Alfresco Documentation

Content Management Interoperability Services (CMIS)

CMIS (Content Management Interoperability Services) is a standard for improving interoperability between ECM systems. It specifies a domain model plus a set of services and protocol bindings for Web Services (SOAP) and AtomPub.

You can get more detail on what CMIS is from OASIS.

Documentation

Note: This page needs to be updated for Alfresco 4

The most current documentation on using CMIS in Alfresco is available at:

https://developer.alfresco.com/

and

http://docs.alfresco.com

There is also a book on the topic called CMIS and Apache Chemistry in Action.

History

The latest published CMIS 1.0 Committee Specification 01.

Keep up-to-date Cover Pages blog posts twitter press.

On May 3rd, 2010, Mary McRae announced CMIS as an approved OASIS Standard.

On April 16, 2010, OASIS initiated a Call For Vote (which closes April 30) to approve the CMIS Committee Specification as an OASIS standard.

On April 9, 2010, Alfresco released Alfresco Community 3.3 with full support for CMIS 1.0 CD07.

On March 12, 2010, the CMIS technical committee approved and submitted CMIS 1.0 Committee Draft CD07 to OASIS for consideration as an OASIS standard.

On January 28, 2010, CMIS entered its second Public Review, which closed on February 12, 2010.

On October 23, 2009, CMIS entered its first Public Review, which closed on December 22, 2009.

On August 18, 2009, Alfresco provided http://cmis.alfresco.com, a hosted version of its CMIS repository and TCK.

On October 06, 2008, OASIS issued a public call for participation in a new technical committee chartered to define specifications for use of Web services and Web 2.0 interfaces to enable information sharing across content management repositories from different vendors. The OASIS CMIS TC works "to standardize a Web services interface specification that will enable greater interoperability of Enterprise Content Management (ECM) systems. CMIS uses Web services and Web 2.0 interfaces to enable rich information to be shared across Internet protocols in vendor-neutral formats, among document systems, publishers and repositories, within one enterprise and between companies."

On October 06, 2008, Alfresco announced the availability of the first implementation of the draft CMIS Specification.

On September 10, 2008, Enterprise Content Management vendors EMC Corporation, IBM Corporation, and Microsoft Corporation announced the publication of Content Management Interoperability Services (CMIS), distributed as a ZIP archive with four prose documents and a collection of schemas, WSDLs, and XML instances. The CMIS objective is to "define a domain model and set of bindings, such as Web Service and RESTful AtomPub that can be used by applications to work with one or more Content Management repositories/systems".

Use Cases

NOTE: The following text is from the excellent Three Fundamental CMIS Use Cases post by Laurence Hart from his Word Of Pie blog *[wordofpie.com/]**.***

CMIS is ideally suited to the following use cases.

Repository-to-Repository (R2R)

This is where content repositories talk directly to each other.

- Managing Records centrally that are stored in other repositories. This is typically called Federated Records Management but is different than the Federated use-case.
- Publishing content from one repository to another. A common scenario is publishing content from a collaboration/ECM system directly to a WCM system for publication to the Intra/Internet.

Application-to-Repository (A2R)

This is where an application that uses content is plugged-into a content repository to handle all content services.

- SharePoint as a front-end. If implemented, SharePoint can become the front-end and any other repository can be the back-end. This would address the existing SharePoint scalability issues without impacting the user experience.
- Collaboration systems. All the new Enterprise 2.0 systems, and existing collaboration applications, could use a robust back-end to provide more features such as de-duplification and records management. This is the same as the SharePoint use-case, just generalized.
- Enterprise Software Applications. Be it BPM, CRM, or any number of applications used by companies, content is becoming a larger part of those systems. Having a central place to manage the content and apply consistent rules is becoming critical.
- Content Enabled Vertical Applications (CEVAs). CMIS can really make life easier for CEVA vendors. Let the domain experts build a solution for the industry and let them plug it into any content system to manage the content. This frees the CEVA vendor from having to worry about the content or maintaining so many interfaces. That means more R&D money for real features.
- Productivity applications. Why not link Word, Excel, Open Office, or any desktop application to a repository using CMIS? Sounds lika a good way to write a portable integration, similar to the ODMA days - *attributed to Twitter user @billtrippe*.

