jahia

CMIS CONNECTOR MODULE DOCUMENTATION

DIGITAL EXPERIENCE MANAGER 7.2

SUMMARY

1 OVERVIEW

1.1 ABOUT CMIS

Content Management Interoperability Services (CMIS) is an open standard that allows different content management systems to inter-operate over the Internet. Specifically, CMIS defines an abstraction layer for controlling diverse document management systems and repositories using web protocols.

CMIS defines a domain model plus bindings that can be used by applications. OASIS, a web standards consortium, approved CMIS as an OASIS Specification on May 1, 2010. CMIS 1.1 has been approved as an OASIS specification on December 12, 2012.

CMIS provides a common data model covering typed files and folders with generic properties that can be set or read. There is a set of services for adding and retrieving documents ('objects'). There may be an access control system, a checkout and version control facility, and the ability to define generic relations. Three protocol bindings are defined, one using WSDL and SOAP, another using AtomPub,[4] and a last browser-friendly one using JSON. The model is based on common architectures of document management systems.

1.2 ABOUT THIS MODULE

This module makes it possible to mount CMIS providers as external JCR content providers, making it possible to access folders and files stored in a CMIS repository as if they were part of the local JCR data structure. It then makes it possible to use all of Digital Experience Manager's UIs to manipulate content from a CMIS repository, or even add mixin on top of CMIS external nodes (such as ratings or comments!).

The module makes it possible to:

- Integrate an external repository that complies to the CMIS standard.

- Provide full CRUD support for CMIS repository inside the Digital Experience ManagerJCR repository
- Use JCR Search to search inside connected CMIS repository

NOTE

This module is not intended to work with an earlier than 7.1.2 DX version.

1.3 MODULE FEATURES

1. The two base CMIS types (document and folder) and any custom sub-types are supported.
2. Full Create, Read, Update and Delete (CRUD) support along with renaming and moving for all supported types. It is also possible to modify content and properties values.
3. Both path and id identification are supported by external provider.
4. Ordering is not supported, as CMIS does not support such a feature.
5. Searching inside a CMIS repository is supported with the following restrictions (see TestQuery.txt for demo queries).
    - Join is not implemented. We return only one node path per row.
    - lower, upper, length, score - are not supported.
    - non mapped properties acts as null.
    - any specific behavior for multivalued properties is not implemented.
    - date type is supported only if cast expression is used (no auto converting).
6. Configurable mapping for custom types. CMIS does not supports language-specific attributes, so if Jahia queries use language conditions they will not be mapped to CMIS queries.

1.4 IMPLEMENTATION NOTES

The underlying CMIS implementation CMIS used is Apache Chemistry. In the default configuration the external CMIS repository is mounted at the /external-cmis-mapped path. To map CMIS types to JCR types we use node types cmis:folder and cmis:document. Common properties are declared in a separate mixin called cmis:base.

If it is needed to map custom CMIS type this can be achieved by extending the cmis:folder or cmis:document nodetypes. Mappings for custom types must then be configured in the spring deployment descriptor of the module. The CmisTypeMapping bean supports type inheritance, so no property mapping duplication is needed.

2 CONFIGURATION

2.1 INSTALLATION

Installing the CMIS provider module is straight-forward. Simply copy the JAR into the digital-factory-data/modules directory or use the administration « Manage Modules » interface to either install it by uploading the file or installing from a public or private Jahia App Store.

2.2 CONNECTING TO AN OPENCMIS IN-MEMORY REPOSITORY

The easiest way to create test environment is to use *OpenCMIS InMemory Repository* https://chemistry.apache.org/java/developing/repositories/dev-repositories-inmemory.html InMemory Repository can be deployed either in separate Tomcat or in the same as for Jahia. Separate Tomcat is preferable because webapp's startup order can not be configured. In other case you must be sure you don't access mount point on startup.

OpenCMIS Workbench may be used as CMIS client.

1. Download and install a new Tomcat instance in which we will deploy the OpenCMIS server. You can download Tomcat from : http://tomcat.apache.org
2. Download the OpenCMIS server from the following location: http://chemistry.apache.org/java/download.html . You should use the "OpenCMIS Server Webapps" commodity package
3. Unpack chemistry-opencmis-server-inmemory-xxx.war into tomcat/webapps/inmemory folder. The repository config file is located in inmemory\WEB-INF\classes\repository.properties. Default configuration is should be ok.
4. You need to adjust the Tomcat configuration to listen on different ports if running it on the same machine as your Digital Experience Manager installation. To do so modify the ports in TOMCAT_INSTALL_DIR/conf/server.xml
5. Start your Tomcat CMIS instance using the startup scripts in TOMCAT_INSTALL_DIR/bin/startup.sh/.bat
6. Go to DX Server Administration and go to System Components / Mount points. (see Connection setup part down below for a complete description of the installation process)
7. Fill in the form with these entries. Configure port, depending on your tomcat config. Make sure you specify a meaningful mount name such as "opencmis"

```
Mount point name=opencmis
repository id=A1
user=dummyuser
password=dummysecret
url=http://localhos:8080/chemistry-opencmis-server-inmemory-1.0.0/atom11
```

Once you confirm the entry there will be CMIS repository mounted in the "External Repositories" of the DX Document Manager tab under /mounts/opencmis.

