Getting Started with CMIS

Examples using Content Management Interoperability Services, Abdera, & Chemistry

Jeff Potts
November, 2009

What is CMIS?

The goal of the Content Management Interoperability Services (CMIS) specification is to provide a set of services for working with rich content repositories. It was founded by IBM, Microsoft, and EMC and then broadened in 2007 to include Alfresco, Open Text, SAP, and Oracle. Many others have since joined the Technical Committee (TC) at OASIS, which is the organization that owns the CMIS spec.

CMIS is not language specific, it does not dictate how a content repository works, and it does not seek to incorporate services that expose every feature of every content repository. Instead, the goal is to define a set of common functions for working with content repositories and services, both REST-style and SOAP, for executing those functions.

People that get excited about CMIS like to say it is as important to content management applications as the standardization of SQL was to database applications in the late 1980's and early 1990's. Whether or not that is certain remains to be seen, but there is definitely a lot of buzz around CMIS right now, client tools and integrations based on CMIS are springing up, and it appears to have broad support among major ECM players.

Having a standard set of services for working with a content repository is an obvious benefit to software vendors who need to rely on content services, and would like to give their customers a choice in how those services are provided. But its usefulness also extends to enterprises, especially when there are a variety of presentation technologies and/or repositories involved.

Most companies have more than one technology in play on the presentation tier. It may not be as varied as the graphic depicts, but it is safe to say that heterogeneity is the norm. In the worst case, your repository has a non-standard interface and its own non-standard presentation framework. If you want to use broader, more popular frameworks, you'll either spend the time and money to write your own wrapper services or you'll just throw up your hands and settle for the presentation tools the vendor provides (Hello, vendor lock-in!).

In the best case, your repository provides an open, services-based API (or a framework to provide your own such API), and each of your custom

applications can work with the repository using those services. It may be a one-off integration that no one else knows much about, but at least the front-end team is happy.

But most companies Optaros deals with have more than one repository. In that case, projects are often forced into a user interface technology decision based on where their content assets reside and leveraging assets across multiple repositories from the same front-end can be tough because each repository has its own proprietary interface.

CMIS helps with this problem by providing a common set of services that sit on top of your repositories. As long as your presentation tier can speak REST or SOAP, it can work with any of the CMIS-compliant repositories in your enterprise, regardless of the underlying implementation. Front-ends are no longer tied to back-ends. Any front-end can talk to any back-end, and they all do it in the same way. A Drupal developer using the Drupal CMIS module doesn't have to relearn anything when the repository is switched from Nuxeo to KnowledgeTree. Someone who knows how to make calls to a CMIS repository could jump from a Wicket project to a Django project and still be familiar with how to get the content out of Alfresco.

So when you hear people say, "CMIS is SQL for content management," that's what they mean.

CMIS can be used as part of any typical document management use case. The key phrase here is "document management"—providing services that sit on top of a file-centric repository is what CMIS is all about. If you are looking for non-relational persistence that has nothing to do with file-based content, keep on looking. There are better-suited non-relational projects out there (do a search for "NoSQL").

About the specification

As far as specifications go, the CMIS spec is actually quite readable. In its current form it is about 226 pages, but a lot of that is simply API documentation, which you can skim over until you really need it (or perhaps use as a sleep aid). If you have a content management background, the concepts will be immediately familiar.

If you plan to do any work whatsoever with CMIS, you really need to have the specification document handy. The home page of the CMIS Technical Committee is: http://www.oasis-open.org/committees/cmis. The specification document can be downloaded as a PDF from http://docs.oasis-open.org/cmis/CMIS/v1.0/cd04/cmis-spec-v1.0.pdf.

Before diving in to a set of examples, it makes sense to spend some time becoming familiar with the spec. CMIS defines:

- A domain model
- A query language
- Protocol bindings
- A standard set of services

Let's look at each of these at a high level.

Domain Model

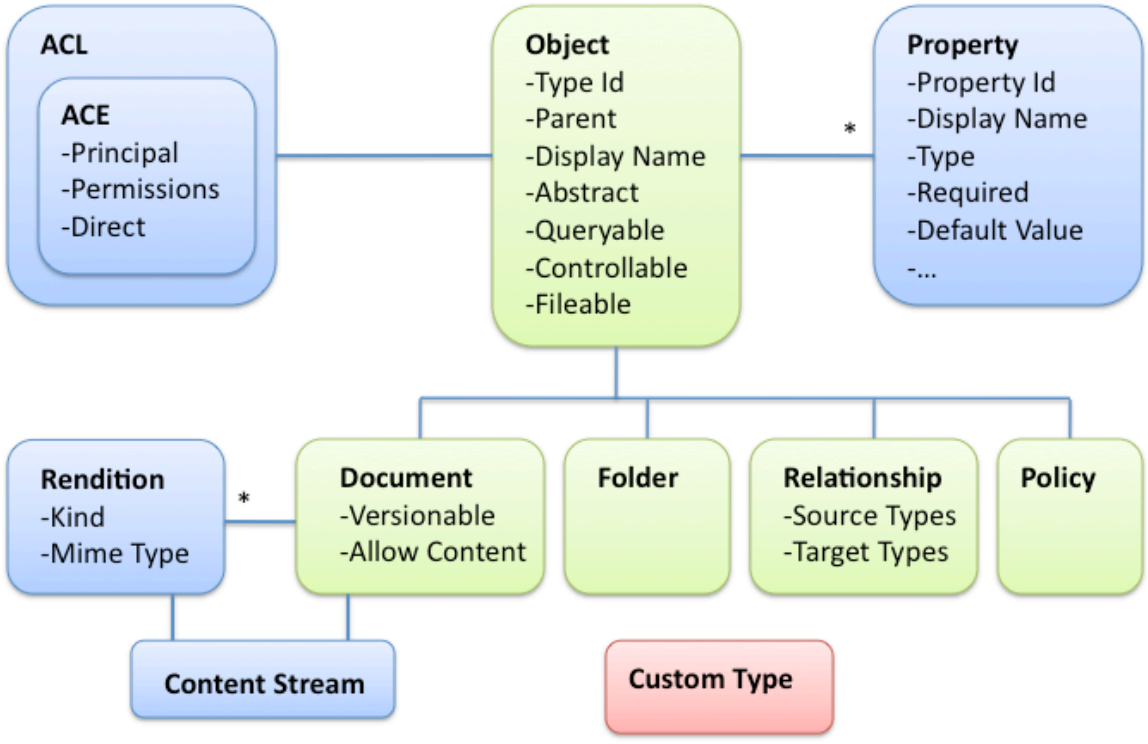
The Domain Model defines the concepts and relationships of CMIS repositories. It provides a common vocabulary for all of us to use so that when we're standing around at a cocktail party talking about CMIS everyone is speaking the same language. Let's go through some of the common terms.

A **Repository** is a container of objects. The container has a set of "capabilities" which may be different depending on implementation. A repository might not implement certain optional capabilities, but is still considered CMIS-compliant.

Objects are the entities a repository manages. All objects have an **Object Type**. An object type is a classification given to an object that declares all objects of that classification to have some set of properties in common. The type structure is hierarchical so child types inherit the properties of their parent.

There are only four base types, and they are probably the same four you'd list if I asked you to make them up on the spot: **Document, Folder, Relationship,** and **Policy**. Okay, so Policy may not have rolled off your tongue but you were probably thinking "Permission" which isn't really the same thing at all, but worth partial credit.

The graphic below shows the CMIS meta model:



As the graphic shows, all four objects have a shared set of attributes such as the object's type ID, its display name, whether or not it is query-able, whether or not it is controllable by a policy, etc. Documents have additional

attributes such as whether or not they are version-able and whether or not they are allowed to have a content stream (a file).

Objects have properties, and each property has a set of attributes such as the property ID, its display name, its query name, whether or not it is required, etc.

Custom types can extend the CMIS types. Exactly how those custom types are defined is up to the underlying repository.

Document Objects and **Folder Objects** are self-explanatory. **Relationship Objects** are used to create a graph between objects. Relationships have a source and a target. They relate two completely independent objects—creating or deleting a relationship is not supposed to affect the objects on either side of the relationship.

Document Objects are the only object type that can be **versioned**. Each version of a Document is its own first-class object and will have its own unique ID. Multiple versions of the same object will share a series ID.

A **Policy Object** is an arbitrary string that can be "applied" to an object. The specification purposefully leaves this vague and does not enforce any behavior based on whether or not a policy is applied to an object. That's completely up to the repository and the application. CMIS does allow object types to declare whether or not they can have a policy applied to them. If they can, they are said to be "controllable".

A **Content Stream** is a binary stream of data with a mime type, otherwise known as a file. A Document Object does not necessarily have to have a content stream. Folders cannot have a content stream. Repositories may have certain restrictions on how (or if) content streams get updated. You'll see more on that during the check-out/check-in example a little later.

CMIS does support **Renditions**, which are alternate forms of the content stream. The most common example is a thumbnail but there is no limit to the kind or number of renditions supported. You cannot create or update a rendition directly through CMIS—that's up to the repository. CMIS simply lets you retrieve the renditions.

An **Access Control List** is a list of principles, such as people, groups, or roles, mapped to permissions. CMIS starts with a basic set of permissions (read, write, all), which can be extended. CMIS repositories differ in how (or if) ACLs can be retrieved and managed through CMIS services.

Now you're armed and dangerous. This discussion may have instilled some nagging questions in your soul about things like referential integrity related to relationship objects, or what CMIS knows about a given property of a given type, or who knows what else. My goal wasn't to re-print the spec here--go read it to get the gory details.

Query Language

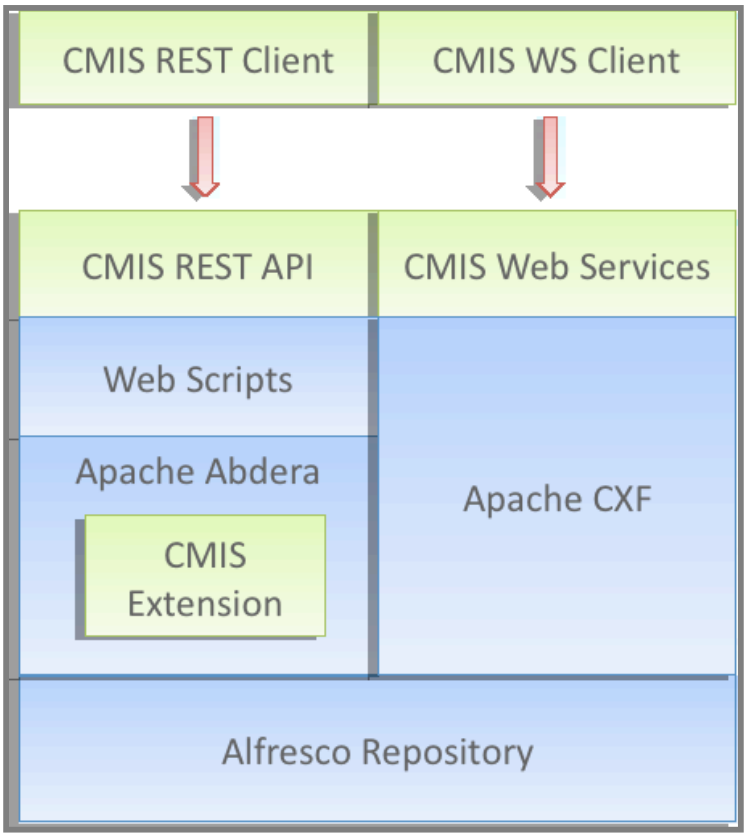
CMIS specifies a **query** language based on SQL-92. It is read-only and has limited support for joins. Depending on the repository, it can do both full-text

and metadata-based queries. You'll see several different CMIS queries in the examples section.

Protocol Bindings

To be fully CMIS compliant, a repository must provide the two bindings required by the spec: RESTful AtomPub and Web Services. (As an aside, please don't shoot the messenger regarding what is or isn't true REST, RESTful, or REST-like. I'm going to use the language used by the spec and if you're bothered by it please do take it up with the OASIS TC).

How each repository implements the bindings is specific to the repository. The graphic shows how Alfresco does it at a high level:



Using the Restful AtomPub Binding, a client will start out with the URL of the Repository Information page (also known as the "getRepositoryInfo" service). That's the main entry point into the repository. The client can inspect the links on the page to understand how to make additional GET, POST, PUT, and DELETE requests to the repository.

Responses from requests made via the AtomPub Binding are almost always in the form of an Atom XML feed or an Atom XML entry. Usually the response has been extended to include CMIS-specific tags as well. You can use Atom-specific libraries to parse the results (Apache Abdera is an example), generic XML parsers, or something you write yourself. Posts are often Atom as well. You'll see plenty of examples of this later on—all of the examples in this document use the Restful AtomPub Binding.

The Web Services Binding uses SOAP. There are two XSD documents that make up the WSDL for the services: one defines the data model and the other the message formats. The services available in the Web Services Binding are an exact match with the list of services provided in the next section.

Services

Regardless of the binding, there is a set of services that a CMIS repository must provide. These services include:

- **Repository Services**: Used to discover information and capabilities of the repository.
- **Navigation Services**: Used to traverse the repository's folder hierarchy.
- **Object Services**: Used to perform CRUD functions on objects.
- **Multi-filing Services**: If the repository supports putting an object in more than one folder, this service handles it.
- **Discovery Services**: Used to handle queries.
- **Versioning Services**: Used to checkout documents and work with document versions.
- **Relationship Services**: Used to query an object for its relationships.
- **Policy Services**: Used to apply, remove, and query for policies.
- **ACL Services**: Not supported by all repositories, this service is used to return and manage the ACL of an object.