Federated Repositories

This is where an application talks to many different repositories while presenting a singular interface to the user.

- Federated Search. One search hitting multiple repositories.
- Federation instead of migration. Any application can interact with multiple repositories. Instead of migrating all content from an old repository immediately, just have the interface interact with the legacy repository(s) until they are migrated or out-of-date. The application can store new content in the latest repository by default.

CMIS Domain Model

Objects

Objects represent the entities that are in the repository. Each object has a type. There are 4 base types defined by CMIS, Documents, Folders, Relationships and Policies. Every object in the repository will be derived from one of these types. An object will be identified by an Object ID and will have a set of properties associated with it. The properties that an object has is defined by its Object Type. The CMIS specification does specify how new object types are created.

In addition to the metadata properties that define object types, there are some additional attributes that govern some of the behavior of objects within the repository they are listed below

- Versionable: Can be versioned (only document objects are fileable)
- Fileable: Can be filed in a folder (folder object must be fileable, relationship objects are not fileable).

- Queryable: Can be queried (is mapped to a virtual table). Only folder and document objects can be queryable.
- Controllable-Policy: Can have a policy applied to it. Policies are not controllable.
- Controllable-ACL: Can have an ACL applied to it. Policies are not controllable.

Properties

Properties are named values that are associated with each object type. Properties are of a specific type (date, integer, text etc...). Properties can be single valued or multi valued, required or optional. Some properties may be read only or only updatable at certain times. One point to note is that properties can have different names associated with them their display name, ID and query name may all be different.

Document Objects

Document objects represent the enties that we really come to the repository for, the content. Document objects (and only document objects) may have Content Streams (the actual file associated with the document). In some cases it makes sense to have document objects without content streams. Content streams exist only as part of a containing document object. The content stream will have a mimetype associated with it. In addition to a content stream, a document object may contain one or more renditions (alternate views of the content).

Documents objects are also the only objects that are versionable, or for which versions can be exposed via CMIS. Each version of a document object will have their own object ID. All versions of a document make up a Version Series and will share a Version Series ID.

Folder Objects

Folder objects are containers used to organize the document objects within the repository. With the obvious exception of the root folder, folder objects must have one and only one parent folder. A folder has a folder path that is automatically generated representing its place in the repository's hierarchy. A folder object may be defined in a way the limits what object types can it can contain (for example, an accounting related folder could be defined to only contain document objects of type invoice). A folder object may have renditions (for example a folder may have a thumbnail as a rendition representing what is in the folder).

Relationship Objects

Relationship objects define a non-invasive two way relationships between two objects (source and target) in the repository. Manipulating the relationships should not effect any changes to either the source or target objects. Relationship objects are optional for CMIS compliant repositories.

Policy Objects

Policy objects are optional repository specific objects that can be applied to controllable objects. The behavior of policies are not modeled by the CMIS specification. A single policy object may be applied to multiple controllable objects and a single controllable object may have multiple policies applied to it. In order to preserve referential integrity, a policy object can not be deleted if it is applied to one or more controllable objects.

Renditions

Renditions are alternate views of the content stream such as previews, PDF renditions and thumbnails. It is also possible to have a thumbnail rendition object without content streams (i.e. folders). Renditions attributes must include a Stream ID and a mimetype. Additional common attributes for rendition are length, title and kind. The only kind of rendition that the CMIS specification defines is a thumbnail. Thumbnail renditions should only include height and width as attributes. The repository may define its own rendition types in addition to thumbnails.

Renditions cannot be queried unless they have a Rendition Document ID, that allows them to be exposed as documents.

Access Control

Access control is used to specify who can do what with an object in the repository. If the repository supports access control then access control lists are applied to each object within the repository. Access control lists specify what types of access or permissions (read, write etc..) to an object are given to groups or users (known collectively as principles). CMIS defines three permissions cmis:read, cmis:write and cmis:all. When setting an ACL cmis:user can be used to represent the current authemticated user.