3 USING THE CMIS ALFESCO CONNECTOR

3.1 PREREQUISITIES

The current version of the connector works with Alfresco version 4.2 and 5.0.

Alfresco and DX must share the same users. This can be achieved by connecting both DX and Alfresco to the same user directory (LDAP for example)

3.2 INSTALLATION

3.2.1 ALFRESCO SERVER

3.2.1.1 DX Plugin on Alfresco Server

If you use CMIS-Provider 2.1.x in Portal Factory 2.1.x, to make the impersonification work, you need to download alfresco-auth-amp plugin and install it in your Alfresco server by using the following command:

java -jar bin/alfresco-mmt.jar install alfresco-auth-amp-1.0.0.amp tomcat/webapps/alfresco –nobackup

NOTE

Copy the downloaded alfresco-auth-amp-1.0.0.amp file to the root folder of your alfresco install before using the command.

3.2.1.2 Alfresco Configuration

If the Alfresco has more than 1000 authenticated users, set the cache size of the ticket cache to a value above the number of users with access to Alfresco. In the configuration file WEB-INF/classes/cache.properties, set the property cache.ticketsCache.maxItems to the maximum number of authentified user that will log on DX to access Alfresco content.

3.2.2 CMIS provider module in DX

3.2.2.1 Module installation

Install and configure the CMIS provider module in DX. If the jar file is not already available, this module can be found on the Jahia AppStore, with a search on the Module Administration user interface or on the customer extranet section of Jahia.com.

Deploy the module through the Module Administration user interface, or copy the jar file in the digital-factory-data/modules folder.

3.2.2.2 Module Configuration

You can set your custom configuration in the jahia.properties file.

3.2.2.2.1 CMIS properties

Here is the list of all available CMIS properties:

https://chemistry.apache.org/java/developing/dev-session-parameters.html

3.2.2.2.2 Cmis Session Cache

- Tuning cache value

- org.jahia.cmis.session.cache.concurrencyLevel=4

- number of sessions cached
  - org.jahia.cmis.session.cache.maximumSize=128
- time (in minutes) the session are stored
  - org.jahia.cmis.session.cache.expireAfterAccess=30