Some of these services share common characteristics. For example:

- Almost every service that returns a list can be returned as a paged result set.
- Services that return objects can often return additional information about those objects, if you ask for it, like relationships, ACLs, renditions, and allowable actions.
- A repository may choose to give you a change token with an object to reduce the chance that you don't update an out-of-date object. If a repository gives you a change token, it expects it back when you post the change.
- Any of the services may throw an exception. For a list, check the spec.

Setting up a CMIS repository to play with

If you're going to work through the examples in this document you're going to need a CMIS repository. Here are some of the open source CMIS repository options:

Alfresco. Alfresco is an ECM platform that was first-to-market with a CMIS implementation in both the Enterprise and Community versions of their product. Of course CMIS has changed rapidly so the CMIS version supported varies across releases of the product. There are several options for trying out CMIS with Alfresco:

- Use Alfresco's hosted CMIS repository (http://cmis.alfresco.com)
- Download Alfresco Community (http://wiki.alfresco.com/wiki/Download_Community_Edition)

- Run an Alfresco Amazon EC2 Machine image (http://wiki.alfresco.com/wiki/EC2)
- Build from source (svn://svn.alfresco.com/alfresco/HEAD/root)

All of these support CMIS 1.0 Committee Draft 04, with the possible exception of the EC2 image, but it should be updated soon.

Apache Chemistry. Chemistry is an Apache incubator project with the goal of being a reference implementation for CMIS. Chemistry exposes any JCR repository as a CMIS repository. You can check out Chemistry from source, build it with Apache Maven, and have a simple CMIS 1.0 repository running in a matter of minutes. The project's home page is http://incubator.apache.org/chemistry/. You can find links to the Subversion repository from there.

KnowledgeTree. KnowledgeTree is an open source document management system with "experimental/unfinished" support for CMIS. According to their wiki, they are currently supporting up to 0.61c of the CMIS specification. Find out more at http://wiki.knowledgetree.com/CMIS.

Nuxeo. Nuxeo is an open source ECM platform that depends on Apache Chemistry for its CMIS support. In fact, some of the initial Chemistry code comes from Nuxeo—they collaborated with Day Software to start the project. According to Nuxeo's wiki page, Nuxeo currently supports 0.62, only the AtomPub Binding, and does not yet support updates. Find out more at http://www.nuxeo.org/xwiki/bin/view/Main/CMIS/.

Things are moving fast around CMIS. The level of support offered by these projects and the ease with which you can try it out will almost certainly have changed by the time you read this. The nice thing is that you've got a lot of options, so pick one and jump in.

For this tutorial I've chosen to use Alfresco Community built from head (rev 17478). The latest Alfresco Community download, Alfresco Community 3.2r2, includes support for CMIS 1.0cd04, so you don't have to build from source if you don't want to.

If you are using anything less than CMIS 1.0cd04 realize that there are significant changes in CMIS, not the least of which are namespace and URL differences, so copying and pasting the code examples from this paper into your environment is not likely to work if it does not support CMIS 1.0cd04.

Examples: Making HTTP calls against the CMIS repository

Okay, okay, enough theory. Time to get your hands dirty with some examples. The examples in this section show how to perform basic repository operations such as:

- Interrogating the CMIS repository's capabilities;
- Creating, reading, updating, and deleting folders, documents, and relationships;
- Executing CMIS queries; and
- Determining the "allowable actions" for an object.

The examples used in this document will be based entirely on the AtomPub binding.

Setup

I'm going to use curl (http://curl.haxx.se/) to execute GET's, POST's, PUT's, and DELETE's against the CMIS URLs in Alfresco. This is the best way to see exactly how requests and responses are formatted. In a real application, it's likely that a higher-level client API would be used for convenience. That's covered in a later section.

If you are running Ubuntu or Mac OS X, it's easy to get curl through apt-get or Macports. If you are running Windows, Cygwin (http://www.cygwin.com/) is probably your best bet. Of course, nothing in these examples requires curl—anything that can execute GET's, POST's, PUT's, and DELETE's will be fine.

As mentioned earlier, I'm using Alfresco Community locally. If you want to follow along, but you don't want to install your own CMIS repository, Alfresco has made a CMIS 1.0 repository available at http://cmis.alfresco.com. The username/password is admin/admin. If you use it, you might adjust your folder names slightly to avoid collisions with others who do the same.

Authenticating

CMIS doesn't actually dictate how authentication happens other than to say it is up to the underlying protocol to handle the authentication. In Alfresco, you have a couple of options: pass a ticket as an argument or use basic authentication.

Option 1: Pass a ticket

You can do a GET against the login URL (non-CMIS) and Alfresco will return a ticket. You'll need to include the ticket in subsequent calls against the server.

Doing this:

```
curl "http://localhost:8080/alfresco/s/api/login?u=admin&pw=admin"
```

Returns this:

```
<?xml version="1.0" encoding="UTF-8"?>
<ticket>TICKET_44740de7943f7bab5be61f787c62660ff3008299</ticket>
```

If you go this route, you'll want to copy and paste the ticket (everything between the open and closing ticket tags) somewhere convenient. Every call you make from here on out will require it, like this:

```
curl
"http://localhost:8080/alfresco/s/cmis?alf_ticket=TICKET_44740de7943f7b
ab5be61f787c62660ff3008299"
```

If you have to restart your repository, or you want to authenticate as a different user, you'll have to get a new ticket.

Option 2: Basic Authentication

The second option is to take advantage of HTTP Basic Authentication and curl knows how to do that. To use this option, just pass the username and password as an argument to curl, like this:

```
curl -uadmin:admin "http://localhost:8080/alfresco/s/cmis"
```

Note that if you don't want your password showing up in your key history, leave it off and curl will prompt you.

I don't want to fool with the ticket so I'm going to use Option 2.

Getting Basic Repository Information

The entry point into the CMIS repository is the "getRepositoryInfo" service. The service returns the repository information page. In Alfresco, the URL for this page is "/alfresco/s/cmis". (Those already familiar with Alfresco will recognize "/alfresco/s" as the short form of the base URL for web scripts). It's important to note that the URL's that make up a CMIS repository's REST API are not dictated by the CMIS specification. This entry point, then, serves as a kind of Shopping Mall Directory that lets you know where everything is. Theoretically, regardless of the underlying implementation, this URL is all you need to know in order to interact with the repository using CMIS.

Let's invoke the repository information URL for Alfresco and see what the response looks like.

Doing this:

```
curl -uadmin:admin "http://localhost:8080/alfresco/s/cmis"
```

Returns this:

```
<?xml version="1.0" encoding="utf-8"?>
<service xmlns="http://www.w3.org/2007/app"
xmlns:atom="http://www.w3.org/2005/Atom"
xmlns:cmisra="http://docs.oasis-open.org/ns/cmis/restatom/200908/"
xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/"
xmlns:alf="http://www.alfresco.org">
  <workspace>
    <atom:title>Main Repository</atom:title>

    <collection
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/4
eb6a431-3c56-4767-816a-4ceca2295ae2/children">
      <atom:title>root collection</atom:title>
      <cmisra:collectionType>root</cmisra:collectionType>
    </collection>
    <collection href="http://localhost:8080/alfresco/s/cmis/types">
      <atom:title>type collection</atom:title>
      <cmisra:collectionType>types</cmisra:collectionType>
    </collection>
```

```
    <collection
href="http://localhost:8080/alfresco/s/cmis/checkedout">
      <atom:title>checkedout collection</atom:title>
      <accept>application/atom+xml;type=entry</accept>
      <cmisra:collectionType>checkedout</cmisra:collectionType>
    </collection>
    <collection href="http://localhost:8080/alfresco/s/cmis/unfiled">
      <atom:title>unfiled collection</atom:title>
      <accept>application/atom+xml;type=entry</accept>
      <cmisra:collectionType>unfiled</cmisra:collectionType>
    </collection>
    <collection href="http://localhost:8080/alfresco/s/cmis/queries">
      <atom:title>query collection</atom:title>
      <accept>application/cmisquery+xml</accept>
      <cmisra:collectionType>query</cmisra:collectionType>
    </collection>

    <atom:link title="root folder tree" type="application/cmistree+xml"
rel="http://docs.oasis-open.org/ns/cmis/link/200908/foldertree"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/4
eb6a431-3c56-4767-816a-4ceca2295ae2/tree"/>
    <atom:link title="root descendants" type="application/cmistree+xml"
rel="http://docs.oasis-open.org/ns/cmis/link/200908/rootdescendants"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/4
eb6a431-3c56-4767-816a-4ceca2295ae2/descendants"/>
    <atom:link title="type descendants" type="application/cmistree+xml"
rel="http://docs.oasis-open.org/ns/cmis/link/200908/typesdescendants"
href="http://localhost:8080/alfresco/s/cmis/types/descendants"/>

    <cmisra:repositoryInfo>
      <cmis:repositoryId>af0e3a5a-628f-41a5-8a82-
75006933331c</cmis:repositoryId>
      <cmis:repositoryName>Main Repository</cmis:repositoryName>
      <cmis:repositoryDescription></cmis:repositoryDescription>
      <cmis:vendorName>Alfresco</cmis:vendorName>
      <cmis:productName>Alfresco Repository
(Community)</cmis:productName>
      <cmis:productVersion>3.2.0 (r2 @build-
number@)</cmis:productVersion>
      <cmis:rootFolderId>workspace://SpacesStore/4eb6a431-3c56-4767-
816a-4ceca2295ae2</cmis:rootFolderId>
      <cmis:capabilities>
        <cmis:capabilityACL>none</cmis:capabilityACL>

<cmis:capabilityAllVersionsSearchable>false</cmis:capabilityAllVersions
Searchable>
        <cmis:capabilityChanges>none</cmis:capabilityChanges>

<cmis:capabilityContentStreamUpdatability>anytime</cmis:capabilityConte
ntStreamUpdatability>

<cmis:capabilityGetDescendants>true</cmis:capabilityGetDescendants>

<cmis:capabilityGetFolderTree>true</cmis:capabilityGetFolderTree>
        <cmis:capabilityMultifiling>true</cmis:capabilityMultifiling>

<cmis:capabilityPWCSearchable>true</cmis:capabilityPWCSearchable>
        <cmis:capabilityPWCUpdatable>true</cmis:capabilityPWCUpdatable>
        <cmis:capabilityQuery>bothcombined</cmis:capabilityQuery>
```

```
        <cmis:capabilityRenditions>none</cmis:capabilityRenditions>
        <cmis:capabilityUnfiling>false</cmis:capabilityUnfiling>

<cmis:capabilityVersionSpecificFiling>false</cmis:capabilityVersionSpec
ificFiling>
        <cmis:capabilityJoin>none</cmis:capabilityJoin>
      </cmis:capabilities>
      <cmis:cmisVersionSupported>1.0</cmis:cmisVersionSupported>
      <alf:cmisSpecificationTitle>Version 1.0 Committee Draft
04</alf:cmisSpecificationTitle>
    </cmisra:repositoryInfo>
…SNIP…
```

You should refer to the CMIS specification to find out what all of the elements in the response mean. For now, there are a couple of different areas I want to direct your attention to:

- The list of `collection` elements provides a set of URLs that can be drilled down into. The most common collection is the root collection, and you'll learn more on that in a minute, but there are also collections for checked out documents, types, and others.
- The `cmisra:repositoryInfo` element contains useful child elements such as the repository vendor, product, and version, as well as the version of CMIS the repository supports.
- The `cmis:capabilities` element contains child elements that describe which pieces of the CMIS spec are supported and to what degree. If you are writing a client with the goal of true cross-repository portability, you'll want to pay close attention to these. Again, see the specification for a full description of each of the capabilities listed.
- The repository information response can also help you figure out the URL format your CMIS repository expects. I didn't include them here, but there is a set of `cmisra:uritemplate` elements that provide key URL patterns.

Displaying the contents of the Root Directory

In the response from the main entry point, notice the following collection element:

```
<collection
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/4
eb6a431-3c56-4767-816a-4ceca2295ae2/children">
  <atom:title>root collection</atom:title>
  <cmisra:collectionType>root</cmisra:collectionType>
</collection>
```

The title is "root collection" which means invoking that URL will give us a feed of the contents of the root directory, which for Alfresco is Company Home. So, if you want to see a list of the objects in the root directory of the CMIS repository, you simply do a GET against the root collection URL.