Change Log

The repository may have an optional change log that contains an entry for each change made to content in the repository. Each entry has a Change Log Token. The repository must expose the latest change log token if it support change logs. Change log entries include the object ID and the change type (created, updated, deleted or security). Armed with a change log token, a client could retrieve the list of objects that have been changed since the change was made.

A change log need not contain every change for the life of the repository, but it must contain every change made since the earliest change in the log.

Services

The following services are available to the client. Some of these services may provide optional functionality and therefore not be supported in all repositories.

Repository Services

- Get Repositories: Get a list of repositories that can be accessed from this service endpoint
- Get Repository Info: Get information about the specified repository
- Get Type Children, Get Type Descendants: Various ways to discover the object types in a repository
- Get Type Definition: Get the definition (list of properties) of the specified type

Navigation Services

- Get Folder Tree, Get Descendants, Get Children: Retrieve descendant objects (each one has slightly different nuances)
- Get Folder Parent, Get Object Parents: Retrieve an object's parent folder(s)
- Get Checkedout Docs: Retrieve list of checked out documents

Discovery Services

- Query: Execute a CMIS query
- Get Content Changes: Gets a list of changes to the repository; the client can provide an optional change log token that specifies the first event to be included in the list.

Object Services

- Get Object, Get Object By Path: Retrieve objects
- Get Properties, Get Allowable Actions, Get Renditions: Get information about objects
- Get Content Stream: Retrieve an object's content stream
- Create Relationship, Create Document, Create Document From Source, Create Policy, Create Folder: Create objects
- Update Properties, Move Object: Update objects
- Delete Object, Delete Tree: Remove objects
- Set Content Stream, Delete Content Stream: Update content streams

Versioning Services

- Get Properties Of Latest Version, Get Object Of Latest Version: Get information about latest version of object
- Get All Versions: Retrieve an object's version history
- Check Out, Check In, Cancel Check Out: Control locking/unlocking of an object for the purpose of updating
- Delete All Versions: Remove version history

Relationship Services

- Get Object Relationships: Get the relationships associated with an object

Multi-Filing Services

- Add Object To Folder, Remove Object From Folder: File and un-file objects;
    - If multi-filing is supported in the repository, then an object can be added to multiple folders
    - If un-filing is supported in the repository, then an object can be removed from all folders that it is filed in without deleting the object

ACL Services

- Get ACL: Get the permissions associated with an object
- Apply ACL: Set the permissions associated with an object

Policy Services

- Get Applied Policies: Get the policies that are applied to an object
- Apply Policy, Remove Policy: Apply and remove policies to/from an object

Alfresco CMIS Compliance

Alfresco provides a full implementation of the CMIS specification allowing access to its Content Repository via the CMIS AtomPub and Web Service bindings as defined by CMIS 1.0 specification CS01. This implementation is formally available in:

- Alfresco Community 3.3g (released Jun 2010)
- Alfresco Enterprise 3.3 (released Jun 2010)

Access to CMIS 1.0 CS01 in Alfresco 3.3 is also available from:

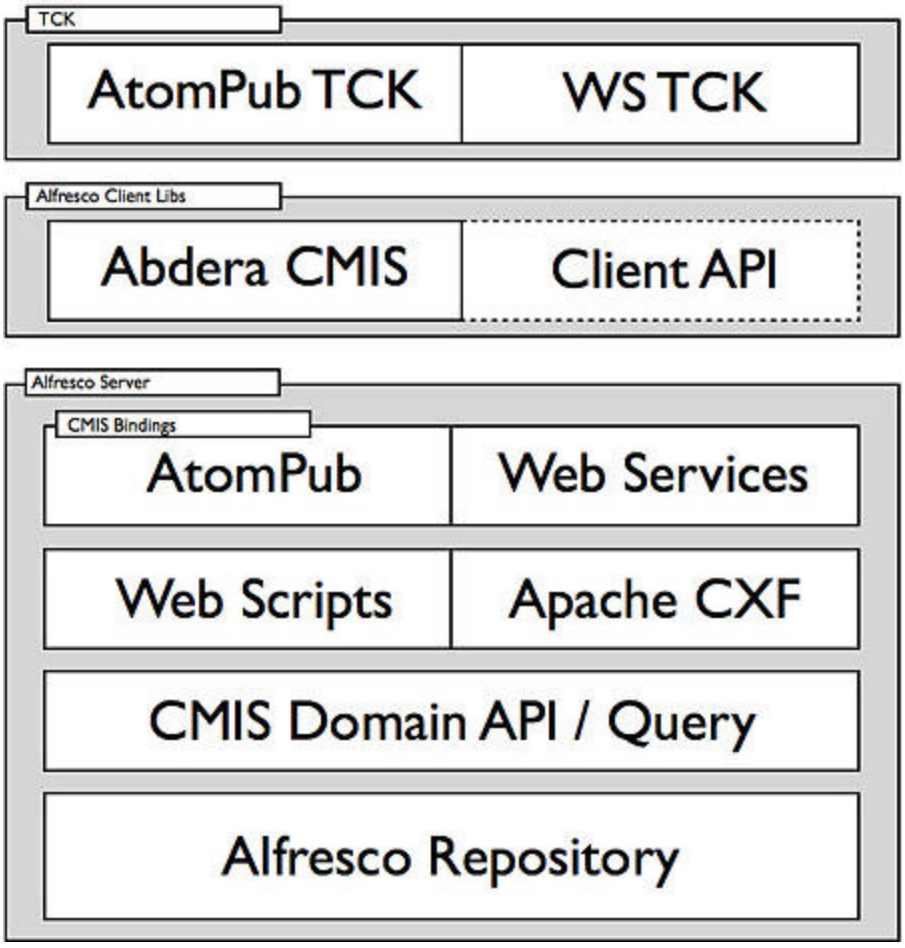
- http://cmis.alfresco.com : hosted CMIS Alfresco 3.3 Content Repository
- svn HEAD : source code for CMIS Alfresco 3.3

Since the introduction of CMIS in 2008, Alfresco has provided an open source implementation of the specification as it evolved. Currently released versions of Alfresco provide the following CMIS compliance:

- Community 3.3 : CMIS 1.0 CS01
- Community 3.2r2 : CMIS 1.0 CD04
- Community 3.2r : CMIS 0.61
- Community 3.2 : CMIS 0.61
- Enterprise 3.2 : CMIS 0.61 (unsupported)

Alfresco CMIS Content Repository

When Alfresco is installed and started, the following link **http:///[host]:[port]/alfresco/service/cmis/index.html** provides access to the Alfresco CMIS implementation, including reference documentation for the AtomPub binding, Web Service WSDLs and TCKs.



CMIS Service URL

The URL you should use to can gain access to the CMIS implementation depends on the Alfresco release, the binding, and the version of CMIS you want to use.

RESTful AtomPub Binding

This binding is fully described in the CMIS 1.0 CS01 specification.

To get started, the Alfresco AtomPub Service Document is available from any installed Alfresco Content Repository at:

CMIS 1.0

For Alfresco 3.x : http://[host]:[port]/alfresco/service/cmis

For Alfresco 4.0.x and Alfresco 4.1.x : http://[host]:[port]/alfresco/cmisatom

For Alfresco 4.2: http://[host]:[port]/alfresco/api/-default-/public/cmis/versions/1.0/atom

CMIS 1.1

For Alfresco 4.2: http://[host]:[port]/alfresco/api/-default-/public/cmis/versions/1.1/atom

Browser Binding

CMIS 1.1

For Alfresco 4.2 : **http://[host]:[port]/alfresco/api/-default-/public/cmis/versions/1.1/browser**

Web Services Binding

This binding is fully described in the CMIS 1.0 CS01 specification.

To get started, the Alfresco Web Service WSDL documents are available from any installed Alfresco Content Repository at:

For Alfresco 3.x : http://[host]:[port]/alfresco/cmis

For Alfresco 4.x : http://[host]:[port]/alfresco/cmisws

Query Language

All capabilities of the CMIS Query Language are supported (except join between Types). The Alfresco FTS language may be embedded in the CMIS QL contains() predicate.