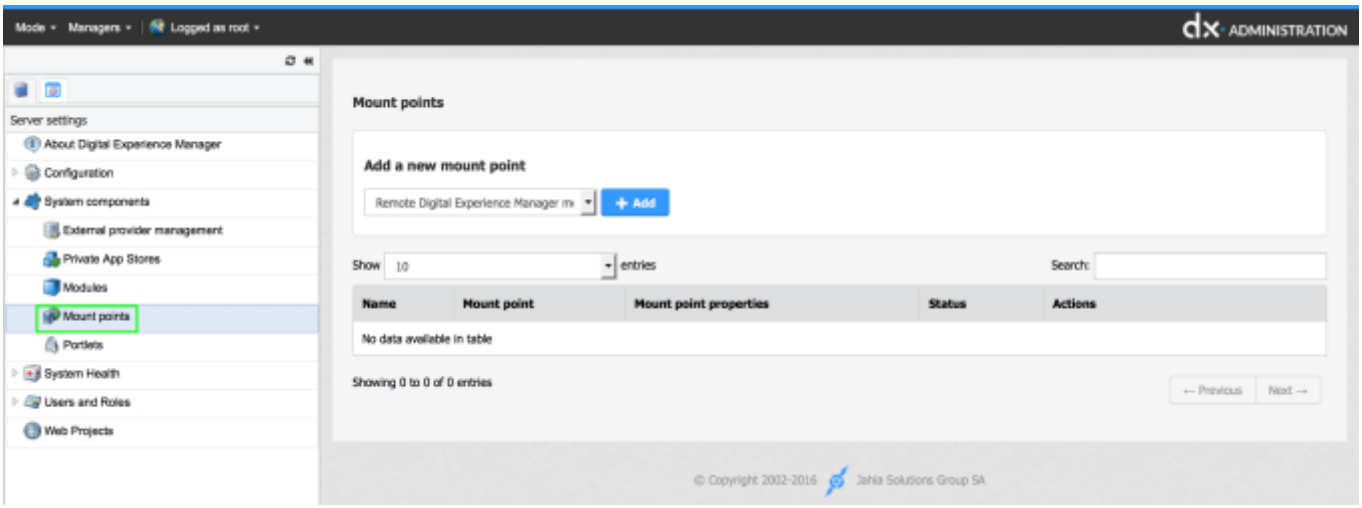
3.2.2.2.3 Other

Maximum number of children (e.g. files) returned on a node, if set to 0, return all the children (default is 0), if set to 100 only the first 100 nodes are listed.

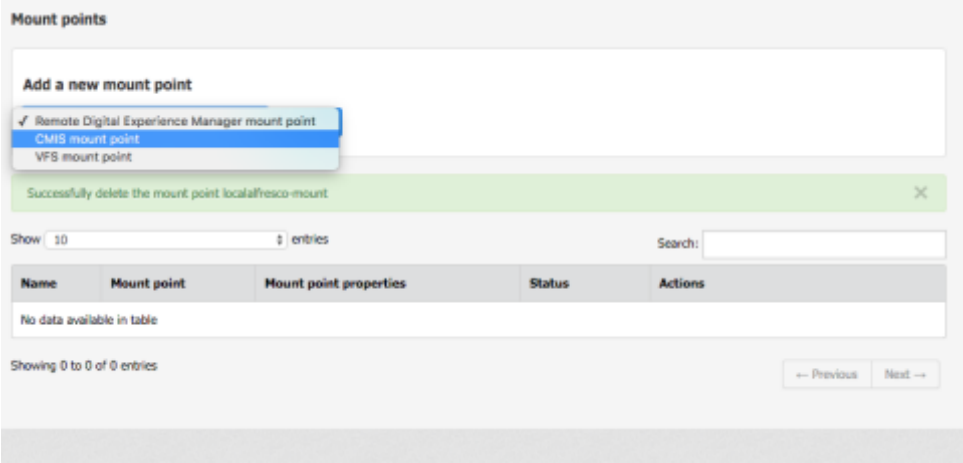
org.jahia.cmis.max.child.nodes=0

3.3 CONNECTION SETUP

Open the DX Server Administration and go to System Components / Mount points.



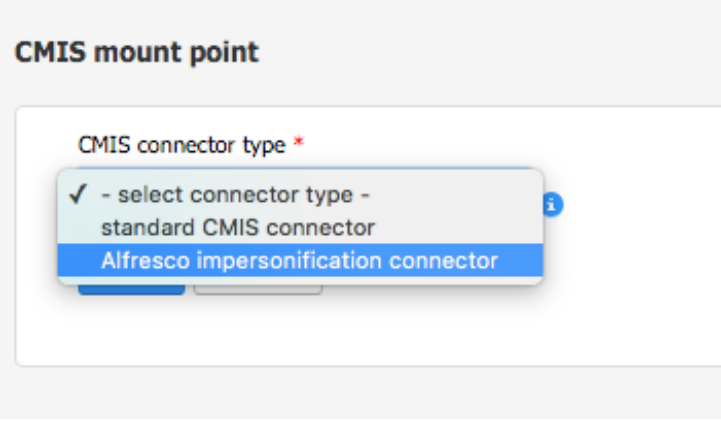
Add a new point, select CMIS mount point



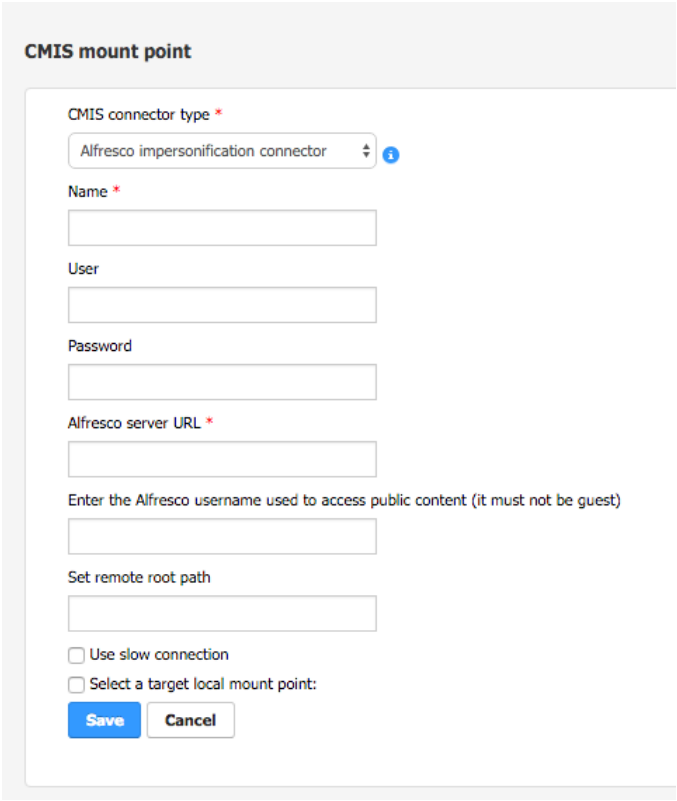
Two type of CMIS Connector to the remote server are available:

- Standard CMIS Connector: This is the default CMIS connector to any CMIS server (including Alfresco). All users will share the same connection.
- Alfresco impersonification connector: this connector is only for Alfresco, allowing the connection to the remote server as the current user.

Select "Alfresco impersonification connector"

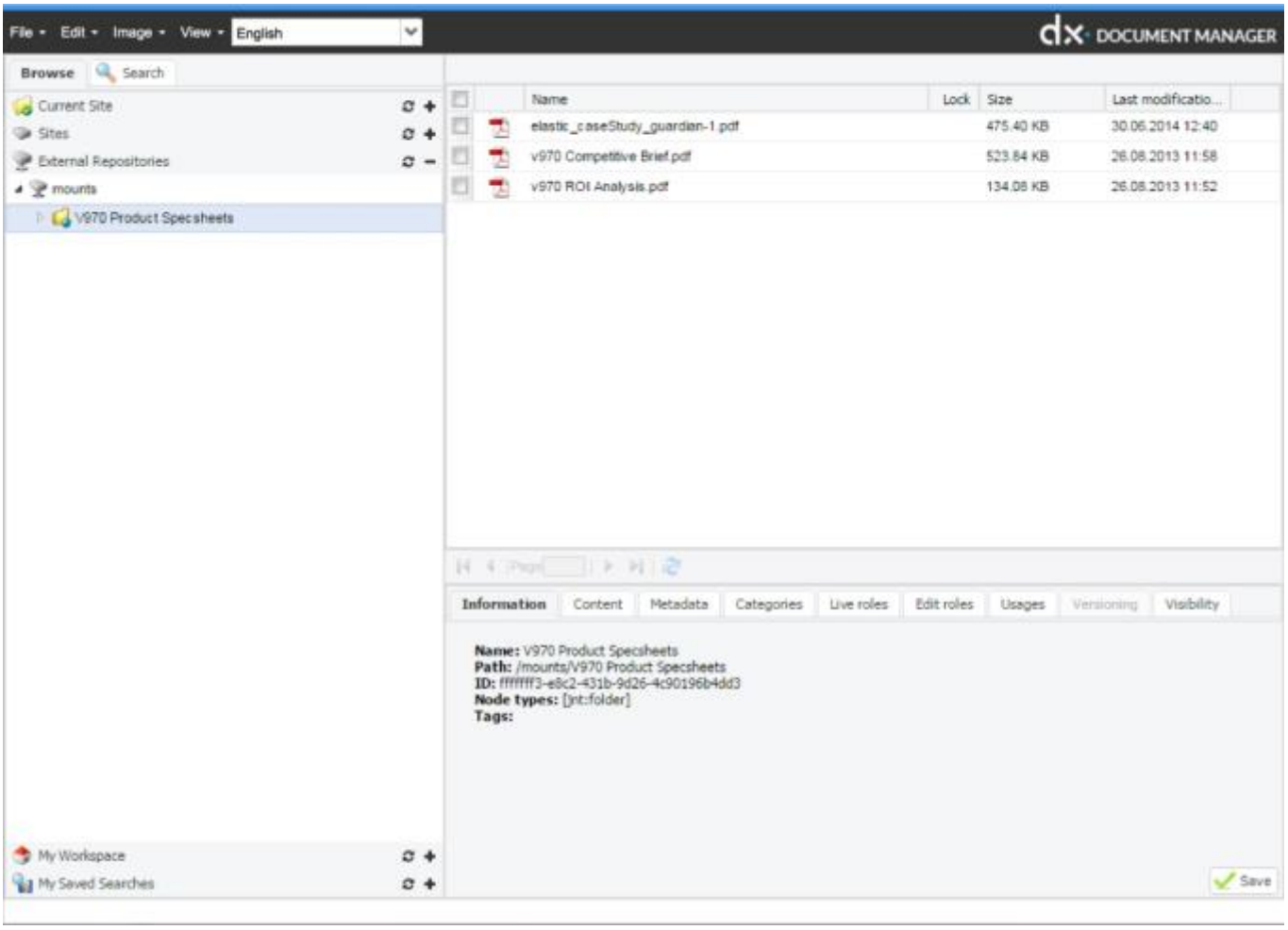


Fill the form to set up the mount point

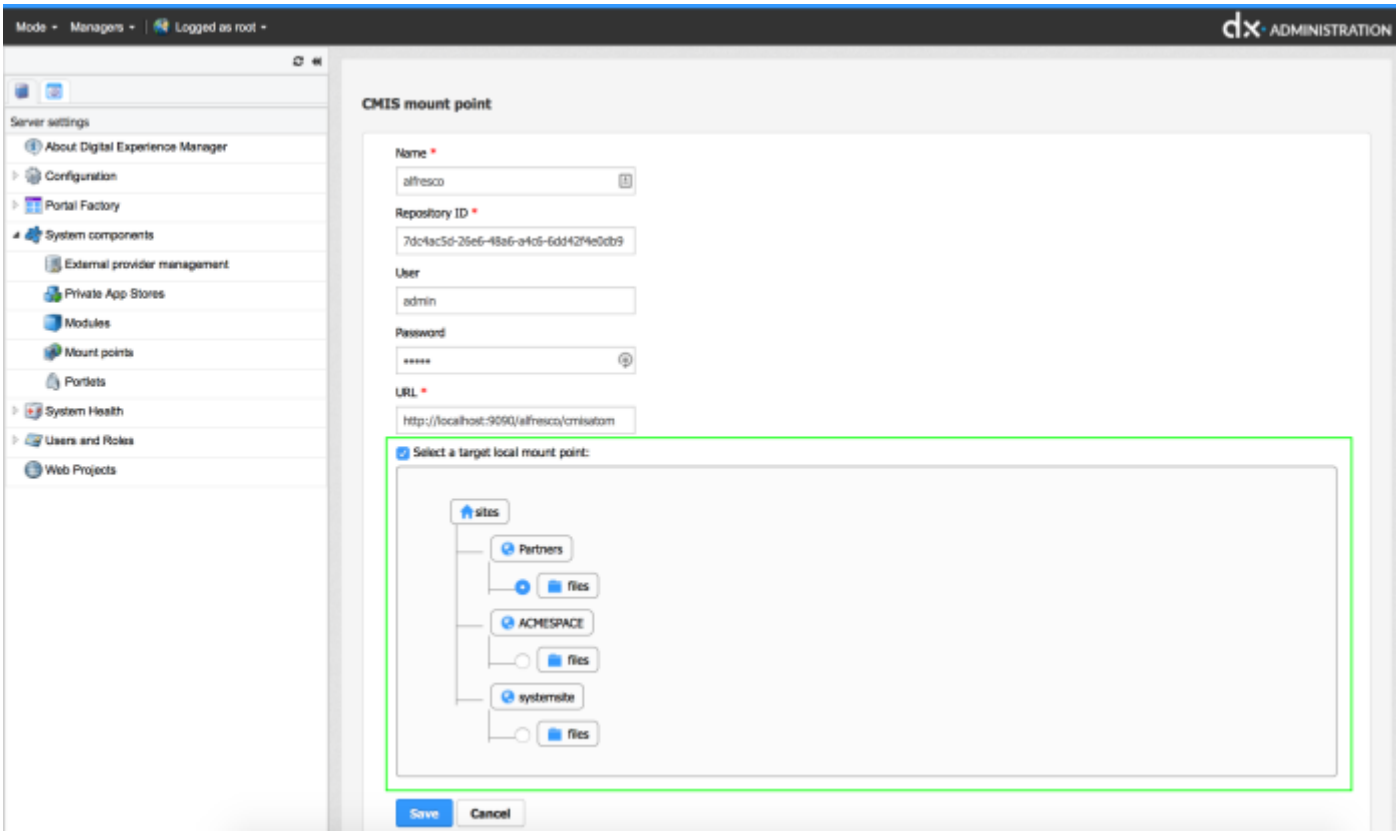


- Name: is the folder name of the mount point. This name will be part of the path of the external content.
- User: Alfresco admin user used to generate ticket and browse the repository as system.
- Password: password of the admin user
- Alfresco server URL: Address of the Alfresco server, it points to the alfresco servlet on Alfresco server. Example: http://my.host/alfresco
- Alfresco user name used to access public content: as CMIS endpoints cannot be accessed by guest user, a user must be set to browse contents as guest. This user can be any user that has access to Alfresco. There is no specific need regarding the rights this user has on alfresco.
- Remote root path: Root path on the remote server where to connect.
- Use slow connection: When activated, most of the queries that ask for child nodes are disabled (see limitations)
- Local mount point: Use the picker to set where the provider has to be mounted locally.