Doing this:

```
curl -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/4eb6a4
31-3c56-4767-816a-4ceca2295ae2/children"
```

Returns this:

```
<feed xmlns="http://www.w3.org/2005/Atom"
xmlns:app="http://www.w3.org/2007/app"
xmlns:cmisra="http://docs.oasis-open.org/ns/cmis/restatom/200908/"
xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/"
xmlns:alf="http://www.alfresco.org"
xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/">
  <author>
    <name>System</name>
  </author>
  <generator version="3.2.0 (r2 @build-number@)">Alfresco
(Community)</generator>

<icon>http://localhost:8080/alfresco/images/logo/AlfrescoLogo16.ico</ic
on>
  <id>urn:uuid:4eb6a431-3c56-4767-816a-4ceca2295ae2-children</id>
  <link rel="service" href="http://localhost:8080/alfresco/s/cmis" />
  <link rel="self"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/4
eb6a431-3c56-4767-816a-4ceca2295ae2/children" />
  <link rel="via"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/4
eb6a431-3c56-4767-816a-4ceca2295ae2" />
  <link rel="down"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/4
eb6a431-3c56-4767-816a-4ceca2295ae2/descendants"
type="application/cmistree+xml" />
  <link rel="http://docs.oasis-open.org/ns/cmis/link/200908/foldertree"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/4
eb6a431-3c56-4767-816a-4ceca2295ae2/tree"
  type="application/cmistree+xml" />
  <link rel="first"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/4
eb6a431-3c56-4767-816a-4ceca2295ae2/children?pageNo=1&amp;pageSize=-
1&amp;guest=&amp;format=atomfeed"
  type="application/atom+xml;type=feed" />
  <link rel="last"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/4
eb6a431-3c56-4767-816a-4ceca2295ae2/children?pageNo=1&amp;pageSize=-
1&amp;guest=&amp;format=atomfeed"
  type="application/atom+xml;type=feed" />
```

```
  <title>Company Home Children</title>
  <updated>2009-11-13T13:14:29.863-06:00</updated>
  <opensearch:totalResults>11</opensearch:totalResults>
  <opensearch:startIndex>0</opensearch:startIndex>
  <opensearch:itemsPerPage>-1</opensearch:itemsPerPage>
  <cmisra:numItems>11</cmisra:numItems>
  <entry>
    <author>
      <name>System</name>
    </author>
    <content>8cd47010-0465-4e94-8a9d-e34417e63591</content>
    <id>urn:uuid:8cd47010-0465-4e94-8a9d-e34417e63591</id>
    <link rel="self"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/8
cd47010-0465-4e94-8a9d-e34417e63591" />
    <link rel="edit"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/8
cd47010-0465-4e94-8a9d-e34417e63591" />
    <link rel="http://docs.oasis-
open.org/ns/cmis/link/200908/allowableactions"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/8
cd47010-0465-4e94-8a9d-e34417e63591/allowableactions" />
    <link rel="http://docs.oasis-
open.org/ns/cmis/link/200908/relationships"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/8
cd47010-0465-4e94-8a9d-e34417e63591/rels" />
    <link rel="up"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/4
eb6a431-3c56-4767-816a-4ceca2295ae2"
    type="application/atom+xml;type=entry" />
    <link rel="down"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/8
cd47010-0465-4e94-8a9d-e34417e63591/children"
    type="application/atom+xml;type=feed" />
    <link rel="down"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/8
cd47010-0465-4e94-8a9d-e34417e63591/descendants"
    type="application/cmistree+xml" />
    <link rel="http://docs.oasis-
open.org/ns/cmis/link/200908/foldertree"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/8
cd47010-0465-4e94-8a9d-e34417e63591/tree"
    type="application/cmistree+xml" />
    <link rel="describedby"
    href="http://localhost:8080/alfresco/s/cmis/type/F:st:sites" />
    <link rel="service"
    href="http://localhost:8080/alfresco/s/cmis" />
    <published>2009-11-13T13:14:40.843-06:00</published>
    <summary>Site Collaboration Spaces</summary>
    <title>Sites</title>
    <updated>2009-11-13T13:14:40.869-06:00</updated>
```

```
    <app:edited>2009-11-13T13:14:40.869-06:00</app:edited>
    <alf:icon>http://localhost:8080/alfresco/images/icons/space-icon-
default-16.gif</alf:icon>
    <cmisra:object>
      <cmis:properties>
        <cmis:propertyId
propertyDefinitionId="cmis:allowedChildObjectTypeIds" />
        <cmis:propertyId propertyDefinitionId="cmis:objectTypeId">
          <cmis:value>F:st:sites</cmis:value>
        </cmis:propertyId>
        <cmis:propertyString
propertyDefinitionId="cmis:lastModifiedBy">
          <cmis:value>System</cmis:value>
        </cmis:propertyString>
        <cmis:propertyString propertyDefinitionId="cmis:path">
          <cmis:value>/Sites</cmis:value>
        </cmis:propertyString>
        <cmis:propertyString propertyDefinitionId="cmis:name">
          <cmis:value>Sites</cmis:value>
        </cmis:propertyString>
        <cmis:propertyString propertyDefinitionId="cmis:createdBy">
          <cmis:value>System</cmis:value>
        </cmis:propertyString>
        <cmis:propertyId propertyDefinitionId="cmis:objectId">
          <cmis:value>
          workspace://SpacesStore/8cd47010-0465-4e94-8a9d-
e34417e63591</cmis:value>
        </cmis:propertyId>
        <cmis:propertyDateTime
propertyDefinitionId="cmis:creationDate">
          <cmis:value>2009-11-13T13:14:40.843-06:00</cmis:value>
        </cmis:propertyDateTime>
        <cmis:propertyString propertyDefinitionId="cmis:changeToken" />
        <cmis:propertyId propertyDefinitionId="cmis:baseTypeId">
          <cmis:value>cmis:folder</cmis:value>
        </cmis:propertyId>
        <cmis:propertyDateTime
propertyDefinitionId="cmis:lastModificationDate">
          <cmis:value>2009-11-13T13:14:40.869-06:00</cmis:value>
        </cmis:propertyDateTime>
        <cmis:propertyId propertyDefinitionId="cmis:parentId">
          <cmis:value>workspace://SpacesStore/4eb6a431-3c56-4767-816a-
4ceca2295ae2</cmis:value>
        </cmis:propertyId>
      </cmis:properties>
    </cmisra:object>
    <cmisra:pathSegment>Sites</cmisra:pathSegment>
  </entry>
  <entry>
    <author>
      <name>System</name>
    </author>
    <content>871feb4c-240a-4475-a678-e2d86911abee</content>
    <id>urn:uuid:871feb4c-240a-4475-a678-e2d86911abee</id>
    <link rel="self"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/8
71feb4c-240a-4475-a678-e2d86911abee" />
    <link rel="edit"
```

```
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/8
71feb4c-240a-4475-a678-e2d86911abee" />
    <link rel="http://docs.oasis-
open.org/ns/cmis/link/200908/allowableactions"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/8
71feb4c-240a-4475-a678-e2d86911abee/allowableactions" />
    <link rel="http://docs.oasis-
open.org/ns/cmis/link/200908/relationships"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/8
71feb4c-240a-4475-a678-e2d86911abee/rels" />
    <link rel="up"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/4
eb6a431-3c56-4767-816a-4ceca2295ae2"
    type="application/atom+xml;type=entry" />
    <link rel="down"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/8
71feb4c-240a-4475-a678-e2d86911abee/children"
    type="application/atom+xml;type=feed" />
    <link rel="down"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/8
71feb4c-240a-4475-a678-e2d86911abee/descendants"
    type="application/cmistree+xml" />
    <link rel="http://docs.oasis-
open.org/ns/cmis/link/200908/foldertree"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/8
71feb4c-240a-4475-a678-e2d86911abee/tree"
    type="application/cmistree+xml" />
    <link rel="describedby"
    href="http://localhost:8080/alfresco/s/cmis/type/cmis:folder" />
    <link rel="service"
    href="http://localhost:8080/alfresco/s/cmis" />
    <published>2009-11-13T13:14:29.927-06:00</published>
    <summary>User managed definitions</summary>
    <title>Data Dictionary</title>
    <updated>2009-11-13T13:14:29.948-06:00</updated>
    <app:edited>2009-11-13T13:14:29.948-06:00</app:edited>
    <alf:icon>
    http://localhost:8080/alfresco/images/icons/space-icon-default-
16.gif</alf:icon>
    <cmisra:object>
      <cmis:properties>
        <cmis:propertyId
propertyDefinitionId="cmis:allowedChildObjectTypeIds" />
        <cmis:propertyString propertyDefinitionId="cmis:path">
          <cmis:value>/Data Dictionary</cmis:value>
        </cmis:propertyString>
        <cmis:propertyString
propertyDefinitionId="cmis:lastModifiedBy">
          <cmis:value>System</cmis:value>
        </cmis:propertyString>
        <cmis:propertyId propertyDefinitionId="cmis:objectTypeId">
          <cmis:value>cmis:folder</cmis:value>
```

```
        </cmis:propertyId>
        <cmis:propertyString propertyDefinitionId="cmis:createdBy">
          <cmis:value>System</cmis:value>
        </cmis:propertyString>
        <cmis:propertyString propertyDefinitionId="cmis:name">
          <cmis:value>Data Dictionary</cmis:value>
        </cmis:propertyString>
        <cmis:propertyId propertyDefinitionId="cmis:objectId">
          <cmis:value>
          workspace://SpacesStore/871feb4c-240a-4475-a678-
e2d86911abee</cmis:value>
        </cmis:propertyId>
        <cmis:propertyDateTime
propertyDefinitionId="cmis:creationDate">
          <cmis:value>2009-11-13T13:14:29.927-06:00</cmis:value>
        </cmis:propertyDateTime>
        <cmis:propertyString propertyDefinitionId="cmis:changeToken" />
        <cmis:propertyId propertyDefinitionId="cmis:baseTypeId">
          <cmis:value>cmis:folder</cmis:value>
        </cmis:propertyId>
        <cmis:propertyId propertyDefinitionId="cmis:parentId">
          <cmis:value>
          workspace://SpacesStore/4eb6a431-3c56-4767-816a-
4ceca2295ae2</cmis:value>
        </cmis:propertyId>
        <cmis:propertyDateTime
propertyDefinitionId="cmis:lastModificationDate">
          <cmis:value>2009-11-13T13:14:29.948-06:00</cmis:value>
        </cmis:propertyDateTime>
      </cmis:properties>
    </cmisra:object>
    <cmisra:pathSegment>Data Dictionary</cmisra:pathSegment>
  </entry>
…SNIP…
```

It's a long response—verbosity is one of the common complaints some have with CMIS—so I've cut it off after the first two entries. What you are looking at is an Atom feed (http://www.ietf.org/rfc/rfc4287.txt) with CMIS (and Alfresco) extensions. Each entry in the feed corresponds to an object in the repository.

Notice that each entry has a set of links. Each link has a relationship ("rel") attribute, an HREF, and sometimes, a type attribute. You can think of these links as a navigation menu of sorts for that particular object. They tell you how to get more information about the object such as the object's relationships or the allowable actions for that object.

Listing the Available Types

You've seen how to determine the capabilities of the repository and how to get the contents of the root node. What about getting information about the types available in the content model? Look back at the repository info response. There's a collection that looks helpful:

```
<collection href="http://localhost:8080/alfresco/s/cmis/types">
  <atom:title>type collection</atom:title>
  <cmisra:collectionType>types</cmisra:collectionType>
```

```
</collection>
```

Invoking the type collection URL will give us a list of base types.

Doing this:

```
curl -uadmin:admin "http://localhost:8080/alfresco/s/cmis/types"
```