To get started, you may also wish to follow the CMIS-SQL Tutorial.

Domain Model and Service Mapping to Alfresco

Alfresco's mapping of the domain model has been designed to allow the broadest access of Alfresco via CMIS.

Enabling the Change Log

Alfresco's implementation of the CMIS Change Log is an extension of the Auditing capabilities already available in Alfresco. By default, the CMIS Change Log is disabled. You'll see this reflected in the CMIS Repository capabilities where the value of capabilityChanges is none.

To enable the CMIS Change Log, the following configuration value must be set - either by setting it in alfresco-global.properties or JMX (if available):

audit.cmischangelog.enabled=true

Once set, the value of capabilityChanges becomes objectidsonly. This means the Alfresco Change Log provides access to which objects have changed (created, updated, deleted and permission modified), but does not provide a list of the properties which have changed.

Aspect Support

As a core feature of Alfresco, Aspects are exposed through the CMIS bindings. CMIS v1.0 does not define any support for Aspects, so Alfresco uses the CMIS extension mechanism to allow the query, read, and write of Aspects and their associated properties.

Aspect Domain Model

Each Aspect in Alfresco is mapped to a Policy Type definition. The mapping of Aspect to Type Definition is exactly the same as Class to Type Definition.

Although each Aspect is represented as a Policy, they are not "creatable", and therefore cannot be applied to objects via CMIS policy services.

The Aspects cm:referenceable, cm:auditable and cm:versionable are not mapped to Policies, as they are already mapped to native document/folder properties defined in the CMIS domain model.

Aspect Query

Alfresco Aspects are queried as if they are tables and joined to Types by ObjectId.

select d.*, o.* from cmis:document as d join cm:ownable as o on d.cmis:objectId = o.cmis:objectId

Aspect CRUD

Support for applying / removing Aspects and reading / writing of Aspect properties is provided through the CMIS extension mechanism and is available through the AtomPub and Web Service bindings.

The Alfresco Aspect extensions are represented by the following XSD:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" targetNamespace="http://www.alfresco.org"
  xmlns:atom="http://www.w3.org/2005/Atom" xmlns:xhtml="http://www.w3.org/1999/xhtml" xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
  xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc" jaxb:extensionBindingPrefixes="xjc" jaxb:version="2.1"
  xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/" version="1.0">


  <xs:import schemaLocation="CMIS-Core.xsd" namespace="http://docs.oasis-open.org/ns/cmis/core/200908/" />


  <xs:element name="setAspects">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="aspectsToAdd" type="xs:string" />
        <xs:element minOccurs="0" maxOccurs="unbounded" name="aspectsToRemove" type="xs:string" />
        <xs:element name="properties" type="cmis:cmisPropertiesType" minOccurs="0" />
      </xs:sequence>
```

```
    </xs:complexType>
  </xs:element>

  <xs:element name="aspects">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="appliedAspects"
type="xs:string" />
        <xs:element name="properties" type="cmis:cmisPropertiesType" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

All objects returned via the CMIS bindings are decorated with the '**aspects'** extension. Using the AtomPub binding as an example, the following 'getObject' call returns:

```
<cmisra:object>
  ...
 <alf:aspects>
   <alf:appliedAspects>P:cm:titled</alf:appliedAspects>
   <alf:properties>
     <cmis:propertyString propertyDefinitionId="cm:description" displayName="Description"
queryName="cm:description">
       <cmis:value>Summary</cmis:value>
     </cmis:propertyString>
     <cmis:propertyString propertyDefinitionId="cm:title" displayName="Title"
queryName="cm:title"/>
   </alf:properties>
 </alf:aspects>
 ...
</cmisra:object>
```

The previous sample demonstrates the retrieval of a folder with the cm:titled aspect applied. The Aspect name P:cm:titled and property names cm:description and cm:title are all described by the associated Policy Type Definition. Note, Aspect properties are represented using the same schema as all other CMIS properties, thus allowing access to display and query names.

