,dc=company,dc=com
ou: People
objectClass: top
objectClass: organizationalUnit

dn: ou=Groups,dc=company,dc=com
ou: Groups
objectClass: top
objectClass: organizationalUnit

dn: uid=fullname,ou=People,dc=company,dc=com
objectClass: inetOrgPerson
sn: Name
cn: Full Name
userPassword: inClearText
telephoneNumber: 1234567890
uid: fullname
givenName: Full
mail: full.name@company.com
o: Company Software Inc.

dn: uid=walrus,ou=People,dc=company,dc=com
objectClass: inetOrgPerson
sn: Rus
cn: Wal Rus
userPassword: inClearText
telephoneNumber: 1234567890
uid: walrus
givenName: Wal
mail: wal.rus@company.com
o: Company Software Inc.

dn: cn=Group One,ou=Groups,dc=company,dc=com
objectClass: groupOfNames
cn: Group One
member: uid=fullname,ou=People,dc=company,dc=com

dn: cn=Group Two,ou=Groups,dc=company,dc=com
objectClass: groupOfNames
cn: Group Two
member: cn=Group One,ou=Groups,dc=company,dc=com
member: uid=walrus,ou=People,dc=company,dc=com

```

Active Directory Tips

- You may need to give special permissions in the Active Directory to the account that you are using to do the LDAP bind (as configured in `ldap.synchronization.java.naming.security.principal`). To do this, open Active Directory Users and Computers, right click on the domain, and select "Delegate Control..." Click "Next", then select the user that you are using for the LDAP bind and click "Next". The permission that they will need is on the next screen "Read all inetOrgPerson information."
- The example URL in `ldap.authentication.java.naming.provider.url` does not use SSL. SSL is recommended for production systems. You'll need to switch the port from 389 (below, non-SSL) to 636 for SSL.
- It is often helpful to screen out non-user accounts and disabled accounts. The default user queries in the **Idap-ad** subsystem type do this by checking bit fields on the `userAccountControl` attribute. For example:

```
userAccountControl:1.2.840.113556.1.4.803:=512
```

Passthru

Introduction

The passthru subsystem can be used to replace the standard Alfresco user database with a Windows server/domain controller, or list of servers, to authenticate users accessing Alfresco. This saves having to create user accounts within Alfresco. The subsystem also supports optional NTLM Single Sign-On (SSO) functions for WebDav and the Alfresco Explorer and Share clients and direct CIFS authentication for the CIFS server. This method of authentication is much more secure than simple LDAP-based authentication or form-based authentication.

Note that only NTLM v1 is supported in this configuration. As NTLMv2 has been designed to avoid "man-in-the-middle" attacks, it would be impossible to use in this 'pass through' style.

Configuration

The passthru subsystem supports the following properties. See [Configuring Subsystems](#) for how to configure these. Also relevant are the configuration steps described in [Alfresco Share SSO using NTLM](#) if you want to enable NTLM-based Single Sign-On (SSO) for the Alfresco Share client.

Domain level properties

The following properties control the set of domain controllers used for authentication. The three properties are mutually exclusive. For example, to set the `passthru.authentication.servers` property, set `passthru.authentication.domain` to be empty and `passthru.authentication.useLocalServer` to be `false`.

passthru.authentication.useLocalServer

A Boolean that when `true` indicates that the local server should be used for passthru authentication by using loopback connections into the server.

passthru.authentication.domain

Sets the domain to use for passthru authentication. This will attempt to find the domain controllers using a network broadcast. Make sure that you use the Windows NetBIOS domain name, not the forest name. The network broadcast does not work in all network configurations. In this case use the `passthru.authentication.servers` property to specify the domain controller list by name or address.

passthru.authentication.servers

A comma delimited list of server names or addresses that are used for authentication. The passthru authenticator will load balance amongst the available servers, and can monitor server online/offline status.

- Each server name/address may be prefixed with a domain name using the format `<domain>\<server>`. If specifying this in `alfresco-global.properties`, remember that the backslash character must be escaped. For example
`passthru.authentication.servers=DOMAIN1\\host1.com,DOMAIN2\\host2.com,host1.com`
- If the client specifies a domain name in its logon request then the appropriate server will be used for the authentication. Domain mappings may also be specified to route authentication requests to the appropriate server (see below).
- If a server handles authentication for multiple domains then multiple entries can be added in the server list prefixed with each domain name.
- There should be at least one entry in the server list that does not have a domain prefix, this is the *catch all* entry that will be used if the client domain cannot be determined from the NTLM request or via domain mapping.

Other properties

ntlm.authentication.sso.enabled

A Boolean that when `true` enables NTLM based Single Sign On (SSO) functionality in the Web clients. When `false` and no other members of the authentication chain support SSO, password-based login will be used.

ntlm.authentication.mapUnknownUserToGuest

Should unknown users automatically be logged on as the Alfresco guest user during Single Sign-On (SSO)?

ntlm.authentication.browser.ticketLogons (new in v3.3 SP3)

Can a ticket parameter in the request URL be used to authenticate with the Alfresco Explorer application? Default is `true`. Note that WebDAV URLs always accept ticket parameters.

passthru.authentication.authenticateCIFS

A Boolean that when `true` enables pass-through authentication for the CIFS server. When `false` and no other members of the authentication chain support CIFS authentication, the CIFS server will be disabled.

passthru.authentication.authenticateFTP

A Boolean that when `true` enables pass-through authentication for the FTP server. The provided password is hashed and checked directly against the domain server securely using NTLM. When `false` and no other members of the authentication chain support FTP authentication, standard chained authentication will be used.

passthru.authentication.guestAccess

Should we allow guest access to Alfresco if the authenticating server indicates the logon was allowed guest access?

passthru.authentication.defaultAdministratorUserNames

A comma separated list of user names who should be considered administrators by default. It's often useful to add the `administrator` user to this list.

passthru.authentication.connectTimeout

The timeout value when opening a session to an authentication server, in milliseconds. The default is 5000.

passthru.authentication.offlineCheckInterval

Specifies how often passthru servers that are marked as offline are checked to see if they are now online. The default check interval is 5 minutes. The check interval is specified in seconds.

passthru.authentication.protocolOrder

Specifies the type of protocols and the order of connection for passthru authentication sessions. The default is to use NetBIOS, if that fails then try to connect using native SMB/port 445. Specify either a single protocol type or a comma delimited list with a primary and secondary protocol type. The available protocol types are `NetBIOS` for NetBIOS over TCP and `TCP/IP` for native SMB.

Domain Mappings

Domain mappings are used to determine the domain a client is a member of when the client does not specify its domain in the logon request. If the client uses a numeric IP address to access the web server it will not send the domain in the NTLM request as the browser assumes it is an Internet address.

To specify the domain mapping rules that are used when the client does not supply its domain in the NTLM request you can use the `filesystem.domainMappings` composite property of the [File Server Subsystem](#). There are two ways of defining a domain mapping, either by specifying an IP subnet and mask, or by specifying a range of IP addresses. The example below, when included in `alfresco-global.properties` defines mappings for two domains, `ALFRESCO` and `OTHERDOM`. For more information on setting composite properties see [Setting composite properties](#).

```
filesystem.domainMappings=ALFRESCO,OTHERDOM
filesystem.domainMappings.value.ALFRESCO.subnet=192.168.1.0
filesystem.domainMappings.value.ALFRESCO.mask=192.168.1.255
filesystem.domainMappings.value.OTHERDOM.rangeFrom=192.168.1.0
filesystem.domainMappings.value.OTHERDOM.rangeTo=192.168.1.100
```

The passthru subsystem can use the domain prefixed server name format of the `passthru.authentication.servers` property along with the domain mappings to route authentication requests to the appropriate server.

A sample NTLM authentication component server list:

```
passthru.authentication.servers=ALFRESCO\ADSERVER,OTHERDOM\OTHERSRV
```

NTLM Client Configuration

By using NTLM authentication to access Alfresco and Alfresco WebDAV sites the web browser can automatically logon.

This assumes you have `ntlm.authentication.sso.enabled` set to true in your passthru subsystem, configuration.

Internet Explorer will use your Windows logon credentials when requested by the web server when NTLM is configured. If IE sends the login popup window this probably means you will need to add the URL of your alfresco server to your IE Intranet sites in

Tools->Options->Security->Local Intranet ->Sites->advanced In the Intranet security->custom, at the bottom of the list did you tick User Authentication->Logon->Automatic login with current user name and password? (the 2nd possibility ""automatic logon only in Intranet zone" may also work depending on your settings)

Firefox and **Mozilla** also support the use of NTLM but you need to add the URI to the Alfresco site that you want to access to *network.automatic-ntlm-auth.trusted-uris* option (available through writing *about:config* in the URL field) to allow the browser to use your current credentials for login purposes.

The **Opera** web browser does not support NTLM authentication, the browser is detected and will be sent to the usual Alfresco logon page.

Kerberos

Introduction

The Java Authentication and Authorization Service (JAAS) is used within the Kerberos subsystem to support Kerberos authentication of user names and passwords. You may choose to use Kerberos against an Active Directory server in preference to LDAP or NTLM as it provides strong encryption without using SSL. It would still be possible to export user registry information via a chained LDAP subsystem.

The disadvantages of using LDAP authentication against Active Directory compared with JAAS/Kerberos are:

- the simplest approach is to use the SIMPLE LDAP authentication protocol, which should be used with SSL;
- AD requires special set up to use digest MD5 authentication (reversible encryption for passwords), which may be difficult retrospectively;
- LDAP can use GSSAPI + Kerberos which would be equivalent but this is more difficult to configure and has not been tested.

For some pointers and background information on JAAS, the Java Authentication and Authorization Service, please see:

- [JAAS home page](#)
- [JAAS article on Wikipedia](#)

Alfresco Configuration

To enable full Kerberos support in Alfresco requires that the CIFS server and the SSO authentication filters each have a Kerberos service ticket. See [Configuring against Active Directory](#) for how this is set up on Active Directory.

The Kerberos subsystem supports the following properties. See [Configuring Subsystems](#) for how to configure these.

kerberos.authentication.realm

The Kerberos realm to authenticate with. The realm should be the domain upper cased; example if the domain is *alfresco.org* then the realm should be *ALFRESCO.ORG*

kerberos.authentication.sso.enabled

A Boolean that when *true* enables SPNEGO/Kerberos based Single Sign On (SSO) functionality in the Web client. When *false* and no other members of the authentication chain support SSO, password-based login will be used.

kerberos.authentication.authenticateCIFS

A Boolean that when *true* enables Kerberos authentication in the CIFS server. When *false* and no other members of the authentication chain support CIFS authentication, the CIFS server will be disabled.

kerberos.authentication.user.configEntryName

The name of the entry in the JAAS configuration file that should be used for password-based authentication. The default value *Alfresco* is a good choice here.

kerberos.authentication.cifs.configEntryName

The name of the entry in the JAAS configuration file that should be used for CIFS authentication. The default value *AlfrescoCIFS* is a good choice here.

kerberos.authentication.http.configEntryName

The name of the entry in the JAAS configuration file that should be used for web-based single-sign on (SSO). The default value *AlfrescoHTTP* is a good choice here.

kerberos.authentication.defaultAdministratorUserNames

A comma separated list of user names who should be considered administrators by default

kerberos.authentication.browser.ticketLogons (new in v3.3 SP3)

Can a ticket parameter in the request URL be used to authenticate with the Alfresco Explorer application? Default is *true*. Note that WebDAV URLs always accept ticket parameters.

New feature (see [ALF-13121](#)) Option to create users either as *user1* or *user1@domain.com* after kerberos authentication)

kerberos.authentication.stripUsernameSuffix=false

CIFS, SPP, WebDAV and the Web Client could be successfully authenticated with kerberos sso. The user is registered as *Administrator@QALAB.ALFRESCO*

kerberos.authentication.stripUsernameSuffix=true

CIFS, SPP, WebDAV and the Web Client could be successfully authenticated with kerberos sso. The user is registered as *Administrator*. This is the *default* behaviour.

Of course, Kerberos needs to be specified in the authentication chain.

Also, in order for Kerberos to work with usernames containing non-ASCII characters, the option below should be added to *JAVA_OPTS* for the Alfresco (Share) JVM:

```
-Dsun.security.krb5.msinterop.kstring=true
```

Making sure Kerberos is set up

Here it is assumed you have a simple configuration and are setting up a machine just to connect to existing Kerberos support, such as Active Directory.

MIT Docs

Kerberos: The Network Authentication Protocol <http://web.mit.edu/kerberos>

Microsoft docs

Kerberos Authentication in Windows Server 2003<http://technet2.microsoft.com/windowsserver/en/technologies/featured/kerberos/default.aspx>

Kerberos Authentication in Windows 2000 Server<http://www.microsoft.com/technet/prodtechnol/windows2000serv/technologies/security/kerberos/default.aspx>

Troubleshooting Kerberos <http://technet2.microsoft.com/WindowsServer/en/library/26ce2e7f-52d6-4425-88cc-1573bc5e646d1033.mspx>

Setting the kerberos configuration can be done from the java command line, using system variables, or using a kerberos config file. This section describes using a config file.

The default locations for the config file are:

- /etc/krb5/krb5.conf [Solaris]
- c:\winnt\krb5.ini [Windows] or
- c:\WINDOWS\krb5.ini [Windows]
- /etc/krb5.conf [Linux]

Configuring against Active Directory

The following instructions detail how to set up accounts under Active Directory for use by Alfresco **running on Java 6**.

Create a user account for the Alfresco CIFS

Create a user account for the Alfresco CIFS server using the Active Directory Users and Computers application. Use the Action->New->User menu, then enter the full name as *Alfresco CIFS* and the user logon name as *alfrescocifs*. Click Next, enter a password, enable 'Password never expires' and disable 'User must change password at next logon'. Click Finish.

Right click the new user account name, select Properties, go to the Account tab and enable the *Do not require Kerberos preauthentication* option in the Account Options section. This sets the **DONT_REQ_PREAUTH** flag in your directory for that user.

Note: This is how you can test from a remote computer that the user has been created and that the flag DONT_REQ_PREAUTH has been set:

```
ldapsearch -h 10.69.69.98 -x -D "CN=Administrator,CN=Users,DC=example,DC=foo" -W -b 'DC=example,DC=foo' '(sAMAccountName=alfrescocifs) sAMAccountName userAccountControl
```

```
# Alfresco CIFS, Users, example.foo
dn: CN=Alfresco CIFS,CN=Users,DC=example,DC=foo
userAccountControl: 4260352
sAMAccountName: alfrescocifs
```

To check that the flag DONT_REQ_PREAUTH is set, we need to check that the value retrieved above for userAccountControl contains (bitwise) the value 4194304

```
python -c "print(4260352 & 4194304==4194304)"
True
```

Create a user account for the Alfresco HTTP

Create a user account for the Alfresco SSO authentication filters as in step 1 using the full name 'Alfresco HTTP' and user logon name as 'alfrescohttp'.

Create the Service Principal Name for Alfresco CIFS

Create the Service Principal Names (SPN) for the Alfresco CIFS and Alfresco HTTP server using the *setspn* utility. The *setspn* utility is a free download from the Microsoft site, and is also part of the [Windows Server 2003 Support Tools download](#).

```
setspn -a cifs/<cifs-server-name> alfrescocifs
setspn -a cifs/<cifs-server-name>.<domain> alfrescocifs
```

Some versions of the *ktutil* command will add the SPN for the principal so you may only need to add the NetBIOS/short name versions of the SPNs. Use the *setspn -l <account-name>* command to check if the *ktutil* command set the SPN.

You can list the SPNs for a server using :-

```
setspn -l <account-name>
```

That is:

```
setspn -l alfrescocifs
```

At this stage you should see two more lines running:

```
ldapsearch -h 10.69.69.98 -x -D "CN=Administrator,CN=Users,DC=example,DC=foo" -W -b 'DC=verylongexample,DC=foo' '(sAMAccountName=alfrescocifs)'
# Alfresco CIFS, Users, example.foo
dn: CN=Alfresco CIFS,CN=Users,DC=example,DC=foo
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: Alfresco CIFS
distinguishedName: CN=Alfresco CIFS,CN=Users,DC=example,DC=foo
displayName: Alfresco CIFS
name: Alfresco CIFS
sAMAccountName: alfrescocifs
userPrincipalName: alfrescocifs@example.foo
servicePrincipalName: cifs/madona.example.foo
servicePrincipalName: cifs/madona
```

Create the Service Principal Name for Alfresco HTTP

Repeat the same commands as above, but for the HTTP service:

```
setspn -a HTTP/<web-server-name> alfrescohttp
setspn -a HTTP/<web-server-name>.<domain> alfrescohttp
```

Result can be tested on the Active directory server using:

```
setspn -l alfrescohttp
```

or remotely (e.g from the alfresco server) using:

```
# Alfresco HTTP, Users, example.foo
dn: CN=Alfresco HTTP,CN=Users,DC=example,DC=foo
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: Alfresco HTTP
distinguishedName: CN=Alfresco HTTP,CN=Users,DC=example,DC=foo
displayName: Alfresco HTTP
name: Alfresco HTTP
sAMAccountName: alfrescohttp
userPrincipalName: alfrescohttp@example.foo
servicePrincipalName: HTTP/madona.example.foo
servicePrincipalName: HTTP/madona
```

optionally set the TRUSTED_FOR_DELEGATION flag on the Alfresco HTTP user

If you plan to use Share with kerberos, then you also need to modify the user account in Active Directory by clicking the radio button *Trust this user for delegation to any service*

(kerberos only). This sets the [TRUSTED_FOR_DELEGATION](#) flag in your directory for that user. If you do not see the delegation tab, see the [ShareActive Directory Configuration](#) section below for how to make it appear and for more information on delegation.

Note: The [TRUSTED_FOR_DELEGATION](#) value is 524288 so to check remotely the 'alfrescohttp' user has that flag set, you just need to do a ldap request for this user and check that the userAccountControl contains (bitwise AND) 524288:

```
ldapsearch -h 10.69.69.98 -x -D "CN=Administrator,CN=Users,DC=example,DC=foo" -W -b 'DC=example,DC=foo' '(sAMAccountName=alfrescohttp)' sAMAccountName userAccountControl

# Alfresco HTTP, Users, example.foo
dn: CN=Alfresco HTTP,CN=Users,DC=example,DC=foo
userAccountControl: 4784640
sAMAccountName: alfrescohttp
```

Do the bitwise AND check:

```
python -c "print(4784640 & 524288 == 524288)"
True
```

Create the keytabs to get passwordless login for the alfresco server

Use the *ktpass* utility to generate key tables for the Alfresco CIFS and web server. The *ktpass* utility is a free download from the Microsoft site, and is also part of the Windows Server 2003 Support Tools download. The *ktpass* command can only be run from the Active Directory server. **Note:** In Windows 2008 and 2008r2 *ktpass* must be run from an elevated command prompt.

Note that -kvno 0 arguments are now required as of JDK 1.6 u22.