Returns this:

```
<feed xmlns="http://www.w3.org/2005/Atom"
xmlns:app="http://www.w3.org/2007/app"
xmlns:cmisra="http://docs.oasis-open.org/ns/cmis/restatom/200908/"
xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/"
xmlns:alf="http://www.alfresco.org"
xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/">
  <author>
    <name>admin</name>
  </author>
  <generator version="3.2.0 (r2 @build-number@)">Alfresco
  (Community)</generator>
  <icon>
  http://localhost:8080/alfresco/images/logo/AlfrescoLogo16.ico</icon>
  <id>urn:uuid:types-base</id>
  <link rel="service"
  href="http://localhost:8080/alfresco/s/cmis" />
  <link rel="self"
  href="http://localhost:8080/alfresco/s/cmis/types" />
  <link rel="first"

href="http://localhost:8080/alfresco/s/cmis/types?pageNo=1&amp;pageSize
=-1&amp;guest="
  type="application/atom+xml;type=feed" />
  <link rel="last"

href="http://localhost:8080/alfresco/s/cmis/types?pageNo=1&amp;pageSize
=-1&amp;guest="
  type="application/atom+xml;type=feed" />
  <title>Base Types</title>
  <updated>2009-11-23T09:43:28.589-06:00</updated>
  <opensearch:totalResults>4</opensearch:totalResults>
  <opensearch:startIndex>0</opensearch:startIndex>
  <opensearch:itemsPerPage>-1</opensearch:itemsPerPage>
  <cmisra:numItems>4</cmisra:numItems>
  <entry>
    <author>
      <name>admin</name>
    </author>
    <content>cmis:folder</content>
    <id>urn:uuid:type-cmis:folder</id>
    <link rel="self"
    href="http://localhost:8080/alfresco/s/cmis/type/cmis:folder" />
    <link rel="describedby"
    href="http://localhost:8080/alfresco/s/cmis/type/cmis:folder" />
    <link rel="down"

href="http://localhost:8080/alfresco/s/cmis/type/cmis:folder/children"
    type="application/atom+xml;type=feed" />
    <link rel="down"
```

```
href="http://localhost:8080/alfresco/s/cmis/type/cmis:folder/descendant
s"
    type="application/cmistree+xml" />
    <link rel="service"
    href="http://localhost:8080/alfresco/s/cmis" />
    <summary>Folder Type</summary>
    <title>Folder</title>
    <updated>2009-11-23T09:43:28.589-06:00</updated>
    <cmisra:type cmisra:id="cmis:folder"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="cmis:cmisTypeFolderDefinitionType">
      <cmis:id>cmis:folder</cmis:id>
      <cmis:localName>folder</cmis:localName>

<cmis:localNamespace>http://www.alfresco.org/model/cmis/1.0/cd04</cmis:
localNamespace>
      <cmis:displayName>Folder</cmis:displayName>
      <cmis:queryName>cmis:folder</cmis:queryName>
      <cmis:description>Folder Type</cmis:description>
      <cmis:baseId>cmis:folder</cmis:baseId>
      <cmis:creatable>true</cmis:creatable>
      <cmis:fileable>false</cmis:fileable>
      <cmis:queryable>true</cmis:queryable>
      <cmis:fulltextIndexed>true</cmis:fulltextIndexed>

<cmis:includedInSupertypeQuery>true</cmis:includedInSupertypeQuery>
      <cmis:controllablePolicy>false</cmis:controllablePolicy>
      <cmis:controllableACL>false</cmis:controllableACL>
    </cmisra:type>
  </entry>
  <entry>
    <author>
      <name>admin</name>
    </author>
    <content>cmis:relationship</content>
    <id>urn:uuid:type-cmis:relationship</id>
    <link rel="self"
    href="http://localhost:8080/alfresco/s/cmis/type/cmis:relationship"
/>
    <link rel="describedby"
    href="http://localhost:8080/alfresco/s/cmis/type/cmis:relationship"
/>
    <link rel="down"

href="http://localhost:8080/alfresco/s/cmis/type/cmis:relationship/chil
dren"
    type="application/atom+xml;type=feed" />
    <link rel="down"

href="http://localhost:8080/alfresco/s/cmis/type/cmis:relationship/desc
endants"
    type="application/cmistree+xml" />
    <link rel="service"
    href="http://localhost:8080/alfresco/s/cmis" />
    <summary>Relationship Type</summary>
    <title>Relationship</title>
    <updated>2009-11-23T09:43:28.589-06:00</updated>
    <cmisra:type cmisra:id="cmis:relationship"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="cmis:cmisTypeRelationshipDefinitionType">
      <cmis:id>cmis:relationship</cmis:id>
      <cmis:localName>relationship</cmis:localName>

<cmis:localNamespace>http://www.alfresco.org/model/cmis/1.0/cd04</cmis:
localNamespace>
      <cmis:displayName>Relationship</cmis:displayName>
      <cmis:queryName>cmis:relationship</cmis:queryName>
      <cmis:description>Relationship Type</cmis:description>
      <cmis:baseId>cmis:relationship</cmis:baseId>
      <cmis:creatable>false</cmis:creatable>
      <cmis:fileable>false</cmis:fileable>
      <cmis:queryable>false</cmis:queryable>
      <cmis:fulltextIndexed>false</cmis:fulltextIndexed>

<cmis:includedInSupertypeQuery>true</cmis:includedInSupertypeQuery>
      <cmis:controllablePolicy>false</cmis:controllablePolicy>
      <cmis:controllableACL>false</cmis:controllableACL>
    </cmisra:type>
  </entry>
  <entry>
    <author>
      <name>admin</name>
    </author>
    <content>cmis:document</content>
    <id>urn:uuid:type-cmis:document</id>
    <link rel="self"
    href="http://localhost:8080/alfresco/s/cmis/type/cmis:document" />
    <link rel="describedby"
    href="http://localhost:8080/alfresco/s/cmis/type/cmis:document" />
    <link rel="down"

href="http://localhost:8080/alfresco/s/cmis/type/cmis:document/children
"
    type="application/atom+xml;type=feed" />
    <link rel="down"

href="http://localhost:8080/alfresco/s/cmis/type/cmis:document/descenda
nts"
    type="application/cmistree+xml" />
    <link rel="service"
    href="http://localhost:8080/alfresco/s/cmis" />
    <summary>Document Type</summary>
    <title>Document</title>
    <updated>2009-11-23T09:43:28.589-06:00</updated>
    <cmisra:type cmisra:id="cmis:document"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="cmis:cmisTypeDocumentDefinitionType">
      <cmis:id>cmis:document</cmis:id>
      <cmis:localName>document</cmis:localName>

<cmis:localNamespace>http://www.alfresco.org/model/cmis/1.0/cd04</cmis:
localNamespace>
      <cmis:displayName>Document</cmis:displayName>
      <cmis:queryName>cmis:document</cmis:queryName>
      <cmis:description>Document Type</cmis:description>
      <cmis:baseId>cmis:document</cmis:baseId>
      <cmis:creatable>true</cmis:creatable>
```

```
      <cmis:fileable>true</cmis:fileable>
      <cmis:queryable>true</cmis:queryable>
      <cmis:fulltextIndexed>true</cmis:fulltextIndexed>

<cmis:includedInSupertypeQuery>true</cmis:includedInSupertypeQuery>
      <cmis:controllablePolicy>false</cmis:controllablePolicy>
      <cmis:controllableACL>false</cmis:controllableACL>
      <cmis:versionable>false</cmis:versionable>
      <cmis:contentStreamAllowed>allowed</cmis:contentStreamAllowed>
    </cmisra:type>
  </entry>
  …SNIP…
```

It's another Atom feed. This time, the feed contains one entry for every base content type in the content model. In the repository I'm working with, I've deployed a custom model that extends Alfresco's out-of-the-box model (if you want to follow along, I used the same SomeCo model used in the ecmarchitect.com custom types tutorial or Chapter 3 of the Alfresco Developer Guide). Let's see what child types descend from cmis:document in the SomeCo model. To do that, specify the type ID followed by either "children" for the immediate children or "descendants" for the children, each of the children's children, and so on.

Doing this:

```
curl -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/type/cmis:document/descendants"
```

Returns this:

```
…SNIP…
<entry>
  <author>
    <name>admin</name>
  </author>
  <content>D:sc:doc</content>
  <id>urn:uuid:type-D:sc:doc</id>
  <link rel="self"
  href="http://localhost:8080/alfresco/s/cmis/type/D:sc:doc" />
  <link rel="describedby"
  href="http://localhost:8080/alfresco/s/cmis/type/cmis:document" />
  <link rel="up"
  href="http://localhost:8080/alfresco/s/cmis/type/cmis:document"
  type="application/atom+xml;type=entry" />
  <link rel="down"
  href="http://localhost:8080/alfresco/s/cmis/type/D:sc:doc/children"
  type="application/atom+xml;type=feed" />
  <link rel="down"

href="http://localhost:8080/alfresco/s/cmis/type/D:sc:doc/descendants"
  type="application/cmistree+xml" />
  <link rel="service"
  href="http://localhost:8080/alfresco/s/cmis" />
  <summary>Someco Document</summary>
  <title>Someco Document</title>
  <updated>2009-11-13T18:50:21.901-06:00</updated>
  <cmisra:type cmisra:id="D:sc:doc"
```

```
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="cmis:cmisTypeDocumentDefinitionType">
    <cmis:id>D:sc:doc</cmis:id>
    <cmis:localName>doc</cmis:localName>
    <cmis:localNamespace>
http://www.someco.com/model/content/1.0</cmis:localNamespace>
    <cmis:displayName>Someco Document</cmis:displayName>
    <cmis:queryName>sc:doc</cmis:queryName>
    <cmis:description></cmis:description>
    <cmis:baseId>cmis:document</cmis:baseId>
    <cmis:parentId>cmis:document</cmis:parentId>
    <cmis:creatable>true</cmis:creatable>
    <cmis:fileable>true</cmis:fileable>
    <cmis:queryable>true</cmis:queryable>
    <cmis:fulltextIndexed>true</cmis:fulltextIndexed>
    <cmis:includedInSupertypeQuery>
    true</cmis:includedInSupertypeQuery>
    <cmis:controllablePolicy>false</cmis:controllablePolicy>
    <cmis:controllableACL>false</cmis:controllableACL>
    <cmis:versionable>false</cmis:versionable>
    <cmis:contentStreamAllowed>allowed</cmis:contentStreamAllowed>
  </cmisra:type>
  <entry>
    <author>
      <name>admin</name>
    </author>
    <content>D:sc:doc</content>
    <id>urn:uuid:type-D:sc:doc</id>
    <link rel="self"
    href="http://localhost:8080/alfresco/s/cmis/type/D:sc:doc" />
    <link rel="describedby"
    href="http://localhost:8080/alfresco/s/cmis/type/cmis:document" />
    <link rel="up"
    href="http://localhost:8080/alfresco/s/cmis/type/cmis:document"
    type="application/atom+xml;type=entry" />
    <link rel="down"
    href="http://localhost:8080/alfresco/s/cmis/type/D:sc:doc/children"
    type="application/atom+xml;type=feed" />
    <link rel="down"

href="http://localhost:8080/alfresco/s/cmis/type/D:sc:doc/descendants"
    type="application/cmistree+xml" />
    <link rel="service"
    href="http://localhost:8080/alfresco/s/cmis" />
    <summary>Someco Document</summary>
    <title>Someco Document</title>
    <updated>2009-11-13T18:50:21.901-06:00</updated>
    <cmisra:type cmisra:id="D:sc:doc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="cmis:cmisTypeDocumentDefinitionType">
      <cmis:id>D:sc:doc</cmis:id>
      <cmis:localName>doc</cmis:localName>
      <cmis:localNamespace>
      http://www.someco.com/model/content/1.0</cmis:localNamespace>
      <cmis:displayName>Someco Document</cmis:displayName>
      <cmis:queryName>sc:doc</cmis:queryName>
      <cmis:description></cmis:description>
      <cmis:baseId>cmis:document</cmis:baseId>
      <cmis:parentId>cmis:document</cmis:parentId>
```

```
      <cmis:creatable>true</cmis:creatable>
      <cmis:fileable>true</cmis:fileable>
      <cmis:queryable>true</cmis:queryable>
      <cmis:fulltextIndexed>true</cmis:fulltextIndexed>
      <cmis:includedInSupertypeQuery>
      true</cmis:includedInSupertypeQuery>
      <cmis:controllablePolicy>false</cmis:controllablePolicy>
      <cmis:controllableACL>false</cmis:controllableACL>
      <cmis:versionable>false</cmis:versionable>
      <cmis:contentStreamAllowed>
      allowed</cmis:contentStreamAllowed>
    </cmisra:type>
    …SNIP…
```

This response contains a list of all of the custom types that descend from Alfresco's cm:content type (I'm just showing SomeCo Document in this snippet). Key elements to pay attention to are "cmis:id" and "cmis:queryName". The id is what should be used when creating new content of that type ("D:sc:doc"). The query name is what should be used when executing CMIS queries against that type.

Creating Folders

You've seen how to query the CMIS repository and the Atom feeds that come back. What happens when you want to create a new object in the repository? Intuitively, if a GET returns a feed of Atom entries, a POSTed Atom entry ought to create a new object in the repository, and luckily, it works just like you'd expect.

Let's create a new "Someco" folder in the repository. Then, we'll create some folders within that to hold some content.

The first step is to create the Atom entry that will get POSTed. I'm going to call mine testCreateSomecoFolder.atom.xml, but the file name is not important. Here's what it looks like:

```
<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
xmlns:cmisra="http://docs.oasis-open.org/ns/cmis/restatom/200908/"
xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/">
    <title>Someco</title>
    <cmisra:object>
       <cmis:properties>
        <cmis:propertyId
propertyDefinitionId="cmis:objectTypeId"><cmis:value>cmis:folder</cmis:
value></cmis:propertyId>
       </cmis:properties>
    </cmisra:object>
</entry>
```

As you can see by the root element and the default namespace, it's an Atom entry. The title element will become the name of the folder. The cmis:objectTypeId property specifies the object type, cmis:folder. If you're familiar with Alfresco, that maps to Alfresco's cm:folder type.

The next step is to post it:

```
curl -X POST -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/p/children" -H "Content-Type:
application/atom+xml" -d @/Users/jpotts/testCreateSomecoFolder.atom.xml
```

That creates a Someco folder in the root directory. Note the content-type header. If you are using something other than curl, make sure you are setting the content-type appropriately.

In Alfresco, there is often more than one form of URL you can use to accomplish the same thing. In this case, I'm using one that accepts the path in which the folder should be created as part of the URL. In the URL above, I didn't specify a path (it is supposed to appear after the "p/") so the root was assumed.

Alfresco provides a web script index that can be used to find different URL variations, arguments, and so on. The URL to see the CMIS family of web scripts is http://localhost:8080/alfresco/s/index/family/CMIS. Go to the URL and do a search for "createFolder" to see a list of the other acceptable URL formats for the POST.

I'll repeat the POST to create several folders. Unfortunately there isn't currently a way to POST multiple entries at once.

```
curl -X POST -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/p/Someco/children" -H "Content-
Type: application/atom+xml" -d
@/Users/jpotts/testCreateMarketingFolder.atom.xml
curl -X POST -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/p/Someco/Marketing/children" -H
"Content-Type: application/atom+xml" -d
@/Users/jpotts/testCreateWhitepapersFolder.atom.xml
curl -X POST -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/p/Someco/children" -H "Content-
Type: application/atom+xml" -d
@/Users/jpotts/testCreateSalesFolder.atom.xml
```

After this, the repository will have the following structure:

```
/
--/Someco
----/Sales
----/Marketing
------/Whitepapers
```

Creating a Document

Now that the folder structure exists in the repository, it's time to load in a few documents. As I mentioned earlier, most of the time, you'll probably be working with a higher-level client API. Uploading files is certainly something

that can be tedious when working directly with the AtomPub binding. First, you encode the file, then you wrap it in Atom XML, then you post it. Let's go through it.

The first step is that the documents you want to upload have to be Base64 encoded. There are a lot of different ways to do that. On Linux, Mac OS X, and Windows (through Cygwin) you can use openssl.

I'm going to upload a Word document and a PDF, so to Base64 encode those, I execute:

```
openssl base64 -in ./sample-a.doc -out ./sample-a.doc.base64
openssl base64 -in ./sample-b.pdf -out ./sample-b.pdf.base64
```

Once that's done, the next step is to wrap the Base64 encoded files with Atom Entry XML. The encoded content is wrapped by the content element, which includes a type attribute that specifies the content's mime type:

```
<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
xmlns:cmisra="http://docs.oasis-open.org/ns/cmis/restatom/200908/"
xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908">
    <title>sample-a.doc</title>
    <summary>A sample whitepaper named Sample A</summary>
    <content type="application/msword">
0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAOwADAP7/CQAGAAAAAAAAAAAAAAACAAAA
gwAAAAAAAAAAEAAAAgAAAAEAAAD+////AAAAAAAAAACAAAAA////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
…SNIP…
</content>
    <cmisra:object>
       <cmis:properties>
         <cmis:propertyId
propertyDefinitionId="cmis:objectTypeId"><cmis:value>D:sc:whitepaper</c
mis:value></cmis:propertyId>
       </cmis:properties>
    </cmisra:object>
</entry>
```

The title will become the name of the file, the summary will be the summary (in Alfresco, the cm:description property), and the type is specified by the cmis:objectTypeId. In this case, I'm creating both of these as Someco Whitepaper objects. If I wanted to set other metadata values on the new object I could add the property values to the list of cmis:properties.