If you don't select a local target, the mount point will appear under External repositories > Mounts in the managers and pickers.

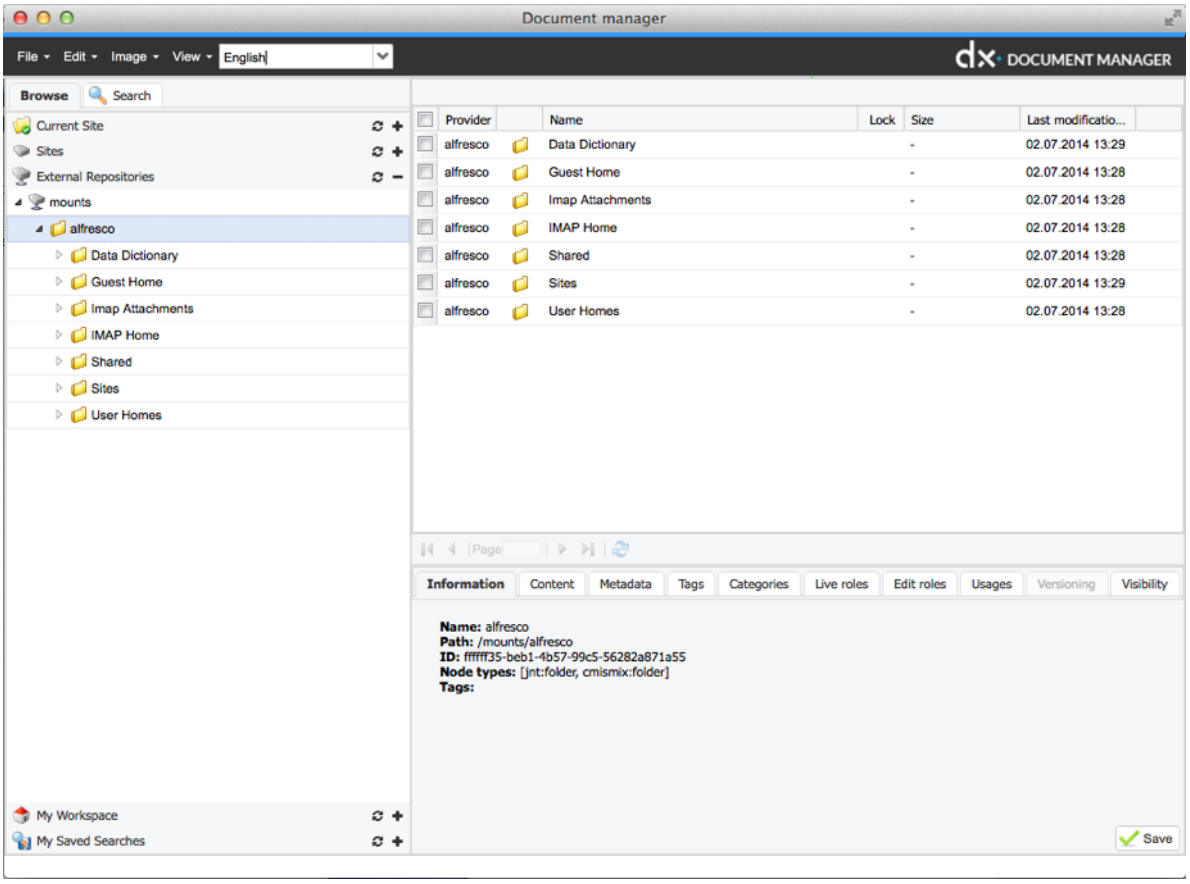


If you select a local target, you can choose where this mount point will appear



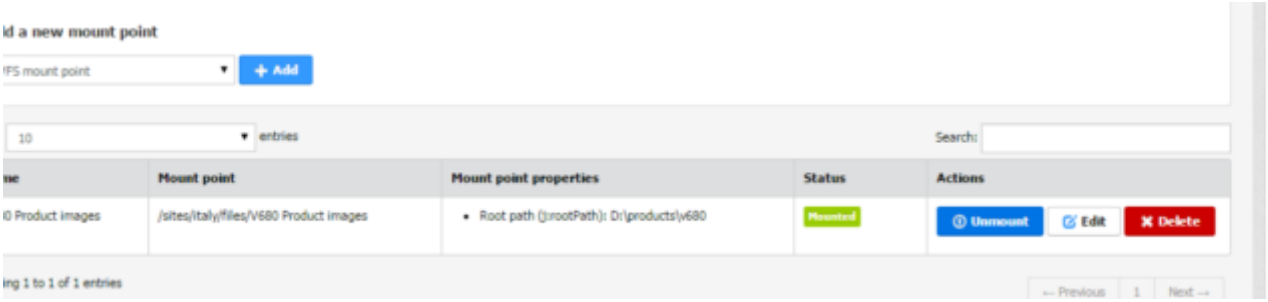
Click “save” to validate your setting.