All CMIS services which allow the creation and update of objects support the '**setAspects'** extension. Using the AtomPub binding as an example, the following may be added to create or update requests:

```
<cmisra:object>
  ...
 <alf:setAspects>
   <alf:aspectsToAdd>P:cm:titled</alf:aspectsToAdd>
   <alf:properties>
     <cmis:propertyString propertyDefinitionId="cm:description" displayName="Description" queryName="cm:description">
       <cmis:value>Summary</cmis:value>
     </cmis:propertyString>
   </alf:properties>
 </alf:setAspects>
 ...
</cmisra:object>
```

The previous sample demonstrates the application of the cm:titled aspect on creation of an object. Multiple aspects may be applied at once. For updates, aspects may also be removed. It is also possible to set Aspect properties without explicitly listing the Aspect in the **aspectToAdd** list. In this case, the Alfresco repository implicitly adds the Aspect.

Alfresco Share Data Lists

Alfresco Share uses the following folder structure to represent Data Lists:

/Sites/{sitename}/dataLists/{dataListContainer}/{dataListItem}

where:

- sitename => name of site
- dataListContainer => a folder (of type dl:dataList) which represents a data list of items
- dataListItem => a content item (derived from type dl:dataListItem) which represents a single data list item

Note: dataListContainer and dataListItem nodes are GUID named, so the cm:name property cannot be used for display purposes

For example, a todo data list in the Site 'cmisdl' containing two todo items is represented as follows:

/Sites/cmisdl/dataLists/1/1

/Sites/cmisdl/dataLists/1/2

Data Lists Access Via CMIS

a) Retrieve dataListContainer (e.g. todo data list container '1' in site 'cmisdl')

http://[host]:[port]/alfresco/s/cmis/p/Sites/cmisdl/dataLists/1

b) Retrieve dataListItem entries in dataListContainer (e.g. todo data list items in container '1' in site 'cmisdl')

http://[host]:[port]/alfresco/s/cmis/p/Sites/cmisdl/dataLists/1/children

c) Retrieve single dataListItem entry (e.g. todo data list item '2' in container '1' in site 'cmisdl')

http://[host]:[port]/alfresco/s/cmis/p/Sites/cmisdl/dataLists/1/2

Data Lists Model Mapping

1) The dl:dataList type is mapped to the CMIS type F:dl:dataList (a cmis:folder)

http://[host]:[port]/alfresco/s/cmis/type/F:dl:dataList

2) The base dl:dataListItem type is mapped to D:dl:dataListItem (a cmis:document)

http://[host]:[port]/alfresco/s/cmis/type/D:dl:dataListItem

3) The derived data list item types are mapped to a sub-type of D:dl:dataListItem e.g. todo data list item

http://[host]:[port]/alfresco/s/cmis/type/D:dl:todoList

Note: The previous CMIS atom feeds and entries contain 'describedby' rel links to the appropriate type descriptions

Data List Advanced Capabilities

Alfresco data list items can support arbitrary associations to other items in the repository, such as a document attachment or person assignment.

Access to these associations is possible via CMIS relationships support:

1) Get relationships of data list item '1' in container '1' in site 'cmisdl'

http://[host]:[port]/alfresco/s/cmis/p/Sites/cmisdl/dataLists/1/1/rels

Note: The associations are not mapped to data list item properties (this may an enhancement in a future Alfresco release).

Note: Only associations between items of type folder and document are exposed (as limited by CMIS domain model), so associations to people are not exposed.

Data List CRUD

Items may also be created, updated, or deleted using the standard CMIS services using the CMIS service URLs in the previous section.

Share and WCM Renditions

See the blog article Alfresco's Interpretation of CMIS Renditions [http://blogs.alfresco.com/wp/cmis/2010/05/17/alfrescos-interpretation-of-cmis-renditions/].

TODO:

Compression

Some AtomPub binding calls can generate large XML responses. These responses compress very well. Compressing CMIS AtomPub responses can increase the performance noticeably if the CMIS client supports it. (Apache Chemistry OpenCMIS supports compression since version 0.3.)