Now the Atom entry XML is ready to post, which is the final step. The URL is identical to the one used earlier to create a folder:

```
curl -X POST -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/p/Someco/Marketing/Whitepapers/c
hildren" -H "Content-Type: application/atom+xml" -d
@/Users/jpotts/testCreateSampleA.atom.xml
```

I'll repeat the same steps for the sample-b.pdf whitepaper, remembering to change the mime type from application/msword to application/pdf.

You can verify everything worked out okay by logging in to the Alfresco Explorer web client, but that feels like cheating. Instead, ask CMIS for the children of the Whitepapers folder:

Doing this:

```
curl -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/p/Someco
/Marketing/Whitepapers/children"
```

Returns this:

```
…SNIP…
<entry>
  <author>
    <name>admin</name>
  </author>
  <content type="application/pdf"

src="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/93
61bb2e-30f6-4d97-a7e3-ec97777bae46/content.pdf" />
  <id>urn:uuid:9361bb2e-30f6-4d97-a7e3-ec97777bae46</id>
  <link rel="self"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/9
361bb2e-30f6-4d97-a7e3-ec97777bae46" />
  <link rel="enclosure"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/9
361bb2e-30f6-4d97-a7e3-ec97777bae46/content.pdf"
  type="application/pdf" />
  <link rel="edit"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/9
361bb2e-30f6-4d97-a7e3-ec97777bae46" />
  <link rel="edit-media"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/9
361bb2e-30f6-4d97-a7e3-ec97777bae46/content.pdf"
  type="application/pdf" />
  <link rel="http://docs.oasis-
open.org/ns/cmis/link/200908/allowableactions"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/9
361bb2e-30f6-4d97-a7e3-ec97777bae46/allowableactions" />
  <link rel="http://docs.oasis-
open.org/ns/cmis/link/200908/relationships"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/9
361bb2e-30f6-4d97-a7e3-ec97777bae46/rels" />
  <link rel="up"

href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/9
361bb2e-30f6-4d97-a7e3-ec97777bae46/parents"
  type="application/atom+xml;type=feed" />
  <link rel="version-history"
```

```
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/9
361bb2e-30f6-4d97-a7e3-ec97777bae46/versions" />
  <link rel="describedby"
  href="http://localhost:8080/alfresco/s/cmis/type/D:sc:whitepaper" />
  <link rel="service"
  href="http://localhost:8080/alfresco/s/cmis" />
  <published>2009-11-13T19:30:33.555-06:00</published>
  <summary>Sample whitepaper B</summary>
  <title>sample-b.pdf</title>
  <updated>2009-11-13T19:30:33.638-06:00</updated>
  <app:edited>2009-11-13T19:30:33.638-06:00</app:edited>

<alf:icon>http://localhost:8080/alfresco/images/filetypes/pdf.gif</alf:
icon>
  <cmisra:object>
    <cmis:properties>
      <cmis:propertyInteger
propertyDefinitionId="cmis:contentStreamLength">
        <cmis:value>117248</cmis:value>
      </cmis:propertyInteger>
      <cmis:propertyId propertyDefinitionId="cmis:objectTypeId">
        <cmis:value>D:sc:whitepaper</cmis:value>
      </cmis:propertyId>
      <cmis:propertyString
propertyDefinitionId="cmis:versionSeriesCheckedOutBy" />
      <cmis:propertyId
propertyDefinitionId="cmis:versionSeriesCheckedOutId" />
      <cmis:propertyId propertyDefinitionId="cmis:versionSeriesId">
        <cmis:value>workspace://SpacesStore/9361bb2e-30f6-4d97-a7e3-
ec97777bae46</cmis:value>
      </cmis:propertyId>
      <cmis:propertyString propertyDefinitionId="cmis:versionLabel" />
      <cmis:propertyBoolean
propertyDefinitionId="cmis:isLatestVersion">
        <cmis:value>true</cmis:value>
      </cmis:propertyBoolean>
      <cmis:propertyBoolean
propertyDefinitionId="cmis:isVersionSeriesCheckedOut">
        <cmis:value>false</cmis:value>
      </cmis:propertyBoolean>
      <cmis:propertyString propertyDefinitionId="cmis:lastModifiedBy">
        <cmis:value>admin</cmis:value>
      </cmis:propertyString>
      <cmis:propertyString propertyDefinitionId="cmis:createdBy">
        <cmis:value>admin</cmis:value>
      </cmis:propertyString>
      <cmis:propertyBoolean
propertyDefinitionId="cmis:isLatestMajorVersion">
        <cmis:value>false</cmis:value>
      </cmis:propertyBoolean>
      <cmis:propertyId propertyDefinitionId="cmis:contentStreamId">
        <cmis:value>cm:content</cmis:value>
      </cmis:propertyId>
      <cmis:propertyString propertyDefinitionId="cmis:name">
        <cmis:value>sample-b.pdf</cmis:value>
      </cmis:propertyString>
      <cmis:propertyString
propertyDefinitionId="cmis:contentStreamMimeType">
```

```
        <cmis:value>application/pdf</cmis:value>
      </cmis:propertyString>
      <cmis:propertyDateTime propertyDefinitionId="cmis:creationDate">
        <cmis:value>2009-11-13T19:30:33.555-06:00</cmis:value>
      </cmis:propertyDateTime>
      <cmis:propertyString propertyDefinitionId="cmis:changeToken" />
      <cmis:propertyString propertyDefinitionId="sc:campaign" />
      <cmis:propertyString propertyDefinitionId="cmis:checkinComment"
/>
      <cmis:propertyId propertyDefinitionId="cmis:objectId">
        <cmis:value>workspace://SpacesStore/9361bb2e-30f6-4d97-a7e3-
ec97777bae46</cmis:value>
      </cmis:propertyId>
      <cmis:propertyBoolean propertyDefinitionId="cmis:isImmutable">
        <cmis:value>false</cmis:value>
      </cmis:propertyBoolean>
      <cmis:propertyBoolean propertyDefinitionId="cmis:isMajorVersion">
        <cmis:value>false</cmis:value>
      </cmis:propertyBoolean>
      <cmis:propertyId propertyDefinitionId="cmis:baseTypeId">
        <cmis:value>cmis:document</cmis:value>
      </cmis:propertyId>
      <cmis:propertyDateTime
propertyDefinitionId="cmis:lastModificationDate">
        <cmis:value>2009-11-13T19:30:33.638-06:00</cmis:value>
      </cmis:propertyDateTime>
      <cmis:propertyString
propertyDefinitionId="cmis:contentStreamFileName">
        <cmis:value>sample-b.pdf</cmis:value>
      </cmis:propertyString>
    </cmis:properties>
  </cmisra:object>
  <cmisra:pathSegment>sample-b.pdf</cmisra:pathSegment>
</entry>
…SNIP…
```

Now let's set a property value on the Sample B whitepaper. Instead of a POST, which is used to create a new resource, do a PUT, which is used to update an existing resource. The Atom is the same, but it includes only the properties that need to be set (although if you leave off the title, Alfresco will reset the name to the object ID for some reason, so even if you aren't changing the name, leave it in):

Here's the Atom entry:

```
<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
xmlns:cmisra="http://docs.oasis-open.org/ns/cmis/restatom/200908/"
xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/">
    <title>sample-b.pdf</title>
    <summary>Sample whitepaper B</summary>
    <cmisra:object>
       <cmis:properties>
       <cmis:propertyString
propertyDefinitionId="sc:campaign"><cmis:value>Foo</cmis:value></cmis:p
ropertyString>
       </cmis:properties>
    </cmisra:object>
```

```
</entry>
```

In this case, I'm setting the `sc:campaign` property to "Foo". Behind the scenes, I know there is a constraint on the `sc:campaign` property that restricts the values the property can contain, and "Foo" isn't one of them, but let's see what happens:

```
curl -X PUT -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/9361bb
2e-30f6-4d97-a7e3-ec97777bae46" -H "Content-Type:
application/atom+xml;type=entry" -d
@/Users/jpotts/testUpdateSampleB.atom.xml
```

The result is an Error 500, integrity violation. A portion of the message appears below:

```
Message:</b></td><td>10130028 Found 1 integrity violations:
Invalid property value:
   Node: workspace://SpacesStore/9361bb2e-30f6-4d97-a7e3-ec97777bae46
   Type: {http://www.someco.com/model/content/1.0}whitepaper
   Property: {http://www.someco.com/model/content/1.0}campaign
   Constraint: 10130027 The value is not an allowed value:
Foo</td></tr>
            <tr><td></td><td> </td></tr>

<tr><td><b>Exception:</b></td><td>org.alfresco.repo.node.integrity.Inte
grityException - 10130028 Found 1 integrity violations:
Invalid property value:
   Node: workspace://SpacesStore/9361bb2e-30f6-4d97-a7e3-ec97777bae46
   Type: {http://www.someco.com/model/content/1.0}whitepaper
   Property: {http://www.someco.com/model/content/1.0}campaign
   Constraint: 10130027 The value is not an allowed value:
Foo</td></tr>
```

I could go look at the custom Alfresco content model XML to find the allowed values, but let's see if I can get that info through CMIS:

Doing this:

```
curl -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/type/D:sc:whitepaper"
```

Returns this:

```
…SNIP…
<cmis:propertyStringDefinition>
  <cmis:id>sc:campaign</cmis:id>
  <cmis:localName>campaign</cmis:localName>

<cmis:localNamespace>http://www.someco.com/model/content/1.0</cmis:loca
lNamespace>
  <cmis:displayName>sc:campaign</cmis:displayName>
  <cmis:queryName>sc:campaign</cmis:queryName>
  <cmis:propertyType>string</cmis:propertyType>
  <cmis:cardinality>multi</cmis:cardinality>
  <cmis:updatability>readwrite</cmis:updatability>
  <cmis:inherited>true</cmis:inherited>
```

```
  <cmis:required>false</cmis:required>
  <cmis:queryable>true</cmis:queryable>
  <cmis:orderable>false</cmis:orderable>
  <cmis:openChoice>false</cmis:openChoice>
  <cmis:choiceString displayName="Private Event Retailing">
    <cmis:value>Private Event Retailing</cmis:value>
  </cmis:choiceString>
  <cmis:choiceString displayName="Application Syndication">
    <cmis:value>Application Syndication</cmis:value>
  </cmis:choiceString>
  <cmis:choiceString displayName="Social Shopping">
    <cmis:value>Social Shopping</cmis:value>
  </cmis:choiceString>
</cmis:propertyStringDefinition>
…SNIP…
```

The `cmis:openChoice` element has a value of false, which means the value must come from one of the values listed in a `cmis:choiceString` element. Editing the Atom entry to change the value from "Foo" to one of the valid choice strings ("Application Syndication", for example) will make the PUT successful.

While we're here, note that the `cmis:cardinality` element is set to "multi" which indicates that this is a multi-value field. To update a multi-value property, simply include multiple value elements, like this:

```
…SNIP…
<cmisra:object>
    <cmis:properties>
        <cmis:propertyString propertyDefinitionId="sc:campaign">
            <cmis:value>Private Event Retailing</cmis:value>
            <cmis:value>Application Syndication</cmis:value>
        </cmis:propertyString>
    </cmis:properties>
</cmisra:object>
…SNIP…
```

Working with Relationships

The SomeCo Document type includes an association called "Related Documents" which can be used to relate any SomeCo Document to any other SomeCo Document (including descendant types). You can create, retrieve, and delete an object's relationships through CMIS.

For example, to create an sc:relatedDocuments relationship between sample-b.pdf and sample-a.doc, first, create an Atom entry that identifies sample-b.pdf's object ID as the source, sample-a.doc's object ID as the target, and the relationship type:

```
<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
xmlns:cmisra="http://docs.oasis-open.org/ns/cmis/restatom/200908/"
xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/">
  <cmisra:object>
    <cmis:properties>
      <cmis:propertyId propertyDefinitionId="cmis:targetId">
```

```
        <cmis:value>workspace://SpacesStore/39c86907-e21f-4165-9b90-
62c3bb7940c3</cmis:value>
      </cmis:propertyId>
      <cmis:propertyId propertyDefinitionId="cmis:objectTypeId">
        <cmis:value>R:sc:relatedDocuments</cmis:value>
      </cmis:propertyId>
      <cmis:propertyId propertyDefinitionId="cmis:sourceId">
        <cmis:value>workspace://SpacesStore/9361bb2e-30f6-4d97-a7e3-
ec97777bae46</cmis:value>
      </cmis:propertyId>
    </cmis:properties>
  </cmisra:object>
</entry>
```

Then, post the Atom to the /rels URL:

```
curl -X POST -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/i/9361bb2e-30f6-4d97-a7e3-
ec97777bae46/rels" -H "Content-Type: application/atom+xml" -d
@/Users/jpotts/testCreateRelationship.atom.xml
```

Did it work? Let's use CMIS to find out.