Any user with the proper rights should then be able to open the “External repositories” tab and you should have a “Mounts” node with an “alfresco” sub node. If you click on this node, you should see the root Alfresco folders as illustrated below:



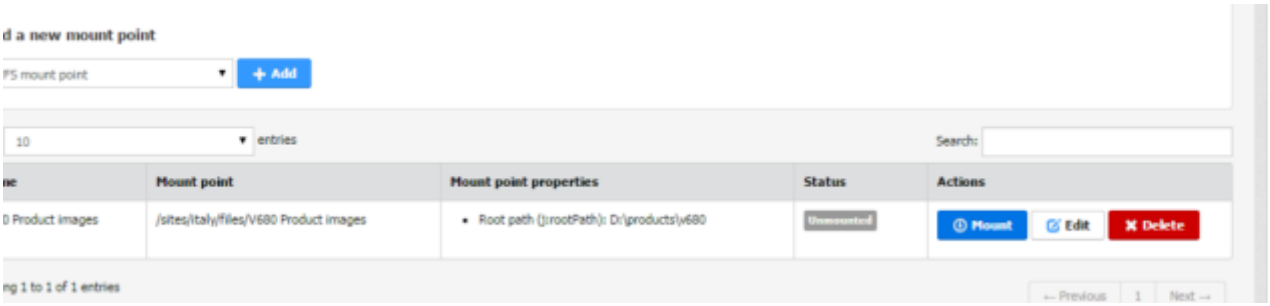
You can use the search to search inside the CMIS repository, but please note that the language search will not work since the external repositories have no notions of languages, so make sure you leave it empty when searching in CMIS repositories.

All information about the alfresco connection will appear in the matrix displayed on the main template of the mount points module.



If you delete the provider entry, the alfresco DAM won't be available for editors and will completely disappear from this list, you'll have to declare it once again if necessary.

When clicking on the Unmount button, the external data source is not available anymore but the provider declaration remains, so it can be reactivated in one click on the "Mount" button;



3.4 USER RIGHTS MANAGEMENT

The alfresco connector uses the current connected user on DX to log on Alfresco.

As Alfresco defines the rights on its resources it is not possible to change them from DX.

Even if the action is displayed to a user on DX side, if the user does not have rights on Alfresco, the operation will fail (see action menu in known issues).

Even if the user has write access to Alfresco content, in order to access managers, a user has to get the required edit role on a content of the site that set the needed permission to display it, like editor or contributor.

3.5 LIMITATIONS

3.5.1 Number of child nodes

When Alfresco contains a lot of files or folders under a folder, it takes time to display the tree: for each parent folder, the system needs to read the content of each of its children folders. To decrease the time to display, there are 2 options available and one suggestion/good practice:

- Use the Slow connection setting. With this option, the manager will not iterate other children's children when displayed. This option can be set from the administration / system components / mount points, in the settings of the mountpoint.
- Use the parameter org.jahia.cmis.max.child.nodes in jahia.properties that limits the number of returned child nodes. If the folder contains more items than the limit defined by this parameter, several files or folders might not be visible in the content trees, in manager or edit mode. It is always possible to query these contents, but some content might not be visible, as the number of returned items will be limited by this parameter.

Avoid to mount directly the /root folder (as it contains the /users directory which may contain lot of data and therefore might take long to load) and prefer a remote mount point like /Sites. Or create several mount points targeting the different directories your users will need to access specificaly.

3.5.2 Guest user

Alfresco does not allow guest user to use CMIS endpoint. To be able to browse content as “guest” on DX, you have to define the Alfresco user that will be used to browse Alfresco’s content as guest. This is done in the “Alfresco user name used to access public content” field. This user will be used only to browse public content. As guest is not allowed by Alfresco, it must be set to another user than guest.

3.5.3 Action menu

In DX edit mode or managers, actions allowed on Alfresco content should reflect the rights given to the current user.