Please refer to the documentation of your servlet engine how to turn compression on.

The following MIME types should be compressed:

- application/atomsvc+xml
- application/atom+xml;type=entry
- application/atom+xml;type=feed
- application/cmisquery+xml
- application/cmisallowableactions+xml
- application/cmisatom+xml
- application/cmistree+xml
- application/cmisacl+xml

To turn on Tomcat compression, edit the file server.xml, which is located under the Tomcat conf directory. The compression configuration is added to the connector element:

```
<Connector port="8080" protocol="HTTP/1.1"
        connectionTimeout="20000"
        redirectPort="8443" URIEncoding="UTF-8"
        compression="on" compressionMinSize="2048"

compressableMimeType="text/html,text/xml,application/atomsvc+xml,application/atom+xml;type
=entry,application/atom+xml;type=feed,application/cmisquery+xml,application/cmisallowableacti
ons+xml,application/cmisatom+xml,application/cmistree+xml,application/cmisacl+xml" />
```

CMIS Client APIs

To use CMIS, you can either directly interact with one of the CMIS bindings or develop against a CMIS client API. Several APIs are "in development" for the following programming languages:

- Java (Apache Chemistry OpenCMIS, Apache Abdera Extension)
- Python
- PHP
- JavaScript

Getting Started

Jeff Potts has created an excellent tutorial using Alfresco, to take you through an overview of the specification as well as provide you some real-world examples such as using curl to make GET, PUT, POST, and DELETE calls against Alfresco to perform CRUD functions on folders, documents, and relationships in the repository.

Apache Chemistry OpenCMIS

OpenCMIS (part of Apache Chemistry) is a collection of Java libraries, frameworks, and tools around the CMIS specification.

The aim of OpenCMIS is to make CMIS simple for Java client and server developers. It hides the binding details and provides APIs and SPIs on different abstraction levels.

OpenCMIS is subdivided into these major areas:

CMIS Client

OpenCMIS provides two CMIS client APIs: the Client API and Provider API. The Client API is a high-level, object orientated API and suitable for most use cases. It sits on top of the Provider API. The Provider API reflects the CMIS domain model.

CMIS Server

The OpenCMIS Server Framework handles both CMIS bindings on the server side and maps them to a common set of Java interfaces. Repository vendors just need to implement those interfaces and don't need to worry about the protocol on the wire.

Authenticating with Alfresco ticket using OpenCMIS for Alfresco 3.4.x

As per https://issues.alfresco.com/jira/browse/ALF-7074: Ticket can be specified in password, if any of the following are true:

a) username is not specified (i.e. null, or length of zero)

b) username is equal to "ROLE_TICKET" (case insensitive)

The CMIS REST API has also been updated to support the above, so it's consistent with Web Services.

Spring Surf CMIS

For developing CMIS-based Web Applications, a prototype integration between Spring Surf and the OpenCMIS Client API exists. It is described in the following blog [blogs.alfresco.com/wp/cmis/2010/03/17/spring-surf-and-opencmis-integration/http://blogs.alfresco.com/wp/cmis/2010/03/17/spring-surf-and-opencmis-integration/].

Checkout the prototype integration source code if you wish to build and explore it.

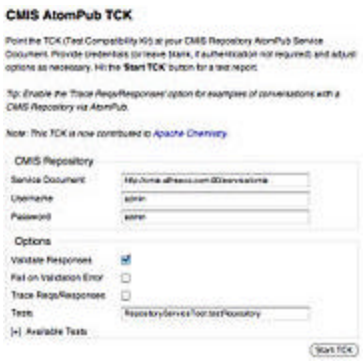
Apache Abdera CMIS Extension

Apache Abdera is an implementation of the Atom Syndication Format (RFC 4287) and Atom Publishing Protocol (RFC 5023) specifications. The CMIS RESTful AtomPub binding is an extension of the Atom Publishing Protocol. Therefore, Apache Abdera may be used as a client library to any CMIS AtomPub binding provider.

To support the custom CMIS schema, a CMIS extension for Apache Abdera is available from Apache Chemistry.