Doing this:

```
curl -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/9361bb
2e-30f6-4d97-a7e3-ec97777bae46/rels?includeSubRelationshipTypes=true"
```

Returns this:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
xmlns:app="http://www.w3.org/2007/app"
xmlns:cmisra="http://docs.oasis-open.org/ns/cmis/restatom/200908/"
xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/"
xmlns:alf="http://www.alfresco.org"
xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/">
  <author>
    <name>admin</name>
  </author>
  <generator version="3.2.0 (r2 @build-number@)">Alfresco
(Community)</generator>
<icon>http://localhost:8080/alfresco/images/logo/AlfrescoLogo16.ico</ic
on>
  <id>urn:uuid:9361bb2e-30f6-4d97-a7e3-ec97777bae46-relationships</id>
  <link rel="service"
  href="http://localhost:8080/alfresco/s/cmis" />
  <link rel="self"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/9
361bb2e-30f6-4d97-a7e3-
ec97777bae46/rels?alf_ticket=TICKET_e5719b83f95e8e16020b69682a71647fef0
ba139&amp;includeSubRelationshipTypes=true" />
  <link rel="via"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/9
361bb2e-30f6-4d97-a7e3-ec97777bae46" />
  <link rel="first"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/9
```

```
361bb2e-30f6-4d97-a7e3-
ec97777bae46/rels?alf_ticket=TICKET_e5719b83f95e8e16020b69682a71647fef0
ba139&amp;includeSubRelationshipTypes=true&amp;pageNo=1&amp;pageSize=-
1&amp;guest=&amp;format=atomfeed"
  type="application/atom+xml;type=feed" />
  <link rel="last"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/9
361bb2e-30f6-4d97-a7e3-
ec97777bae46/rels?alf_ticket=TICKET_e5719b83f95e8e16020b69682a71647fef0
ba139&amp;includeSubRelationshipTypes=true&amp;pageNo=1&amp;pageSize=-
1&amp;guest=&amp;format=atomfeed"
  type="application/atom+xml;type=feed" />
  <title>sample-b.pdf Relationships</title>
  <updated>2009-11-16T22:27:05.776-06:00</updated>
  <opensearch:totalResults>1</opensearch:totalResults>
  <opensearch:startIndex>0</opensearch:startIndex>
  <opensearch:itemsPerPage>-1</opensearch:itemsPerPage>
  <cmisra:numItems>1</cmisra:numItems>
  <entry>
    <author>
      <name>2009-11-17T08:56:42.745-06:00</name>
    </author>
    <content>workspace://SpacesStore/9361bb2e-30f6-4d97-a7e3-
ec97777bae46</content>
    <id>workspace://SpacesStore/9361bb2e-30f6-4d97-a7e3-
ec97777bae46</id>
    <link rel="self"
href="http://localhost:8080/alfresco/s/cmis/rel/s/workspace:SpacesStore
/i/9361bb2e-30f6-4d97-a7e3-
ec97777bae46/type/R:sc:relatedDocuments/target/s/workspace:SpacesStore/
i/39c86907-e21f-4165-9b90-62c3bb7940c3" />
    <link rel="edit"
href="http://localhost:8080/alfresco/s/cmis/rel/s/workspace:SpacesStore
/i/9361bb2e-30f6-4d97-a7e3-
ec97777bae46/type/R:sc:relatedDocuments/target/s/workspace:SpacesStore/
i/39c86907-e21f-4165-9b90-62c3bb7940c3" />
    <link rel="describedby"
href="http://localhost:8080/alfresco/s/cmis/type/R:sc:relatedDocuments"
/>
    <link rel="http://docs.oasis-open.org/ns/cmis/link/200908/source"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/9
361bb2e-30f6-4d97-a7e3-ec97777bae46" />
    <link rel="http://docs.oasis-open.org/ns/cmis/link/200908/target"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/3
9c86907-e21f-4165-9b90-62c3bb7940c3" />
    <link rel="service"
    href="http://localhost:8080/alfresco/s/cmis" />
    <published>2009-11-17T08:56:42.745-06:00</published>
    <summary>workspace://SpacesStore/9361bb2e-30f6-4d97-a7e3-
ec97777bae46</summary>
    <title>workspace://SpacesStore/9361bb2e-30f6-4d97-a7e3-
ec97777bae46</title>
    <updated>2009-11-17T08:56:42.745-06:00</updated>
    <app:edited>2009-11-17T08:56:42.745-06:00</app:edited>
    <cmisra:object>
      <cmis:properties>
        <cmis:propertyId propertyDefinitionId="cmis:targetId">
          <cmis:value>workspace://SpacesStore/39c86907-e21f-4165-9b90-
62c3bb7940c3</cmis:value>
```

```
        </cmis:propertyId>
        <cmis:propertyString propertyDefinitionId="cmis:lastModifiedBy"
/>
        <cmis:propertyId propertyDefinitionId="cmis:objectTypeId">
          <cmis:value>R:sc:relatedDocuments</cmis:value>
        </cmis:propertyId>
        <cmis:propertyId propertyDefinitionId="cmis:sourceId">
          <cmis:value>workspace://SpacesStore/9361bb2e-30f6-4d97-a7e3-
ec97777bae46</cmis:value>
        </cmis:propertyId>
        <cmis:propertyString propertyDefinitionId="cmis:createdBy" />
        <cmis:propertyId propertyDefinitionId="cmis:objectId">
          <cmis:value>workspace://SpacesStore/9361bb2e-30f6-4d97-a7e3-
ec97777bae46</cmis:value>
        </cmis:propertyId>
        <cmis:propertyDateTime propertyDefinitionId="cmis:creationDate"
/>
        <cmis:propertyString propertyDefinitionId="cmis:changeToken" />
        <cmis:propertyId propertyDefinitionId="cmis:baseTypeId">
          <cmis:value>cmis:relationship</cmis:value>
        </cmis:propertyId>
        <cmis:propertyDateTime
propertyDefinitionId="cmis:lastModificationDate" />
      </cmis:properties>
    </cmisra:object>
  </entry>
</feed>
```

Note the use of the "includeSubRelationshipTypes=true" in the POSTed URL. According to the current draft of the specification, the default value for this is false. Unfortunately, if that is set to false, it essentially means, give me only the relationships of the specified type, which is "cmis:relationship", by default. So you must either specify the relationship type you want returned using the relationshipType argument (although I couldn't get this to work) or you must specify true to get all relationships returned.

To delete the relationship, execute a DELETE that includes the source, target, and relationship type:

```
curl -X DELETE -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/rel/s/workspace:SpacesStore/i/77
7aad53-8b18-4f1f-80f6-
e420936b3def/type/R:sc:relatedDocuments/target/s/workspace:SpacesStore/
i/39c86907-e21f-4165-9b90-
62c3bb7940c3?alf_ticket=TICKET_aba36d4cd35771601b434cbbd80282f5fa4f88ca
"
```

If you didn't have access to the model, how would you know which relationships exist and how they are defined? In Alfresco, relationships (associations) are part of the type, so you might think the relationship would show up as part of the CMIS entry for a specific type. But in CMIS, a relationship is a first class type and any associations in your content model are treated as children of "cmis:relationship". So, to get a list of the available relationships, you do a GET on the CMIS type collection URL and ask for the descendants of cmis:relationship, like this:

```
curl -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/type/cmis:relationship/descendan
ts"
```

Which returns:

```
…SNIP…
<entry>
  <author>
    <name>admin</name>
  </author>
  <content>R:sc:relatedDocuments</content>
  <id>urn:uuid:type-R:sc:relatedDocuments</id>
  <link rel="self"
href="http://localhost:8080/alfresco/s/cmis/type/R:sc:relatedDocuments"
/>
  <link rel="describedby"
  href="http://localhost:8080/alfresco/s/cmis/type/cmis:relationship"
/>
  <link rel="up"
  href="http://localhost:8080/alfresco/s/cmis/type/cmis:relationship"
  type="application/atom+xml;type=entry" />
  <link rel="down"
href="http://localhost:8080/alfresco/s/cmis/type/R:sc:relatedDocuments/
children"
  type="application/atom+xml;type=feed" />
  <link rel="down"
href="http://localhost:8080/alfresco/s/cmis/type/R:sc:relatedDocuments/
descendants"
  type="application/cmistree+xml" />
  <link rel="service"
  href="http://localhost:8080/alfresco/s/cmis" />
  <summary>Related Documents</summary>
  <title>Related Documents</title>
  <updated>2009-11-17T09:41:03.297-06:00</updated>
  <cmisra:type cmisra:id="R:sc:relatedDocuments"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="cmis:cmisTypeRelationshipDefinitionType">
    <cmis:id>R:sc:relatedDocuments</cmis:id>
    <cmis:localName>relatedDocuments</cmis:localName>
    <cmis:localNamespace>
    http://www.someco.com/model/content/1.0</cmis:localNamespace>
    <cmis:displayName>Related Documents</cmis:displayName>
    <cmis:queryName>sc:relatedDocuments</cmis:queryName>
    <cmis:description></cmis:description>
    <cmis:baseId>cmis:relationship</cmis:baseId>
    <cmis:parentId>cmis:relationship</cmis:parentId>
    <cmis:creatable>true</cmis:creatable>
```

```
    <cmis:fileable>false</cmis:fileable>
    <cmis:queryable>false</cmis:queryable>
    <cmis:fulltextIndexed>false</cmis:fulltextIndexed>
    <cmis:includedInSupertypeQuery>
    true</cmis:includedInSupertypeQuery>
    <cmis:controllablePolicy>false</cmis:controllablePolicy>
    <cmis:controllableACL>false</cmis:controllableACL>
    <cmis:allowedSourceTypes>D:sc:doc</cmis:allowedSourceTypes>
    <cmis:allowedTargetTypes>D:sc:doc</cmis:allowedTargetTypes>
  </cmisra:type>
</entry>
…SNIP…
```

Alfresco supports aspects as well as types. In brief, an aspect is like a "free-floating" type—it exists separate from the inheritance hierarchy of types. If you are familiar with the SomeCo content model, you may notice that one of the SomeCo associations doesn't show up as a child of cmis:relationship: sc:ratings. That's because that particular association is defined in an aspect called sc:rateable, and the CMIS specification does not currently support aspects. Unfortunately, that makes the sc:ratings association invisible to CMIS.

Check-out/Check-in

Some CMIS repositories require that a document be checked out before its content can be updated. You can check that for a given repository by inspecting its ContentStreamUpdatability capability. Here's what it looks like for Alfresco:

```
<cmis:capabilityContentStreamUpdatability>anytime</cmis:capabilityConte
ntStreamUpdatability>
```

The value of "anytime" means the document does not have to be checked out to be updated—it can be updated at any time. If it were set to "pwconly" it would mean a checkout is required. If it were set to "none" it would mean the content stream could never be updated, and that would be the saddest CMIS repository ever, in my opinion (or at least the most boring). Almost as bad as one that doesn't support queries, but I digress.

Let's check out one of the sample documents uploaded earlier, make some changes, and check it back in. To check out a document you must post an Atom entry that contains the object's object ID.

Here's the Atom:

```
<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
xmlns:cmisra="http://docs.oasis-open.org/ns/cmis/restatom/200908/"
xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/">
  <cmisra:object>
    <cmis:properties>
      <cmis:propertyId
propertyDefinitionId="cmis:objectId"><cmis:value>workspace://SpacesStor
e/39c86907-e21f-4165-9b90-62c3bb7940c3</cmis:value></cmis:propertyId>
    </cmis:properties>
```

```
  </cmisra:object>
</entry>
```

Which gets posted like this:

```
curl -X POST -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/checkedout" -H "Content-Type:
application/atom+xml;type=entry;charset=UTF-8" -d
@/Users/jpotts/testCheckoutSampleA.atom.xml
```

What comes back is an Atom entry of the Private Working Copy (PWC) of the checked out document:

```
<entry xmlns="http://www.w3.org/2005/Atom"
xmlns:app="http://www.w3.org/2007/app"
xmlns:cmisra="http://docs.oasis-open.org/ns/cmis/restatom/200908/"
xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/"
xmlns:alf="http://www.alfresco.org">
  <author>
    <name>admin</name>
  </author>
  <content type="application/msword"
src="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/b6
10bf02-c142-45d3-a941-52cf71862b39/content.doc" />
  <id>urn:uuid:b610bf02-c142-45d3-a941-52cf71862b39</id>
  <link rel="self"
href="http://localhost:8080/alfresco/s/cmis/pwc/s/workspace:SpacesStore
/i/b610bf02-c142-45d3-a941-52cf71862b39" />
  <link rel="enclosure"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/b
610bf02-c142-45d3-a941-52cf71862b39/content.doc"
  type="application/msword" />
  <link rel="edit"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/b
610bf02-c142-45d3-a941-52cf71862b39" />
  <link rel="edit-media"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/b
610bf02-c142-45d3-a941-52cf71862b39/content.doc"
  type="application/msword" />
  <link rel="http://docs.oasis-
open.org/ns/cmis/link/200908/allowableactions"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/b
610bf02-c142-45d3-a941-52cf71862b39/allowableactions" />
  <link rel="http://docs.oasis-
open.org/ns/cmis/link/200908/relationships"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/b
610bf02-c142-45d3-a941-52cf71862b39/rels" />
  <link rel="up"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/b
610bf02-c142-45d3-a941-52cf71862b39/parents"
  type="application/atom+xml;type=feed" />
  <link rel="version-history"
href="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/b
610bf02-c142-45d3-a941-52cf71862b39/versions" />
  <link rel="describedby"
  href="http://localhost:8080/alfresco/s/cmis/type/D:sc:whitepaper" />
  <link rel="service"
  href="http://localhost:8080/alfresco/s/cmis" />
```