```
ktpass -princ cifs/<cifs-server-name>.<domain>@<realm> -pass <password> -mapuser <domainnetbios>\alfrescocifs
-crypto RC4-HMAC-NT -ptype KRB5_NT_PRINCIPAL -out c:\temp\alfrescocifs.keytab -kvno 0

ktpass -princ HTTP/<web-server-name>.<domain>@<realm> -pass <password> -mapuser <domainnetbios>\alfrescohttp
-crypto RC4-HMAC-NT -ptype KRB5_NT_PRINCIPAL -out c:\temp\alfrescohttp.keytab -kvno 0
```

The principal should be specified using the server name and domain in lowercase with the realm in uppercase. The service types should match *cifs* and *HTTP*.

E.g. *cifs/server.alfresco.org@ALFRESCO.ORG*.

The realm should be the domain upper cased; example if the domain is *alfresco.org* then the realm should be *ALFRESCO.ORG*

<web-server-name> is the host name that is running the Alfresco server.

<cifs-server-name> is the NetBIOS name of the Alfresco CIFS server when running on an Active Directory client or the host name for a client that is not an Active Directory client, ie. not logged onto the domain.

<domain> is the DNS domain, example *alfresco.org*

<domainnetbios> is the netbios name, example *alfresco*

Note: Some old versions (Windows 2003 or earlier?) of the ktpass command can generate invalid keytab files, download the latest version of the support tools from the Microsoft site to avoid any problems.

Note: RC4-HMAC-NT seems to be suitable only for Windows XP clients, for Windows 7 clients use AES256-SHA1 or even better, generate a tabfile with both keys.

See: <http://sourceforge.net/p/spnego/discussion/1003768/thread/960ba7ad/>

Transfer the keytab files

Copy the key table files created in the above step to the server where Alfresco will run. Copy the files to a protected area such as C:\etc\ or /etc.

Normally, one must enter a password to authenticate using Kerberos. The problem with this is when scripts or programs need to authenticate without human interaction: When alfresco boots with kerberos, the server needs to authenticate and you don't want to enter the kerberos password each time alfresco boots. The keytab allows you to authenticate to kerberos without having to type the password. A keytab is basically a file containing one or more Kerberos account name (you will see these referred to as 'principals') and an encrypted value derived from the password. Using a keytab file, you can authenticate to kerberos without human interaction. As the keytab allow to login, the permission on the keytab files should be well thought: you typically want to restrict access to those keytab files to the 'tomcat' user (the user running the webapp server).

Set default options on the server

Set up the Kerberos ini file, the default location is C:\WINNT\krb5.ini or /etc/krb5.conf.

```
[libdefaults]
default_realm = ALFRESCO.ORG
default_tkt_enctypes = rc4-hmac
default_tgs_enctypes = rc4-hmac

[realms]
ALFRESCO.ORG = {
  kdc = adsrv.alfresco.org
  admin_server = adsrv.alfresco.org
}

[domain_realm]
adsrv.alfresco.org = ALFRESCO.ORG
.adsrv.alfresco.org = ALFRESCO.ORG
```

Note: The realm should be specified in uppercase.

Test the keytab/krb5.conf with kinit

The 'kinit' command can be used to simulate the kerberos queries made at alfresco boot time, e.g:

```
kinit -k -t /etc/keys/alfrescocifs.keytab "cifs/madona.example.foo"
```

Note: to list the SPN in a keytab you can use:

```
root@madona:/etc/keys# klist -k alfrescohttp.keytab
Keytab name: FILE:alfrescohttp.keytab
KVNO Principal
-----
0 HTTP/madona.example.foo@EXAMPLE.FOO
```

Configure Java security

Set up the Java login configuration file.

For JBoss 5, this is done in \$JBoss_HOME/server/default/conf/login-config.xml. Add the following entries inside the <policy> tag.

```
<application-policy name="Alfresco">
  <authentication>
    <login-module code="com.sun.security.auth.module.Krb5LoginModule" flag="sufficient"/>
  </authentication>
</application-policy>

<application-policy name="AlfrescoCIFS">
  <authentication>
    <login-module code="com.sun.security.auth.module.Krb5LoginModule" flag="required">
      <module-option name="debug">true</module-option>
      <module-option name="storeKey">true</module-option>
      <module-option name="useKeyTab">true</module-option>
      <module-option name="isInitiator">false</module-option>
      <module-option name="keyTab">/etc/alfrescocifs.keytab</module-option>
      <module-option name="principal">cifs/<cifs-server-name>.domain</module-option>
    </login-module>
  </authentication>
</application-policy>

<application-policy name="AlfrescoHTTP">
  <authentication>
    <login-module code="com.sun.security.auth.module.Krb5LoginModule" flag="required">
      <module-option name="debug">true</module-option>
      <module-option name="storeKey">true</module-option>
      <module-option name="isInitiator">false</module-option>
      <module-option name="useKeyTab">true</module-option>
      <module-option name="keyTab">/etc/alfrescohttp.keytab</module-option>
      <module-option name="principal">HTTP/<web-server-name>.<domain></module-option>
    </login-module>
  </authentication>
</application-policy>
```

For other environments, this would usually be configured in the JRE\lib\security folder (e.g: /usr/local/jdk1.6.0_03/jre/lib/security).

Create a file named *java.login.config* with the following entries :-

```
Alfresco {
    com.sun.security.auth.module.Krb5LoginModule sufficient;
};

AlfrescoCIFS {
    com.sun.security.auth.module.Krb5LoginModule required
    storeKey=true
    useKeyTab=true
    keyTab="/etc/alfrescocifs.keytab"
    principal="cifs/<cifs-server-name>.<domain>";
};

AlfrescoHTTP {
    com.sun.security.auth.module.Krb5LoginModule required
    storeKey=true
    useKeyTab=true
    keyTab="/etc/alfrescohttp.keytab"
    principal="HTTP/<web-server-name>.<domain>";
};

com.sun.net.ssl.client {
    com.sun.security.auth.module.Krb5LoginModule sufficient;
};

other {
    com.sun.security.auth.module.Krb5LoginModule sufficient;
};
```

Enable the login config file in the main Java security configuration file, usually at JRE\lib\security\java.security. Add the following line :-

```
login.config.url.1=file:${java.home}/lib/security/java.login.config
```

Kerberos Client Configuration

Kerberos Internet Explorer

In order to get Internet Explorer to negotiate Kerberos authentication rather than NTLM, make sure:

- The alfresco web server is in the Local Intranet security zone. Check Tools > Internet Options > Security > Local Intranet and make sure that a pattern matching the protocol and domain name is included, e.g. <http://server.com> or http://*.company.com (the IP does not work)
- Automatic login is enabled. Check Tools > Internet Options > Security > Custom Level and make sure *Automatic login with current username and password* is selected.

Kerberos Firefox

When using 'firefox on Windows' as client, you will need to add your alfresco server name to the *network.negotiate-auth.trusted-uris* variable You can access the variable going to the special URL: about:config

When using 'firefox under Linux', you will need to add your alfresco server name to *network.negotiate-auth.trusted-uris* as above but you will need in addition to get a kerberos ticket using the *kinit* command. Note that the ticket can correspond to a different user than your linux username

```
kinit user1
```

where user1 is an active directory user. Notes:

- if the client and the server are on the same machine, you will need to go to the external interface. The loopback interface won't be able to authenticate.
- when using HTTP Kerberos SSO, you will need to enter the hostname in the URL, not the IP.
- You can view your tickets using *klist*.

Kerberos Chrome

Kerberos Chrome on Linux

1) create a ticket on the linux client:

```
kinit -f -p user1
```

```
klist Ticket cache: FILE:/tmp/krb5cc_1000 Default principal: user1@EXAMPLE.FOO
```

```
Valid starting Expires Service principal 14/12/2012 12:10 14/12/2012 22:10 krbtgt/EXAMPLE.FOO@EXAMPLE.FOO renew until 15/12/2012 12:10
```


2) To use alfresco explorer:

chromium --auth-server-whitelist=madona:8080 <http://madona:8080/alfresco>

3) To use share:

chromium --auth-server-whitelist=madona:8080 --auth-negotiate-delegate-whitelist=madona:8080<http://madona:8080/alfresco>

Kerberos Chrome on Windows

on windows the permitted list consists of those servers in the Local Machine or Local Intranet security zone, which is the behavior present in IE. Chrome will thus work with Alfresco Explorer as soon as IE works. There is no need to change AuthServerWhitelist as in Linux.

For Share, however, you will need to do a configuration change as Chrome does not allow by default Kerberos Delegation (which is necessary for Share). On modern versions of Chrome, direct registry edit to the key

```
Software\Policies\Chromium\AuthNegotiateDelegateWhitelist
```

may not be taken into account, so it is recommended to use the Policy Editor **gpedit.msc**

1. Download policy templates from http://dl.google.com/dl/edgedl/chrome/policy/policy_templates.zip
2. Fire up gpedit.msc and do "Computer Configuration" -> right-click "Administrative Templates" -> "Add/Remove templates" -> click "Add..." -> select "windows/adm/en-US/chrome.adm" from the policy_templates.zip download
3. In