Test Compatibility Kits

AtomPub TCK



AtomPub TCK

The CMIS AtomPub TCK provides a series of tests that exercise a provider of the CMIS AtomPub binding. Tests are implemented as a series of JUnit test cases.

The Alfresco Content Repository has the AtomPub TCK built-in. It's accessible from:

http://[host]:[port]/alfresco/service/cmis/index.html

By default, the TCK tests the host Alfresco Content Repository, but may test any CMIS AtomPub binding provider.

The TCK supports the following options:

- the tests to execute
- the username / password to authenticate with
- the AtomPub service document url
- validate server responses against CMIS XSDs?
- trace requests and responses
- default document, folder and relationship types to create

The TCK has now been contributed to Apache Chemistry where its source code is now maintained.

Web Services TCK

The CMIS Web Services TCK provides a series of tests that exercise a provider of the CMIS Web Service binding. Tests are implemented as a series of JUnit test cases.

At this time, the Web Services TCK are tied to Alfresco.

Interoperability

Several initiatives are underway to implement both CMIS providers and clients.

Discussion of interoperability problems between CMIS clients or servers [groups.google.com/group/cmis-interop]

CMIS Providers

Vendor Repositories

- Alfresco Draft CMIS Implementation
- Hosted Alfresco Draft CMIS Implementation (username/password=demo/demo) Alfresco CMIS Demo Site
- EMC Documentum
- IBM FileNet P8 Content Manager 4.0
- sensenet
- Day Software (hosted)
- KnowledgeTree
- CMIS for SharePoint

CMIS Toolkits

Client APIs, Protocol Bindings, SPI, Test Repositories

- Apache Chemistry
- CMIS FileShare by Florian Mueller
- NCMIS (.NET)
- PHP CMIS Client API Module
- Python cmislib
- Flex CMIS Client API
- CMIS Spaces ActionScript (Flex+Browser, Flex+AIR) Client APIs for both CMIS Atom REST and CMIS Web Services
- Apache Chemistry AtomPub TCK Java CMIS Client
- A CMIS 1.0 Public Review Maven Archetype for simple CMIS app scaffolding using Chemistry AtomPub TCK CMISClient

CMIS Clients (User Interfaces, Tools)

User Interfaces, Tools

- CMIS Federated Search from AIIM iECM Demo
- Drupal CMIS Alfresco Module
- Shane Johnson's Flex/AIR CMIS Browser Source code Google Group
- Steve Reiner's CMIS Spaces Clients (Flex+AIR and Flex+Browser) on Google Code on Alfresco Forge Blog
- Sten Andersons JavaFX Client
- Jan Pfitzner's Ext GWT Client
- Integrating External Document Repositories with SharePoint Server 2007
- sensenet CMIS Client sensenet CMIS Aggregation
- CMIS iPhone client (alpha)
- CMIS Explorer (open source, WAR using Struts2)

Also see this other list CMIS clients [groups.google.com/group/cmis-interop].

PlugFests (Compatibility Testing Events)

- Redmond Aug 08
- Basel Apr 09 Report Day 1 live Day 2 live and results Day 1 report Day 2 report Interop Videos

Resources

If you still haven't had enough of CMIS...

- Alfresco Press Release

- Read what our CTO, John Newton has to say about CMIS
- Alfresco Labs 3 Final download
- Alfresco Draft CMIS Implementation Blog
- Alfresco Draft CMIS Implementation Forum
- CMIS Developer Toolbox
- CMIS Top Ten Questions.
- View the CMIS webinar recording on-demand
- "Getting Started with CMIS" tutorial on ECMarchitect.com
- Take the 1-minute CMIS Survey!
- EMC Corporation Press Release EMC Developer Network: CMIS Community EMC CMIS Webinar
- IBM's WebSite IBM CMIS Prototype for FileNet P8 Content Manager 4.0
- Microsoft Sharepoint Blog Announcement, Microsoft ECM Blog Announcement

See the original document for Associated URLs.

This page was last modified on 3 June 2014, at 16:15. This page has been accessed 568,948 times.