```
  <published>2009-11-16T22:27:40.846-06:00</published>
  <summary>A sample whitepaper named Sample A</summary>
  <title>sample-a (Working Copy).doc</title>
  <updated>2009-11-16T22:27:40.911-06:00</updated>
  <app:edited>2009-11-16T22:27:40.911-06:00</app:edited>
<alf:icon>http://localhost:8080/alfresco/images/filetypes/doc.gif</alf:
icon>
  <cmisra:object>
    <cmis:properties>
      <cmis:propertyInteger
propertyDefinitionId="cmis:contentStreamLength">
        <cmis:value>68608</cmis:value>
      </cmis:propertyInteger>
      <cmis:propertyId propertyDefinitionId="cmis:objectTypeId">
        <cmis:value>D:sc:whitepaper</cmis:value>
      </cmis:propertyId>
      <cmis:propertyString
propertyDefinitionId="cmis:versionSeriesCheckedOutBy">
        <cmis:value>admin</cmis:value>
      </cmis:propertyString>
      <cmis:propertyId
propertyDefinitionId="cmis:versionSeriesCheckedOutId">
        <cmis:value>
        workspace://SpacesStore/b610bf02-c142-45d3-a941-
52cf71862b39</cmis:value>
      </cmis:propertyId>
      <cmis:propertyId propertyDefinitionId="cmis:versionSeriesId">
        <cmis:value>workspace://SpacesStore/39c86907-e21f-4165-9b90-
62c3bb7940c3</cmis:value>
      </cmis:propertyId>
      <cmis:propertyString propertyDefinitionId="cmis:versionLabel" />
      <cmis:propertyBoolean
propertyDefinitionId="cmis:isLatestVersion">
        <cmis:value>false</cmis:value>
      </cmis:propertyBoolean>
      …SNIP…
```

Now that the document is checked out and a PWC exists in the repository, there are a couple of different possibilities. One is that you might change your mind and decide to cancel the checkout. To do that, simply delete the PWC, like this:

```
curl -X DELETE -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/pwc/s/workspace:SpacesStore/i/b6
10bf02-c142-45d3-a941-52cf71862b39"
```

One way to verify the document is no longer checked out is to make sure it doesn't appear in the "checked out" collection. If you forgot what the URL is for the checked out collection, take a look at the repository information response at the start of these examples.

Another possibility is that you want to update the file and check it back in. Let's do that. Essentially, updating the PWC is just like creating a new document, except you'll use a PUT instead of a POST because you are updating the content stream of the PWC. Just like earlier, the updated document has to be Base64 encoded, then it can be PUT.

The Atom entry looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
xmlns:cmisra="http://docs.oasis-open.org/ns/cmis/restatom/200908/"
xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/">
    <title>sample-a (Working Copy).doc</title>
    <content type="application/msword">
0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAAAAAABAAAA
JwAAAAAAAAAEAAAKQAAAAEAAAD+////AAAAACYAAAD/////////////////////
…SNIP…
</content>
</entry>
```

The PUT invokes the change:

```
curl -X PUT -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/i/b977cad7-6cf4-4fe8-b306-
a8ccf6ffd783" -H "Content-Type: application/atom+xml" -d
@/Users/jpotts/testUpdateSampleA.atom.xml
```

The last step is to check in the working copy. To do that, PUT an empty Atom entry to the PWC with "checkin=true" and an optional checkin comment. The empty Atom entry looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom"/>
```

And the POST looks like this:

```
curl -X PUT -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/pwc/i/b977cad7-6cf4-4fe8-b306-
a8ccf6ffd783?checkin=true&checkinComment=minor%20changes" -H "Content-
Type: application/atom+xml" -d
@/Users/jpotts/testCheckinSampleA.atom.xml
```

Querying

As discussed earlier, the CMIS specification provides a relational-like query mechanism for CMIS repositories. Each type is like a table, each object is like a row, and each property is like a column in a relational database. CMIS query syntax is based on a subset of SQL-92 so it should be familiar to most developers.

Let's start out with a simple query that selects all instances of sc:whitepaper. All queries are wrapped in cmis:query XML and posted to the query URL. Here's the CMIS XML required to produce this simple query:

```
<cmis:query xmlns:cmis="http://docs.oasis-
open.org/ns/cmis/core/200908/">
  <cmis:statement>
    <![CDATA[SELECT * FROM sc:whitepaper]]>
  </cmis:statement>
  <cmis:skipCount>0</cmis:skipCount>
  <cmis:maxItems>5</cmis:maxItems>
</cmis:query>
```

Posting the query, as such:

```
curl -X POST -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/queries" -H "Content-Type:
application/cmisquery+xml" -d @/Users/jpotts/testTypeQuery.cmis.xml
```

Returns a feed of entries in which each entry is a query result:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
xmlns:app="http://www.w3.org/2007/app" xmlns:cmisra="http://docs.oasis-
open.org/ns/cmis/restatom/200908/" xmlns:cmis="http://docs.oasis-
open.org/ns/cmis/core/200908/" xmlns:alf="http://www.alfresco.org"
xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/">
  <author>
    <name>admin</name>
  </author>
  <generator version="3.2.0 (r2 @build-number@)">Alfresco
(Community)</generator>
  <icon>
  http://localhost:8080/alfresco/images/logo/AlfrescoLogo16.ico</icon>
  <id>urn:uuid:resultset</id>
  <link rel="service"
  href="http://localhost:8080/alfresco/s/cmis" />
  <link rel="self"
href="http://localhost:8080/alfresco/s/cmis/queries?alf_ticket=TICKET_4
4740de7943f7bab5be61f787c62660ff3008299" />
  <link rel="first"
href="http://localhost:8080/alfresco/s/cmis/query?q=SELECT%20*%20FROM%2
0sc:whitepaper&amp;skipCount=0&amp;maxItems=5"
  type="application/atom+xml;type=feed" />
  <title>Result set for SELECT * FROM sc:whitepaper</title>
  <updated>2009-11-17T16:02:58.040-06:00</updated>
  <opensearch:totalResults>2</opensearch:totalResults>
  <opensearch:startIndex>0</opensearch:startIndex>
  <opensearch:itemsPerPage>5</opensearch:itemsPerPage>
  <cmisra:numItems>2</cmisra:numItems>
```

```
  <entry>
    <author>
      <name>admin</name>
    </author>
    <content type="application/msword"
src="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/39
c86907-e21f-4165-9b90-62c3bb7940c3/content.doc" />
    <id>urn:uuid:39c86907-e21f-4165-9b90-62c3bb7940c3</id>
    …SNIP…
    <title>sample-a.doc</title>
    <updated>2009-11-16T23:11:20.793-06:00</updated>
    <alf:icon>
    http://localhost:8080/alfresco/images/filetypes/doc.gif</alf:icon>
    <cmisra:object>
      <cmis:properties>
        <cmis:propertyInteger
propertyDefinitionId="cmis:contentStreamLength">
          <cmis:value>22528</cmis:value>
        </cmis:propertyInteger>
        <cmis:propertyId propertyDefinitionId="cmis:objectTypeId">
          <cmis:value>D:sc:whitepaper</cmis:value>
        </cmis:propertyId>
        <cmis:propertyId propertyDefinitionId="cmis:objectId">
          <cmis:value>workspace://SpacesStore/39c86907-e21f-4165-9b90-
62c3bb7940c3;2.0</cmis:value>
        </cmis:propertyId>
        <cmis:propertyDateTime
propertyDefinitionId="cmis:lastModificationDate">
          <cmis:value>2009-11-16T23:11:20.793-06:00</cmis:value>
        </cmis:propertyDateTime>
        …SNIP…
      </cmis:properties>
    </cmisra:object>
  </entry>
  <entry>
    <author>
      <name>admin</name>
    </author>
    <content type="application/pdf"
src="http://localhost:8080/alfresco/s/cmis/s/workspace:SpacesStore/i/77
7aad53-8b18-4f1f-80f6-e420936b3def/content.pdf" />
    <id>urn:uuid:777aad53-8b18-4f1f-80f6-e420936b3def</id>
    …SNIP…
    <title>sample-b.pdf</title>
    <updated>2009-11-17T15:45:19.497-06:00</updated>
    <alf:icon>
    http://localhost:8080/alfresco/images/filetypes/pdf.gif</alf:icon>
    <cmisra:object>
      <cmis:properties>
        <cmis:propertyInteger
propertyDefinitionId="cmis:contentStreamLength">
          <cmis:value>117248</cmis:value>
        </cmis:propertyInteger>
        <cmis:propertyId propertyDefinitionId="cmis:objectTypeId">
          <cmis:value>D:sc:whitepaper</cmis:value>
        </cmis:propertyId>
        …SNIP…
      </cmis:properties>
    </cmisra:object>
```

```
  </entry>
</feed>
```

I've trimmed the resulting feed heavily. Each entry should be very familiar to you by now. For the rest of this section I will omit the response entirely.

It's important to note that the name (of a property, of a type, etc.) used in a query may be different than the identifier you've seen so far. For example, the CMIS object type ID for whitepaper is "D:sc:whitepaper". But the query we just executed used "sc:whitepaper". When you look at the result of a getType call you can see where these are specified:

```
…SNIP…
<cmis:id>D:sc:whitepaper</cmis:id>
<cmis:localName>whitepaper</cmis:localName>
<cmis:localNamespace>http://www.someco.com/model/content/1.0</cmis:loca
lNamespace>
<cmis:displayName>Someco Whitepaper</cmis:displayName>
<cmis:queryName>sc:whitepaper</cmis:queryName>
…SNIP…
```

Properties work exactly the same way, although in this case, the id matches the query name:

```
…SNIP…
<cmis:propertyStringDefinition>
<cmis:id>sc:campaign</cmis:id>
<cmis:localName>campaign</cmis:localName>
<cmis:localNamespace>http://www.someco.com/model/content/1.0</cmis:loca
lNamespace>
<cmis:displayName>sc:campaign</cmis:displayName>
<cmis:queryName>sc:campaign</cmis:queryName>
<cmis:propertyType>string</cmis:propertyType>
<cmis:cardinality>multi</cmis:cardinality>
<cmis:updatability>readwrite</cmis:updatability>
<cmis:inherited>true</cmis:inherited>
<cmis:required>false</cmis:required>
<cmis:queryable>true</cmis:queryable>
<cmis:orderable>false</cmis:orderable>
<cmis:openChoice>false</cmis:openChoice>
 …SNIP…
```

The table below shows several additional query examples.

CMIS Query	What it does
`SELECT cmis:name FROM sc:whitepaper where contains('sample')`	Select the name of every whitepaper with "sample" somewhere in their text.
`SELECT cmis:name, Score() as relevance FROM sc:whitepaper where contains('sample') order by relevance DESC`	Select the name of whitepapers with "sample" somewhere in their text, ordered by full-text relevance, descending.
`SELECT cmis:name from sc:whitepaper`	Select whitepapers with the

`where sc:isActive = true`	isActive flag set to true (which happens to be a property defined by an aspect).
`SELECT cmis:name from sc:whitepaper where not(sc:isActive = true)`	Select whitepapers with the isActive flag not set to true. This will return whitepapers where the property is not true as well as whitepapers where the property is unset.
`SELECT cmis:name from sc:marketingDoc where any sc:campaign in ('Social Shopping')`	Select instances of Marketing Documents and their children where the multi-value property, sc:campaign, contains the value "Social Shopping" (note that Whitepaper is a child type of Marketing Document).
`SELECT cmis:name,sc:published from sc:whitepaper where sc:published >= '2009-11-10T00:00:00.000-06:00' and sc:published < '2009-11-18T00:00:00.000-06:00'`	Select whitepapers published (sc:published, a property of the sc:webable aspect) on or after November 10, 2009 and before 18, 2009.
`SELECT cmis:name from cmis:document where in_folder('workspace://SpacesStore/3935ce21-9f6f-4d46-9e22-4f97e1d5d9d8') and contains('contract')`	Select all content in the Sales folder containing the word "contract".
`SELECT cmis:name from cmis:document where in_tree('workspace://SpacesStore/3935ce21-9f6f-4d46-9e22-4f97e1d5d9d8') and contains('contract') and cm:description like "%sign%"`	Select all content in the Sales folder or any of its descendant folders containing the word "contract" and a description like "%sign%".

There are a couple of things to note about this last query. First, not all CMIS repositories support searching both metadata and full-text content in the same query. A CMIS repository declares how it handles search in the capabilities listed as part of the repository information response. For example, Alfresco supports metadata and full-text queries simultaneously, so its capability looks like this:

```
<cmis:capabilityQuery>bothcombined</cmis:capabilityQuery>
```

Other possible values for this capability include:

- none
- metadataonly
- fulltextonly

- bothseparate
- bothcombined

The second thing to notice is that the piece of metadata being searched is the cm:description property. If you go look at the getType response for cmis:document you won't see cm:description. But, if you've worked with Alfresco you know it's there, and that it is displayed as the "summary" in the Alfresco Explorer client. CMIS doesn't know about it because the description is part of the cm:titled aspect and CMIS doesn't support aspects. Queries are the only piece of Alfresco's CMIS implementation where some form of aspect support is provided. Clearly, lack of aspect support severely limits the usefulness of CMIS for many Alfresco implementations, so hopefully full support for aspects will be added in the near future.

Working with permissions & allowable actions

You've already seen several examples of optional "capabilities" of a CMIS repository. Access Control Lists (ACLs) are another example. According to the specification, the possible values for "capabilityACL" are:

- none: The repository does not support any CMIS ACL services
- discover: The repository supports discovery of ACLs through CMIS
- manage: The repository supports the management of ACLs through CMIS

In the daily dev build I am working with, Alfresco reports "none" for this capability. If I were building an application on top of Alfresco using 100% CMIS I'd have to figure out another way to set ACLs on my objects. Maybe I'd use a rule to set the permissions. Or maybe I'd write a custom web script. The point is there are going to be things you just can't get done through CMIS that you will have to work around.