```
Local Computer Policy
-> Computer Configuration
-> Administrative Templates
-> Classic Administrative Templates (ADM)
-> Google
-> Google Chrome
-> Policies for HTTP Authentication
-> Kerberos delegation server whitelist
```

Tick 'Enabled' and enter your share server name(s) as value.

4. Fire up Chrome and policy works, chrome://policy lists the settings I made.

Notes Setting **AuthNegotiateDelegateWhitelist** to "" did not work for me and I had to enter the share server name 'madona' **see also**

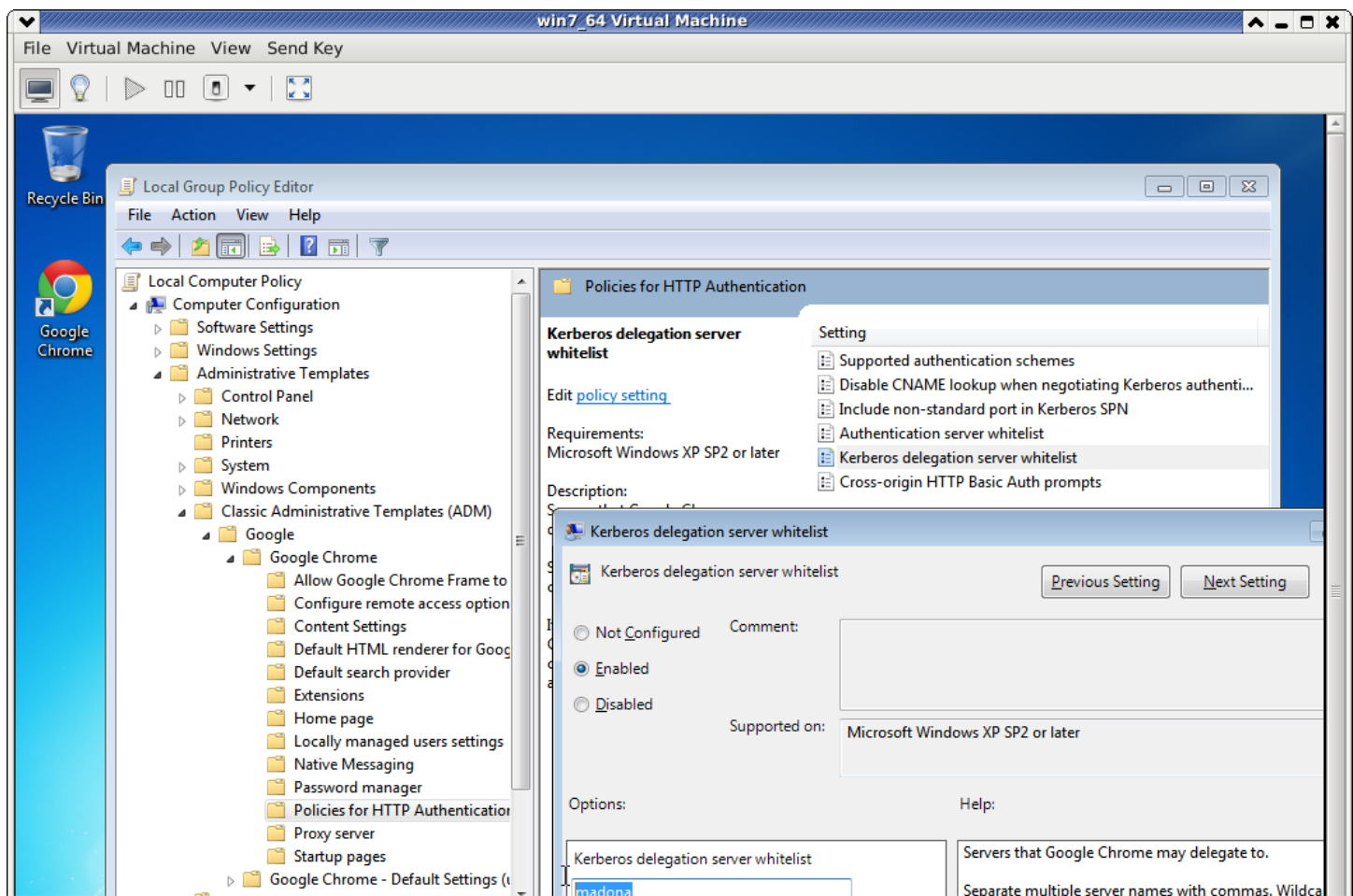
<http://www.chromium.org/administrators/policy-list-3>

<http://www.chromium.org/developers/design-documents/http-authentication>

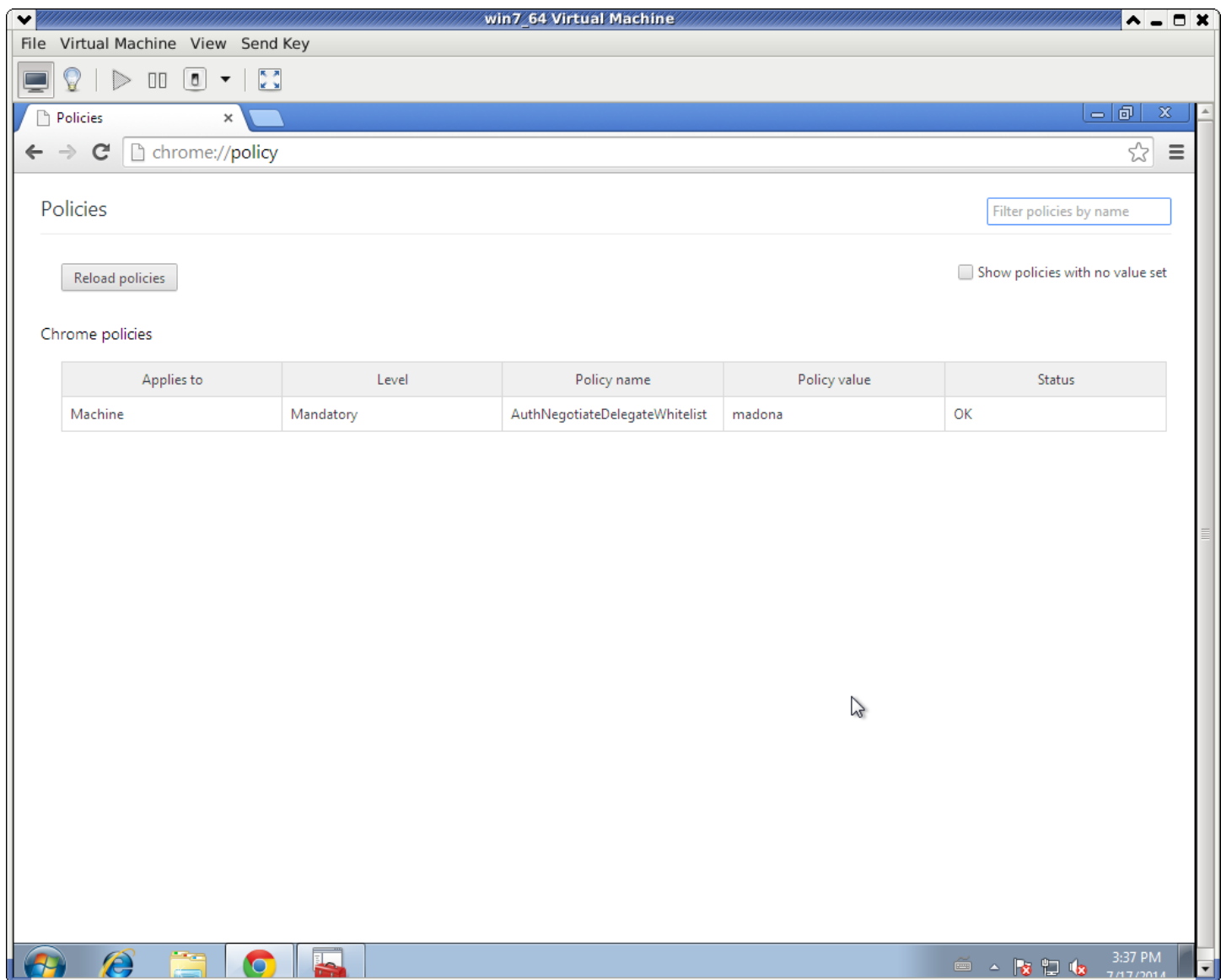
<http://code.google.com/p/chromium/issues/detail?id=259236>

Some screen shots

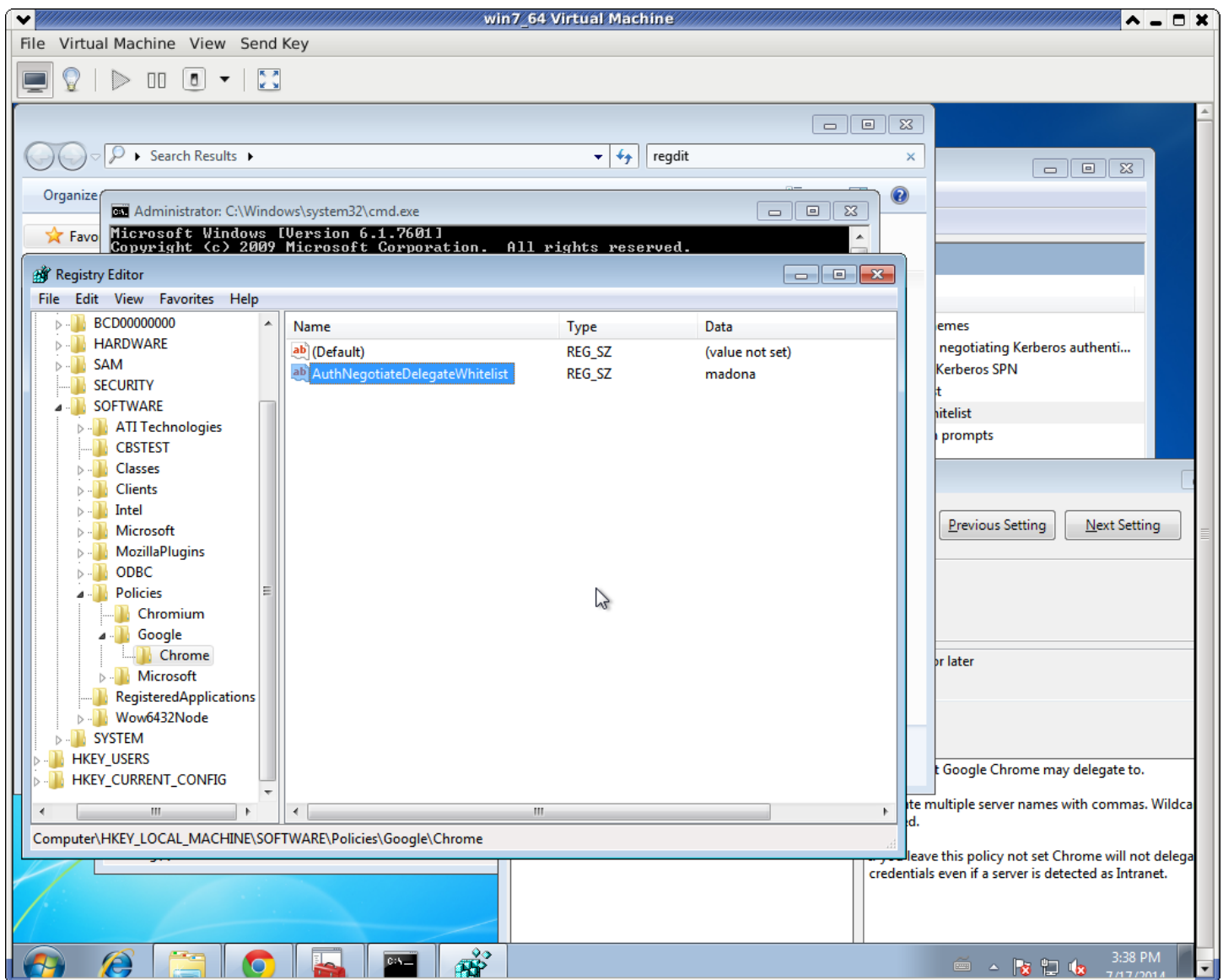
The policy editor:



The policy as seen by Chrome:



The policy as seen in the registry (with 'regedit')



Kerberos Servlet Container Configuration

The servlet container can have some restrictions on the size of the header. For example in tomcat, the size of the header is controlled by `maxHttpHeaderSize` in the `server.xml`.

In the embedded Jetty server used by the VTI Sharepoint server, the size is controlled by `headerBufferSize` in `vti-context.xml`

When using kerberos authentication, headers can become very long (e.g. more than 9000 characters). You thus may want to increase that value to for instance 32768

See <https://issues.alfresco.com/jira/browse/ALF-13810>

Note that because the request data is intercepted by the servlet container, and does not hit the alfresco webapp, you typically do not see any error message in the alfresco logs.

If debugging is 'on' for kerberos, you will see the first part of the request in the logs, e.g.:

```
13:34:50,534 DEBUG [app.servlet.KerberosAuthenticationFilter] New Kerberos auth request from 10.15.107.17 (10.15.107.17:50047)
```

and then nothing.

A way to confirm that your server has a limitation of requests larger than say 10000 bytes is very easy: you can just send a request with more than many characters:

```
for i in `seq 1 10000`; do header="A$header"; done
curl -v http://madona:8080/alfresco/faces/jsp/dashboards/container.jsp -H "Authorization: Negotiate $header"
```

(we neglect the bytes for the string "Authorization: Negotiate " and the other headers in that test)

If the test returns a long HTML content, then your server does accept large headers. if you obtain an error like that:

```
< HTTP/1.1 400 Bad Request
< Server: Apache-Coyote/1.1
< Transfer-Encoding: chunked
< Date: Thu, 17 Jan 2013 15:16:43 GMT
< Connection: close
* Closing connection #0
```

then your server does not accept long headers.

Debugging

You can debug kerberos issues using the log4j properties below:

```
log4j.logger.org.alfresco.web.app.servlet.KerberosAuthenticationFilter=debug
log4j.logger.org.alfresco.repo.webdav.auth.KerberosAuthenticationFilter=debug
```

A sample login output follows:

```
18:46:27,915 DEBUG [app.servlet.KerberosAuthenticationFilter] New Kerberos auth request from 192.168.4.95 (192.168.4.95:38750)
18:46:28,063 DEBUG [app.servlet.KerberosAuthenticationFilter] User user1 logged on via Kerberos
```

Some java options are also helpful:

```
-Dsun.security.krb5.debug=true
-Dsun.security.jgss.debug=true
```

the 'kinit' command can be used to simulate the kerberos queries made at alfresco boot time, e.g:

```
kinit -k -t /etc/keys/alfrescocifs.keytab "cifs/madona.example.foo"
```

(Note on Linux you can also add the -V option) A successful output will be:

```
Authenticated to Kerberos v5
```

An example of failure is:

```
kinit: Cannot contact any KDC for realm 'EXAMPLE.FOO' while getting initial credentials
```

the 'klist' command can be used to look at the ticket cache or at the keytabs. Here is an example with the keytab:

```
klist -k /etc/keys/alfrescocifs.keytab
Keytab name: WRFILE:/etc/keys/alfrescocifs.keytab
KVNO Principal
-----
  4 cifs/madona.example.foo@EXAMPLE.FOO
```

Note that if you modify the accounts in Active Directory, for instance if you tick the "use DES encryption types for this account" option for the cifs and http accounts, you need to "reset" the password of the account AND regenerate the keytab. If not you will get "Checksum Failed" errors. See: <https://issues.alfresco.com/jira/browse/DOC-211> 📄

The servicePrincipalName (SPN) attribute is a multivalued, nonlinked attribute within the Active Directory directory. this attribute can be shown while logged on the Active Directory server using the `setspn -l` command as mentioned above; a sample output is:

```
setspn -l alfrescocifs
Registered ServicePrincipalNames for CN=Alfresco CIFS,DC=example,DC=foo:
cifs/madona.example.foo
cifs/madona
```

But it can thus also be shown with standard LDAP clients, without the need of being logged on the Active Directory server. For instance the commands below shows how to use the "ldapsearch" tool to check the *servicePrincipalName* and *userPrincipalName* of the created users. The example below assumes the AD server is at IP 10.69.69.99, the domain is 'example.foo', and the alfresco server name is 'madona'.

```
ldapsearch -h 10.69.69.99 -x -D "CN=Administrator,CN=Users,DC=example,DC=foo" \
-W -b 'DC=example,DC=foo' '(servicePrincipalName=cifs*)'
```

will show

```
# Alfresco CIFS, example.foo
dn: CN=Alfresco CIFS,DC=example,DC=foo
...
userPrincipalName: cifs/madona.example.foo@EXAMPLE.FOO
servicePrincipalName: cifs/madona
servicePrincipalName: cifs/madona.example.foo
...
```

and

```
ldapsearch -h 10.69.69.99 -x -D "CN=Administrator,CN=Users,DC=example,DC=foo" \
-W -b 'DC=example,DC=foo' '(servicePrincipalName=HTTP*)'
```

will show

```
# Alfresco HTTP, example.foo
dn: CN=Alfresco HTTP,DC=example,DC=foo
...
userPrincipalName: HTTP/madona.example.foo@EXAMPLE.FOO
servicePrincipalName: HTTP/madona
servicePrincipalName: HTTP/madona.example.foo
...
```

Finally, network dumpers (tcpdump, snoop, wireshark, Microsoft Network Monitor, etc...) are the ultimate tools to understand why a kerberos login fails.

To create the network dump files, you need to record all the packets going to/from port 88 of the Active Directory server, eg:

```
tcpdump -s0 -w kerberos_dump1.pcap port 88
```

The example below shows a kerberos login failure with a KRB5KDC_ERR_S_PRINCIPAL_UNKNOWN error; this error occurs because the host name sent (apache.foo) does not correspond to the host the server expects (madona.example.foo). Entering the correct URL and having a DNS setup with no error should solve this kind of issues.

kerb_88.pcap - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: + Expression... Clear Apply

| No. | Time | Source | Destination | Protocol | Info |
|-----|----------|-------------|-------------|----------|--|
| 1 | 0.000000 | 10.69.69.1 | 10.69.69.99 | KRB5 | TGS-REQ |
| 2 | 0.000000 | 10.69.69.99 | 10.69.69.1 | KRB5 | KRB Error: KRB5KDC_ERR_S_PRINCIPAL_UNKNOWN |
| 3 | 0.000000 | 10.69.69.1 | 10.69.69.99 | KRB5 | TGS-REQ |
| 4 | 0.000000 | 10.69.69.99 | 10.69.69.1 | KRB5 | KRB Error: KRB5KDC_ERR_S_PRINCIPAL_UNKNOWN |

Frame 2 (133 bytes on wire, 133 bytes captured)

Ethernet II, Src: CadmusCo_a1:44:82 (08:00:27:a1:44:82), Dst: 36:be:1f:00:0f:58 (36:b)

Internet Protocol, Src: 10.69.69.99 (10.69.69.99), Dst: 10.69.69.1 (10.69.69.1)

User Datagram Protocol, Src Port: kerberos (88), Dst Port: 36124 (36124)

Kerberos KRB-ERROR

Pvno: 5

MSG Type: KRB-ERROR (30)

stime: 2010-08-26 12:15:13 (UTC)

susec: 522040

error_code: KRB5KDC_ERR_S_PRINCIPAL_UNKNOWN (7)

Realm: EXAMPLE.FOO

Server Name (Service and Host): HTTP/apache.foo

Name-type: Service and Host (3)

Name: HTTP

Name: apache.foo

String component that is part of a PrincipalName (kerberos.name_string), 10 bytes

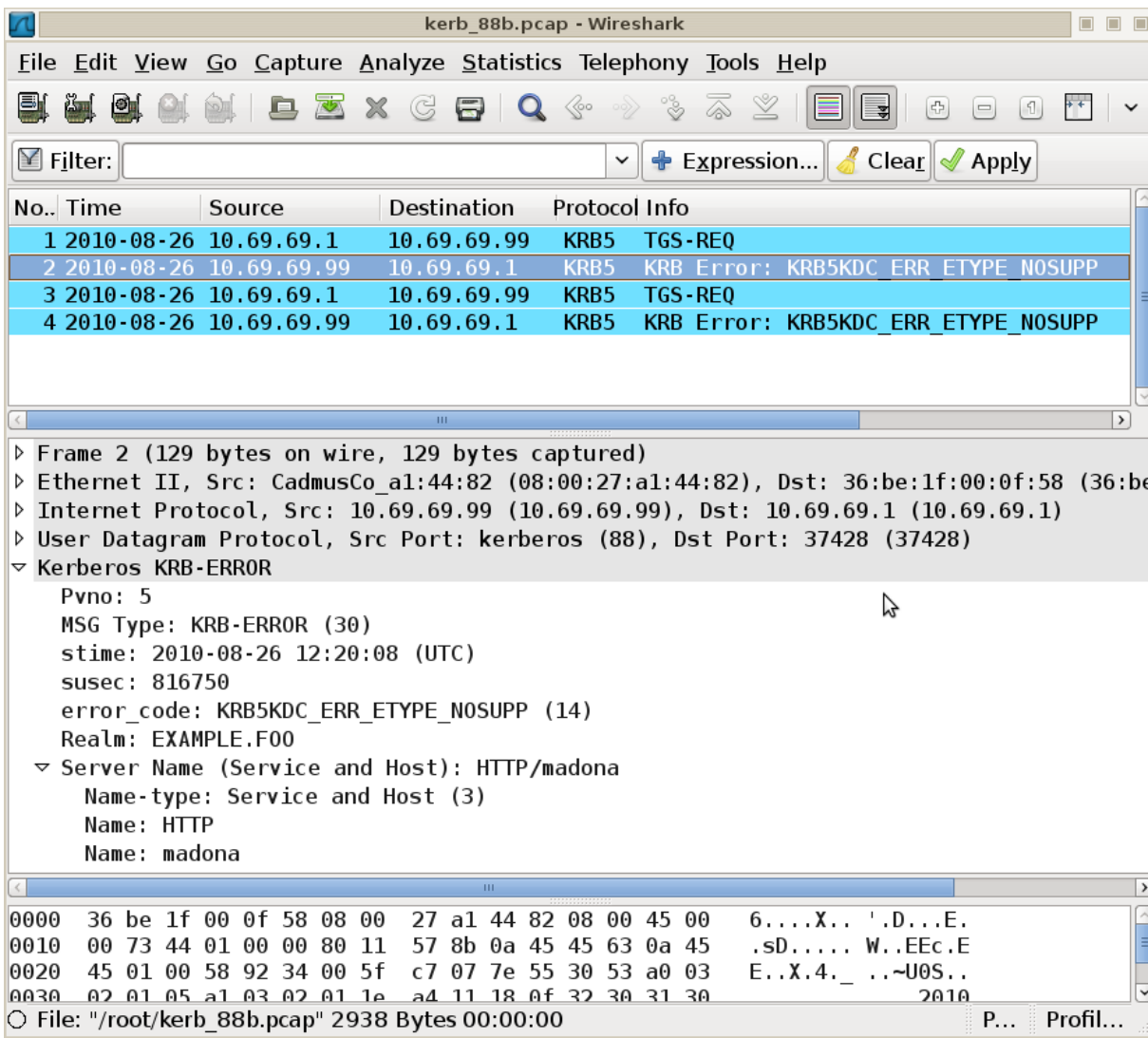
The example below shows a logon failure with a KRB5KDC_ERR_ETYPE_NOSUPP which occurs when the kerberos client uses a encoding type which is not supported by the server. Having the same encoding in the /etc/krb5.conf file i.e lines:

```
default_tkt_enctypes = rc4-hmac
default_tgs_enctypes = rc4-hmac
```

and in the command used to generate the keytabs, i.e. option

```
-crypto RC4-HMAC-NT
```

of the ktpass command, should solve this issue.



The image shows a Wireshark capture of a Kerberos KRB-ERROR packet. The packet list at the top shows four frames: a TGS-REQ, a KRB Error, another TGS-REQ, and another KRB Error. The selected packet (Frame 2) is a KRB Error with the following details:

- Pvno: 5
- MSG Type: KRB-ERROR (30)
- stime: 2010-08-26 12:20:08 (UTC)
- susec: 816750
- error_code: KRB5KDC_ERR_ETYPE_NOSUPP (14)
- Realm: EXAMPLE.FOO
- Server Name (Service and Host): HTTP/madona
 - Name-type: Service and Host (3)
 - Name: HTTP
 - Name: madona

The packet bytes section shows the raw data in hexadecimal and ASCII:

```

0000  36 be 1f 00 0f 58 08 00 27 a1 44 82 08 00 45 00  6...X..'.D...E.
0010  00 73 44 01 00 00 80 11 57 8b 0a 45 45 63 0a 45  .sD....W..EEc.E
0020  45 01 00 58 92 34 00 5f c7 07 7e 55 30 53 a0 03  E..X.4._..~U0S..
0030  02 01 05 a1 03 02 01 1e a4 11 18 0f 32 30 31 30  2010
  
```

The status bar at the bottom indicates the file is "/root/kerb_88b.pcap" with 2938 bytes captured over 00:00:00.

Share Kerberos SSO

This is a new feature introduced in Alfresco release 3.4.0.

Share Server Configuration

The way to modify the Share web application to achieve SSO using the external authentication subsystem is very similar to the way to achieve SSO using NTLM, see section [Alfresco Share SSO using NTLM](#). We underline below the specifics to the kerberos subsystem.

1. Follow the alfresco server instructions above
2. in the Share web-extension folder, copy or rename share-config-custom.xml.sample into share-config-custom.xml
3. Activate the <kerberos> section following the comments in the file and replace the password, realm, and endpoint-spn by the correct values for the AlfrescoHTTP user (which were used earlier to create the keytab files). The Realm value should be capitalized.
4. Uncomment the **two** <config evaluator="string-compare" condition="Remote"> sections.
5. Rename the <config evaluator="string-compare" condition="KerberosDisabled" replace="true"> section to <config evaluator="string-compare" condition="Kerberos" replace="true">.
6. in the (Sun Java) jre/lib/security/java.login.config file created above, add a new section:

```

ShareHTTP {
    com.sun.security.auth.module.Krb5LoginModule required
    storeKey=true
    useKeyTab=true
    keyTab="/etc/keys/alfrescohttp.keytab"
    principal="HTTP/madona.example.foo";
};
  
```

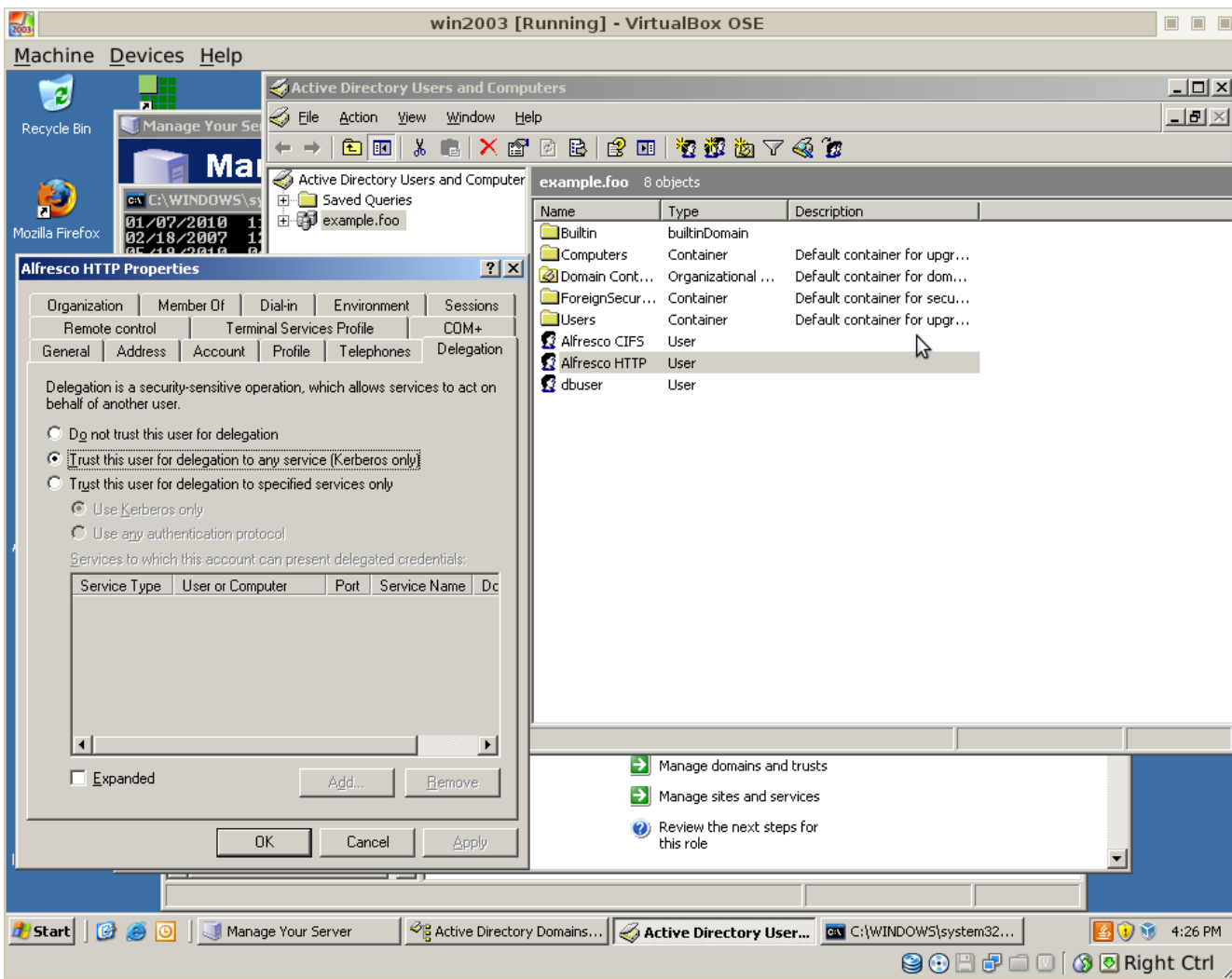
7. Restart the Alfresco server.

Active Directory Configuration

You have to modify the *alfrescohttp* user created during the alfresco kerberos setup described above. In this user "delegation" tab, click the radio button:

Trust this user for delegation to any service (kerberos only)

see screen shot below



If you do not see the delegation tab, follow:

<http://technet.microsoft.com/en-us/library/cc757194%28WS.10%29.aspx>

If you cannot see the Delegation tab, do one or both of the following:

- Register a Service Principal Name (SPN) for the user account with the Setspn utility in the support tools on your CD. Delegation is only intended to be used by service accounts, which should have registered SPNs, as opposed to a regular user account which typically does not have SPNs.
- Raise the functional level of your domain to Windows Server 2003. For more information, see below:

To raise the domain functional level see <http://technet.microsoft.com/en-us/library/cc776703%28WS.10%29.aspx> :

1. Open Active Directory Domains and Trusts.
2. In the console tree, right-click the domain for which you want to raise functionality, and then click Raise Domain Functional Level.
3. In Select an available domain functional level, do one of the following:
 - To raise the domain functional level to Windows 2000 native, click Windows 2000 native, and then click Raise.
 - To raise domain functional level to Windows Server 2003, click Windows Server 2003, and then click Raise.

This will cause the delegation tab to appear, so right-click on user "Alfresco HTTP", select properties, tick "Trust this user for delegation to any service (kerberos only)". Click Apply. Click OK.

Client Configuration

Windows Clients Configuration

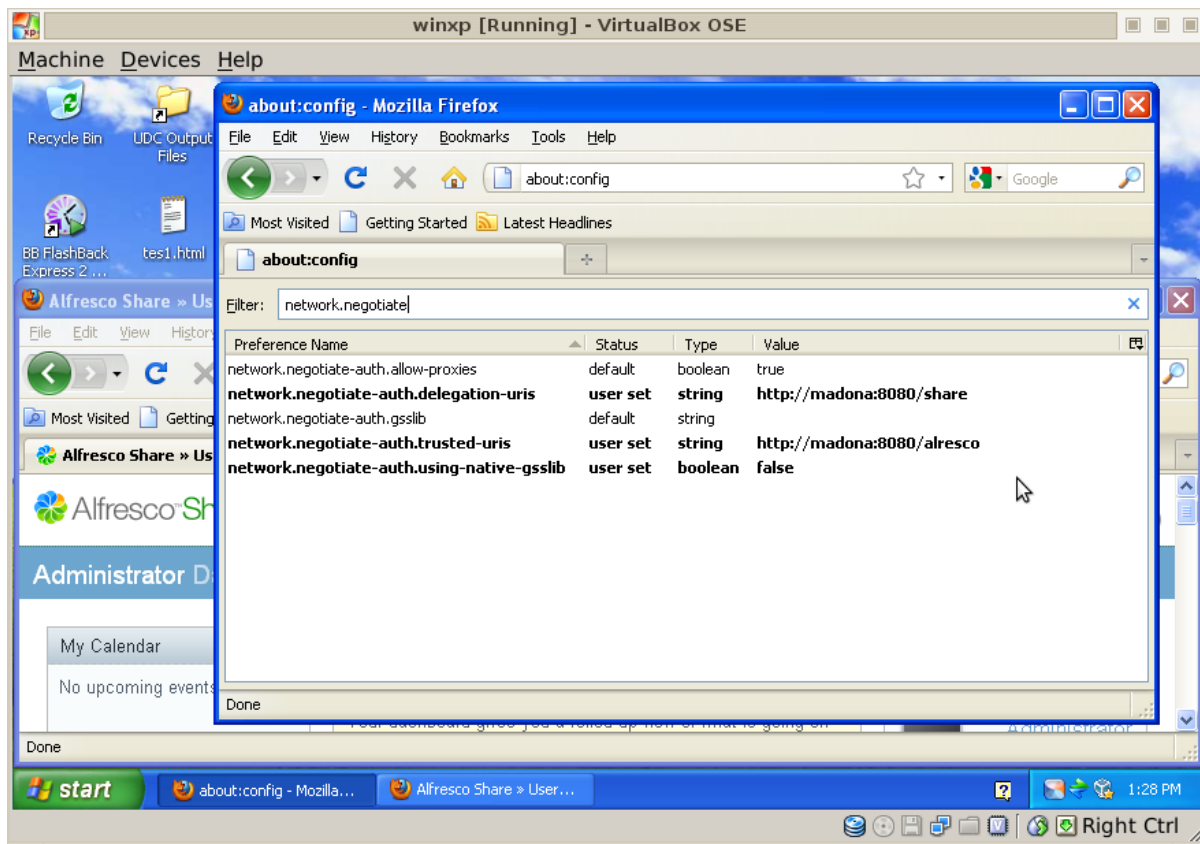
IE configured as above should work without modifications. However, if you see in alfresco.log errors like these:

```
14:56:28,574 WARN [site.servlet.KerberosSessionSetupPrivilegedAction] credentials can not be delegated!
```

This means that the delegation was not configured correctly on the AD server.

To make Firefox work on windows on the share URL with kerberos SSO, you need to modify (in the about:config special URL) the variables below:

```
network.negotiate-auth.delegation-uris
network.negotiate-auth.trusted-uris
network.negotiate-auth.using-native-gsslib
```

for instance:

If you forget to set:

```
network.negotiate-auth.delegation-uris
```

you will have in your alfresco logs errors like those:

```
14:56:28,574 WARN [site.servlet.KerberosSessionSetupPrivilegedAction] credentials can not be delegated!
```

Linux Clients Configuration

Linux clients (Firefox) do not work: see <http://issues.alfresco.com/jira/browse/ALF-6286>

Note that the kinit command has to set the 'Forwardable' flag to true when creating the ticket:

```
kinit -f -p user1
```

A traffic dump will show the flag, see screen shot below:

The image shows a Wireshark 1.8.2 interface with a packet capture named 'dump1.pcap'. The packet list shows frames 40 to 51. Frame 43 is selected, showing details for a Kerberos TGS-REQ message. The KDCOptions field is expanded, showing flags: Forwardable (1), Forwarded (1), Proxiable (0), and Allow Postdate (0). The packet bytes pane shows the raw data for the selected packet.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-----------------|-------------|-------------|----------|--------|----------------------------------|
| 40 | 10:19:45.936994 | 10.69.69.99 | 10.69.69.1 | KRB5 | 1306 | TGS-REP |
| 41 | 10:19:45.937003 | 10.69.69.99 | 10.69.69.1 | KRB5 | 1306 | TGS-REP |
| 42 | 10:19:45.937003 | 10.69.69.99 | 10.69.69.1 | KRB5 | 1306 | TGS-REP |
| 43 | 10:19:45.941390 | 10.69.69.1 | 10.69.69.99 | KRB5 | 1257 | TGS-REQ |
| 44 | 10:19:45.941398 | 10.69.69.1 | 10.69.69.99 | KRB5 | 1257 | TGS-REQ |
| 45 | 10:19:45.942343 | 10.69.69.99 | 10.69.69.1 | KRB5 | 1216 | TGS-REP |
| 46 | 10:19:45.942350 | 10.69.69.99 | 10.69.69.1 | KRB5 | 1216 | TGS-REP |
| 47 | 10:19:45.942350 | 10.69.69.99 | 10.69.69.1 | KRB5 | 1216 | TGS-REP |
| 48 | 10:19:45.962559 | 127.0.0.1 | 127.0.0.1 | HTTP | 191 | GET /alfresco/wcs/touch HTTP/1.1 |
| 49 | 10:19:45.963469 | 127.0.0.1 | 127.0.0.1 | HTTP | 575 | HTTP/1.1 401 Unauthorized (text) |
| 50 | 10:19:45.963491 | 127.0.0.1 | 127.0.0.1 | TCP | 56 | 36967 > http-alt [ACK] Seq=620 A |
| 51 | 10:19:45.965593 | 127.0.0.1 | 127.0.0.1 | TCP | 90 | [TCP segment of a reassembled PD |

Frame 43: 1257 bytes on wire (10056 bits), 1257 bytes captured (10056 bits)

- Linux cooked capture
- Internet Protocol Version 4, Src: 10.69.69.1 (10.69.69.1), Dst: 10.69.69.99 (10.69.69.99)
- User Datagram Protocol, Src Port: 60456 (60456), Dst Port: kerberos (88)
- Kerberos TGS-REQ
 - Pvno: 5
 - MSG Type: TGS-REQ (12)
 - padata: PA-TGS-REQ
 - KDC_REQ_BODY
 - Padding: 0
 - KDCOptions: 60000000 (Forwardable, Forwarded)
 - ..1.. = Forwardable: FORWARDABLE tickets are allowed/requested
 - ..1.. = Forwarded: This ticket has been FORWARDED
 - ...0 = Proxiable: Do NOT use proxiable tickets
 - ...0 = Proxy: This ticket has NOT been proxied
 - ...0 = Allow Postdate: We do NOT allow the ticket to be postdated

0490 05 00 60 00 00 00 a2 0d 1b 0b 45 58 41 4d 50 4c ..EXAMPLE
 04a0 45 2e 46 4f 4f a3 20 30 1e a0 03 02 01 02 a1 17 E.F00.0.....
 04b0 30 15 1b 06 6b 72 62 74 67 74 1b 0b 45 58 41 4d 0...krbt gt..EXAM
 04c0 50 4c 45 2e 46 4f 4f a5 11 18 0f 31 39 37 30 30 PLE.F00. ...19700
 04d0 31 30 31 30 30 30 30 30 5a a7 06 02 04 50 f9 10100000 07...P.

Known issues

Share Kerberos SSO does not work with IBM java <http://issues.alfresco.com/jira/browse/ALF-6284>

External

Introduction

The external authentication subsystem correspond to the case where you do not want to use any of the other authentication subsystems available but prefer to use an external authentication layer. This layer authenticates the user and then passes the authenticated user username to alfresco. There are two main mechanisms to pass the authenticated user username to alfresco: one based on the servlet standard and one based on the HTTP protocol. Usually you are looking for a loose coupling between alfresco and your authentication layer where the coupling consists in passing the authenticated user username to alfresco from the authentication layer. External authentication can implement several SSO (Single Sign On) mechanisms.

REMOTE_USER CGI variable based external authentication

The external authentication subsystem can be used to integrate Alfresco with any external authentication system that can be integrated with your application server in such a way that the identity of the logged-in user is passed to servlets via the `HttpServletRequest.getRemoteUser()` method. As this is the standard way for application servers to propagate user identities to servlets, it should be compatible with a number of SSO solutions, including CAS.

In practice this method can be used when the authentication layer can talk the AJP protocol. An apache web server using mod_proxy and an authentication module will be able to use this method.

Another case where this method can be used is when you have a java class doing the authentication. This method is usually more intrusive because it typically either require a modification of the application web.xml file to include additional filter or relies on the application server features (WAS (WebSphere Application Server) has LTPA. Tomcat has valves...). Modifying the web.xml file is in general to be discouraged.

Header based external authentication

The subsystem also allows a 'proxy' user to be configured, such that requests made through this proxy user are made in the name of an alternative user whose name is carried in a configured HTTP request header. This allows, for example the Share application and other Alfresco Surf applications to act as a client to an SSO-protected Alfresco application and assert the user name in a secure manner. This can be used with most hardware authentication devices or authentication appliances.

Examples

See [Alfresco With mod_auth_cas](#) for a step-by-step example of configuring the Alfresco Explorer and Share applications behind the Apache mod_auth_cas CAS client using the REMOTE_USER CGI variable passed by AJP..

See [Alfresco With Shibboleth](#) for a step-by-step example of configuring the Alfresco Explorer and Share applications behind Shibboleth.

Configuration

The external subsystem supports the following properties. See [Configuring Subsystems](#) for how to configure these.

external.authentication.enabled

A Boolean property that when `true` indicates that this subsystem is active and will trust remote user names asserted to it by the application server.

external.authentication.defaultAdministratorUserNames

A comma separated list of user names who should be considered administrators by default.

external.authentication.proxyUserName

The name of the remote user that should be considered the 'proxy user'. The default is `alfresco-system`. Requests made by this user will be made under the identity of the user named in the HTTP Header indicated by the `external.authentication.proxyHeader` property. If not set, then the HTTP Header indicated by the `external.authentication.proxyHeader` property is always assumed to carry the user name (but please note that this would not be secure unless this application is not directly accessible by other clients).

external.authentication.proxyHeader

The name of the HTTP header that carries the name of a proxied user. The default is `X-Alfresco-Remote-User`, as used by the Alfresco Repository application.

external.authentication.userIdPattern

An optional regular expression to be used to extract a user ID from the HTTP header. The portion of the header matched by the first bracketed group in the regular expression will become the user name. If not set (the default), then the entire header contents are assumed to be the proxied user name.

Alfresco Share using external SSO

The way to modify the Share web application to achieve SSO using the external authentication subsystem is very similar to the way to achieve SSO using NTLM, see section [Alfresco Share SSO using NTLM](#).

We detail the configuration for the most recent versions below. If you use **older versions**, please refer to [Alfresco Share SSO using NTLM](#) to understand the differences. The main differences are on the file name used to define the connector endpoints and their parameters and the changes that may need to be applied to the Share application `web.xml` file.

Since 3.4.9 and 4.0.2, the Share application may be configured to accept a user name from an HTTP header provided by an external authentication system. Prior to this, a custom filter was required even though the Alfresco Explorer application and Repository would have accepted a header value as an indication that the user was authenticated.

You need to make a configuration change to the Share application. This is very similar to using NTLM with Share. Find the `.sample` configuration override file

```
\tomcat\shared\classes\alfresco\web-extension\share-config-custom.xml.sample
```

Copy and rename the file to:

```
\tomcat\shared\classes\alfresco\web-extension\share-config-custom.xml
```

Then edit the file:

- uncomment the following two sections
- change the connector used by the endpoint in the second section to use **alfrescoHeader** rather than `alfrescoCookie`.
- set the name of the header used by the external SSO in the **userHeader** element of the `alfrescoHeader` connector
- change the **endpoint-url** value to point to your Alfresco Server location.

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

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

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

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

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

NOTE: For external SSO switch the endpoint connector to "AlfrescoHeader" and set
the userHeader to the name of the HTTP header that the external SSO
uses to provide the authenticated user name.
-->
<config evaluator="string-compare" condition="Remote">
  <remote>
    <keystore>
      <path>alfresco/web-extension/alfresco-system.p12</path>
      <type>pkcs12</type>
      <password>alfresco-system</password>
    </keystore>

    <connector>
      <id>alfrescoCookie</id>
      <name>Alfresco Connector</name>
      <description>Connects to an Alfresco instance using cookie-based authentication</description>
      <class>org.alfresco.web.site.servlet.SlingshotAlfrescoConnector</class>
      <userHeader>SsoUserHeader</userHeader>
    </connector>
```

```

<connector>
  <id>alfrescoHeader</id>
  <name>Alfresco Connector</name>
  <description>Connects to an Alfresco instance using header and cookie-based authentication</description>
  <class>org.alfresco.web.site.servlet.SlingshotAlfrescoConnector</class>
  <userHeader>SsoUserHeader</userHeader>
</connector>

<endpoint>
  <id>alfresco</id>
  <name>Alfresco - user access</name>
  <description>Access to Alfresco Repository WebScripts that require user authentication</description>
  <connector-id>alfrescoHeader</connector-id>
  <endpoint-url>http://localhost:8080/alfresco/wcs</endpoint-url>
  <identity>user</identity>
  <external-auth>true</external-auth>
</endpoint>
</remote>
</config>

```

To configure both Share and the Alfresco Repository to use the same HTTP header value the `external.authentication.proxyHeader` property needs to be set to the same value as the `userHeader` value.

```
external.authentication.proxyHeader=SsoUserHeader
```

See <https://issues.alfresco.com/jira/browse/MNT-2766> if Alfresco RSS feed returns 401 HTTP error code. You have to modify the appropriate section in `share-config-custom.xml` comparing with the same file provided in issue <https://issues.alfresco.com/jira/browse/MNT-2766>

Security concerns

Keep in mind that, [as stated in this forum post](#), activating external authentication makes Alfresco accept external authentication tokens, so you should make sure that no untrusted direct access to Alfresco's HTTP or AJP ports is allowed.

Testing the REMOTE_USER CGI variable

When external authentication relies on the value of the `REMOTE_USER` CGI variable (typically set by an apache module and then passed to the alfresco layer using AJP), it could be useful to confirm the CGI variable is correctly passed.

One easy way to do this is to create a simple test.jsp file in: `'tomcat/webapps/alfresco/test.jsp'`:

```

getRemoteUser() is set to:
<% out.print (request.getRemoteUser()); %>

```

and go to:

<http://apacheserverfrontend.foo/alfresco/test.jsp>

Idem with the Share app. See for instance <http://issues.alfresco.com/jira/browse/ALF-4230>

NTLM dependency on HTTP keep alives

NTLM authentication works only when `KeepAlive` is on. NTLM authenticates a connection not a request, see for instance <http://davenport.sourceforge.net/ntlm.html#ntlmHttpAuthentication>. After receiving a response from a server with a `WWW-Authenticate: NTLM` header, the client resubmits the request with an "Authorization" header containing a Type 1 message parameter. The Type 1 message is Base-64 encoded for transmission. From this point forward, the connection is kept open; closing the connection requires reauthentication of subsequent requests. This implies that the server and client must support persistent connections, via either the HTTP 1.0-style "Keep-Alive" header or HTTP 1.1 (in which persistent connections are employed by default).

See also <https://issues.alfresco.com/jira/browse/ALF-6300>

Categories: [Security](#) | [Authentication](#) | [Single Sign On](#) | [LDAP](#) | [NTLM](#) | [Kerberos](#) | [CAS](#) | [Subsystems](#) | [3.2](#)