There is an issue in the following context: if the mount point is set in a site, and a user that has no write access on Alfresco server but has a role that give write access set on the site node on DX, he will be able to execute all edit actions (delete / upload etc.) on Alfresco content, but the actions will fail, upload file will fail silently, other actions return an error to the UI.

3.5.4 Encoding of file names

DX parse and converts ASCII codes when saving a node name while Alfresco does not and just store the string literally.

Example:

If the file is named myFile%20.docx in Alfresco

It will be displayed as myFile%20.docx in DX File Manager when the file appears.

If the file is renamed directly through DX interfaces, the %20 will be encoded by DX and resolved as a white space.

Let's assume that an author renames the file myFile%20encoded.docx when saved the name will be saved in DX File manager as myFile encoded.docx but in Alfresco this will result into a non coherent name.

This behavior cannot be fixed right now to match at the same time the way both systems work.

Consequently, we strongly recommend to train the authors and let them know that they should not rename mounted files when the name contains a %something in it, to avoid any confusion in the Alfresco repository.

4 ADVANCED CONFIGURATION

The Digital Experience Manager CMIS external provider is fully configurable using a Spring configuration file provided in the CMIS module at the following location: src/main/resources/META-INF/spring/cmis-provider.xml using a special CmisConfiguration bean. The CmisConfiguration bean has two main properties:

- repositoryProperties - map of connection related configuration properties. By default, repositoryProperties are configured for usage of general jahia.properties
- typeMapping - list of CmisTypeMapping beans. There are 2 base mappings: for cmis:document and cmis:folder.

4.1 MAPPING CONFIGURATION

CMIS - JCR mapping is configurable using the Spring configuration file. For each JCR type there must be CmisTypeMapping bean. Each CmisTypeMapping may have list of property mappings. Each property may support 3 modes (“mode”) - rwc. R - read, W - write (update), C - property that will be set on creation. By default, each property has only read mode active.

4.1.1 Property mapping example

```
<bean class="org.jahia.modules.external.cmis.CmisTypeMapping" id="cmis_document"
    p:jcrName="cmis:document" p:cmisName="cmis:document">
     <property name="properties">
         <list >
             <bean p:cmisName="cmis:createdBy" p:jcrName="jcr:createdBy"
    class="org.jahia.modules.external.cmis.CmisPropertyMapping" />
             <bean p:cmisName="cmis:description" p:jcrName="cmis:description"
    class="org.jahia.modules.external.cmis.CmisPropertyMapping" p:mode="rwc" />
             <bean p:cmisName="cmis:contentStreamFileName"
    p:jcrName="cmis:contentStreamFileName"
    class="org.jahia.modules.external.cmis.CmisPropertyMapping" p:mode="rw"/>
         </list>
     </property>
</bean>
```

**This fragment is not real mapping example. We use it only for documentation purposes. **

In this fragment we map CMIS *cmis:createdBy* property on JCR *jcr:createdBy* property as read only. In the next line *cmis:description* is mapped to a JCR property with the same name in read, create and write modes. In the next line the *cmis:contentStreamFileName* property is mapped to a JCR property with the same name in read and write modes. For this property create mode is not set. it means you can't set a value for this property on node creation, but it may be modified later. If the create mode is set without write mode, it is possible to set property value on creation, but it may not be modified later.

CmisTypeMapping beans may be organized in trees to benefit from inheritance. Child beans will inherit all parent attribute mappings. Local attribute mappings will override inherited. Inheritance may be configured in two ways:

a) by using child properties with a list of embedded beans;
b) by using separate beans linked by parent property.

4.1.2 Example 1

```
<property name="typeMapping">
    <list>
    <bean class="org.jahia.modules.external.cmis.CmisTypeMapping" id="cmis_document"
   p:jcrName="cmis:document"  p:cmisName="cmis:document">
        <property name="children">
            <list>
                <bean class="org.jahia.modules.external.cmis.CmisTypeMapping"
   p:jcrName="cmis:image" p:cmisName="cmis:image">
                </bean>
            </list>
        </property>
    </bean>
    </list>
</property>
```

4.1.3 Example 2

```
<property name="typeMapping">
    <list>
    <bean class="org.jahia.modules.external.cmis.CmisTypeMapping" id="cmis_document"
   p:jcrName="cmis:document"  p:cmisName="cmis:document">
    </bean>
    <bean class="org.jahia.modules.external.cmis.CmisTypeMapping"
   p:jcrName="cmis:image" p:cmisName="cmis:image">
```

```
        <property name="parent" ref="cmis_document"/>
    </bean>
    </list>
</property>
```

The first example is more visual while the second one is more flexible. You may even inherit mappings even from other modules. **Don't forget to include then parent bean in the typeMapping list.**