Even though Alfresco won't let me manage the ACL of an object, I can still get some idea of what I'm allowed to do. The underlying permissions on an object are mapped to a set of "allowable actions" or things the current user is allowed to do to a given object. The set of allowable actions is defined by CMIS and they are queryable.

For example, let's use the web client to tweak the ACL on the Whitepapers folder object, and then observe the changes to the allowable actions for that object.

First, to get a "before" picture, I'll do a GET on the folder with the "includeAllowableActions" flag set to true:

```
curl  -uadmin:admin
"http://localhost:8080/alfresco/s/cmis/p/Someco/Marketing/Whitepapers?
includeAllowableActions=true"
```

The result is an Atom entry, as usual, with a new element, cmis:allowableActions tacked on to the end:

```
…SNIP…
<cmis:allowableActions>
  <cmis:canDeleteObject>true</cmis:canDeleteObject>
```

```
  <cmis:canUpdateProperties>true</cmis:canUpdateProperties>
  <cmis:canGetFolderTree>true</cmis:canGetFolderTree>
  <cmis:canGetProperties>true</cmis:canGetProperties>
  <cmis:canGetObjectRelationships>true</cmis:canGetObjectRelationships>
  <cmis:canGetObjectParents>true</cmis:canGetObjectParents>
  <cmis:canGetFolderParent>true</cmis:canGetFolderParent>
  <cmis:canGetDescendants>true</cmis:canGetDescendants>
  <cmis:canMoveObject>true</cmis:canMoveObject>
  <cmis:canApplyPolicy>false</cmis:canApplyPolicy>
  <cmis:canGetAppliedPolicies>false</cmis:canGetAppliedPolicies>
  <cmis:canRemovePolicy>false</cmis:canRemovePolicy>
  <cmis:canGetChildren>true</cmis:canGetChildren>
  <cmis:canCreateDocument>true</cmis:canCreateDocument>
  <cmis:canCreateFolder>true</cmis:canCreateFolder>
  <cmis:canCreateRelationship>true</cmis:canCreateRelationship>
  <cmis:canCreatePolicy>false</cmis:canCreatePolicy>
  <cmis:canGetACL>false</cmis:canGetACL>
  <cmis:canApplyACL>false</cmis:canApplyACL>
</cmis:allowableActions>
…SNIP…
```

The current user, admin, has full reign over this folder. Now I'll update the ACL of the folder to make tuser1 a consumer (read-only access), then I'll get a ticket as tuser1 and re-execute the GET. The result is shown below, with the values that have changed in bold:

```
…SNIP…
<cmis:allowableActions>
  <cmis:canDeleteObject>false</cmis:canDeleteObject>
  <cmis:canUpdateProperties>false</cmis:canUpdateProperties>
  <cmis:canGetFolderTree>true</cmis:canGetFolderTree>
  <cmis:canGetProperties>true</cmis:canGetProperties>
  <cmis:canGetObjectRelationships>true</cmis:canGetObjectRelationships>
  <cmis:canGetObjectParents>true</cmis:canGetObjectParents>
  <cmis:canGetFolderParent>true</cmis:canGetFolderParent>
  <cmis:canGetDescendants>true</cmis:canGetDescendants>
  <cmis:canMoveObject>false</cmis:canMoveObject>
  <cmis:canApplyPolicy>false</cmis:canApplyPolicy>
  <cmis:canGetAppliedPolicies>false</cmis:canGetAppliedPolicies>
  <cmis:canRemovePolicy>false</cmis:canRemovePolicy>
  <cmis:canGetChildren>true</cmis:canGetChildren>
  <cmis:canCreateDocument>false</cmis:canCreateDocument>
  <cmis:canCreateFolder>false</cmis:canCreateFolder>
  <cmis:canCreateRelationship>false</cmis:canCreateRelationship>
  <cmis:canCreatePolicy>false</cmis:canCreatePolicy>
  <cmis:canGetACL>false</cmis:canGetACL>
  <cmis:canApplyACL>false</cmis:canApplyACL>
</cmis:allowableActions>
…SNIP…
```

So in Alfresco's current implementation, you can't retrieve a list of who has what permissions, but at least you can tell what the current user is able to do to a given object.

Client examples

One of the nice things about CMIS is that you don't really need a client API at all if you don't want one. Your application can simply make calls to the repository over HTTP and parse the responses.

JavaScript

Here's a simple server-side JavaScript example that uses the parse-it-yourself approach. This is a snippet from an Alfresco Surf web script controller that is used as part of a CMIS browser component:

```
…SNIP…
// retrieve the feed
var connector = remote.connect("alfresco");
var feed = connector.get("/cmis/p/" + path + "/children");
var xml = loadFeed(feed);

// set up model
model.title = xml.*::title.toString();
var items = new Array();
for each (entry in xml.*::entry)
{
  var item = { };
  item["title"] = entry.*::title.toString();
  item["icon"] = entry.*::icon.toString();
  item["id"] = entry.*::id.toString().substring(9);
  item["nodeRef"] = "workspace://SpacesStore/" + item["id"];
  item["url"] = context.linkBuilder.object(item["nodeRef"]);
  items.push(item);
}
model.items = items;
```

In this case, the loadFeed function contains a call to the E4X library (http://www.ibm.com/developerworks/library/ws-ajax1/) that loads the Atom XML returned by the GET into a DOM object. The for-each iterates over the Atom entries, snags values out of the XML, and puts them in the model. The view (not shown) then formats the model data into a nice list of clickable folders and documents.

Apache Abdera

Parsing Atom feeds your self can get tedious. And why should you do it when there are Atom parsers available for most common languages? In Java, one such library is called **Apache Abdera** (http://abdera.apache.org/). Abdera, an Apache Incubator project, is an open source implementation of the Atom Sydnication Format and Atom Publishing Protocol specifications. It includes both server-side and client-side libraries. The client side libraries can be used to retrieve, parse, and create Atom feeds. The server side libraries would be used if you wanted to implement your own Atom server, which doesn't really make sense in this context.

For example, here's a snippet below showing how to iterate over the entries in an Atom feed using the Abdera parser. You should recognize the URL from the earlier examples:

```
…SNIP…
Parser parser = abdera.getParser();
URL url = new
URL("http://localhost:8080/alfresco/s/cmis/p/Someco/children");
String encoding =
     new BASE64Encoder().encode(getUserNamePassword().getBytes());
URLConnection uc = url.openConnection();
uc.setRequestProperty ("Authorization", "Basic " + encoding);
InputStream content = (InputStream)uc.getInputStream();
Document<Feed> doc = parser.parse(content);
Feed feed = doc.getRoot();
System.out.println(feed.getTitle());
for (Entry entry : feed.getEntries()) {
   System.out.println("\t" + entry.getTitle());
}
assertTrue(feed.getEntries().size() > 0);
…SNIP…
```

So Abdera understands Atom, and that's helpful, but that library doesn't know anything about the CMIS domain. Why does that matter? If you're just doing a GET and parsing the resulting feed, it really doesn't. But when you start creating new Atom entries that contain CMIS extensions, it starts to get a little cumbersome. Here's another Abdera example that creates a simple Atom entry representing a CMIS folder and then uses the Abdera client to post it:

```
…SNIP…
// Build the entry
Entry entry = abdera.newEntry();
entry.setTitle("TestAbderaFolder");
entry.setSummary("test Abdera folder");
```

```
ExtensibleElement objElement =
     (ExtensibleElement) entry.addExtension(NS_CMIS_RESTATOM, "object",
CMISRA);
ExtensibleElement propsElement =
     objElement.addExtension(NS_CMIS_CORE, "properties", CMIS);
ExtensibleElement stringElement =
     propsElement.addExtension(NS_CMIS_CORE, "propertyId", CMIS);
stringElement.setAttributeValue("propertyDefinitionId",
     "cmis:objectTypeId");
Element valueElement =
     stringElement.addExtension(NS_CMIS_CORE, "value", CMIS);
valueElement.setText("cmis:folder");

// Post it
AbderaClient client = new AbderaClient();
String encoding = new BASE64Encoder().encode("admin:admin".getBytes());

RequestOptions options = new RequestOptions();
options.setHeader("Authorization", "Basic " + encoding);
ClientResponse response = null;
response =
     client.post(
          "http://localhost:8080/alfresco/s/cmis/p/Someco/children",
          entry,
          options);
…SNIP…
```

To write this code you have to know the structure of the CMIS extensions to Atom and you're working with XML concepts, not CMIS concepts. What you'd really like to be able to do is say something like "createFolder", but that's a higher level concept than what Atom knows or cares about. That's where Chemistry comes in.

Apache Chemistry

Apache Chemistry (http://incubator.apache.org/chemistry/), also an incubator project, is an open source CMIS implementation and it includes:

- A server, or, more accurately, a layer that makes any JCR-compliant repository a CMIS repository
- A client
- A Technology Compatibility Kit (TCK) that can be used to run tests against a CMIS repository to verify compliance to the spec

Here are a few short JUnit tests I wrote that use the Chemistry client:

```
…SNIP…
public void testGetRepositoryInfo() {
  ContentManager contentManager = new APPContentManager(CMIS_REPO_URL);
  contentManager.login(USER, PASS);
  Repository repository = contentManager.getDefaultRepository();
  RepositoryInfo repositoryInfo = repository.getInfo();
  System.out.println("Product:" + repositoryInfo.getProductName());
  System.out.println("Vendor:" + repositoryInfo.getVendorName());
  System.out.println("Version:" +
     repositoryInfo.getVersionSupported());
}
```

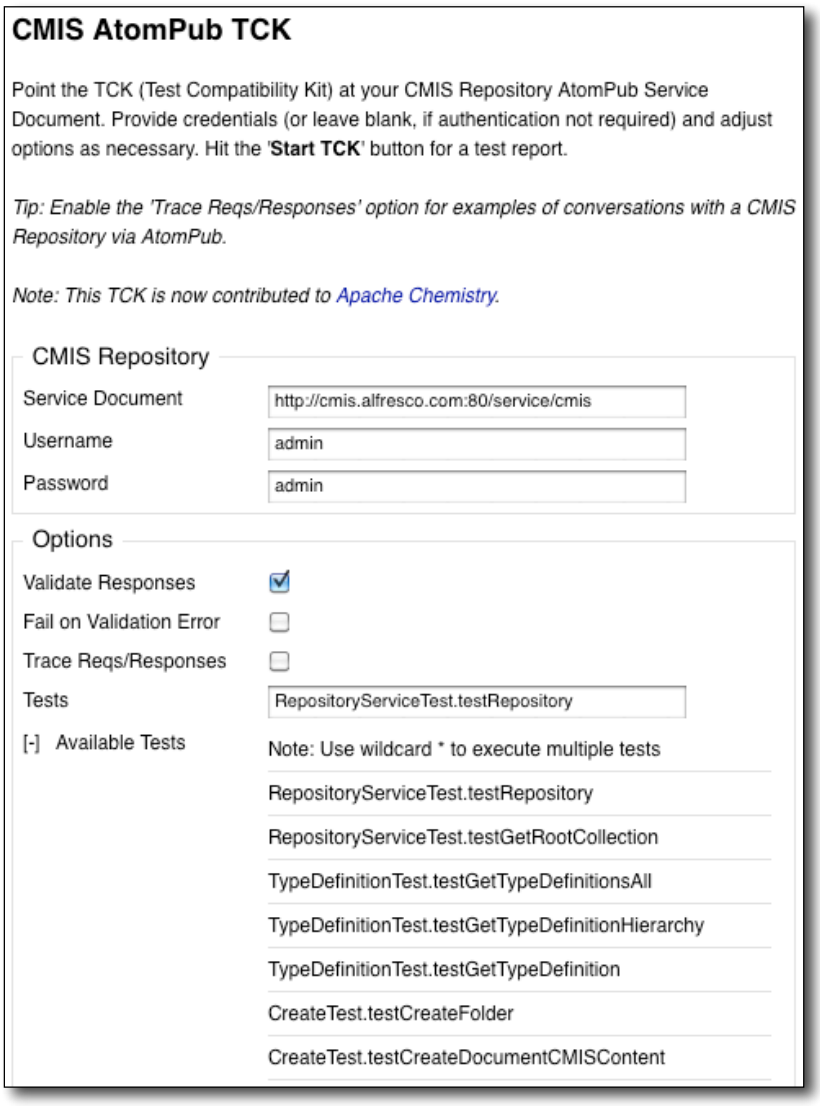
```
public void testGetTypes() {
  ContentManager contentManager = new APPContentManager(CMIS_REPO_URL);
  contentManager.login(USER, PASS);
  Repository repository = contentManager.getDefaultRepository();
  Collection<Type> types = repository.getTypes("cmis:document");
  for (Type type : types) {
      System.out.println("Type id:" + type.getId());
  }
}

public void testGetRootChildren() {
  APPContentManager contentManager =
     new APPContentManager(CMIS_REPO_URL);
  contentManager.login(USER, PASS);
  Repository repository = contentManager.getDefaultRepository();
  RepositoryInfo repositoryInfo = repository.getInfo();
  ObjectId rootObjId = repositoryInfo.getRootFolderId();
  APPConnection connection =
      (APPConnection) repository.getConnection(null);
  CMISObject rootObj = connection.getObject(rootObjId);
  System.out.println("Root object id:" + rootObj.getId());
  List<ObjectEntry> children = connection.getFolderTree(rootObjId, 1,
null, false);
  for (ObjectEntry entry : children) {
      System.out.println(entry.getValue("cmis:name"));
  }
}
…SNIP…
```

Unfortunately, the Chemistry client isn't all there yet. For example, many of the "write" functions (checkout(), checkin(), and cancelCheckout(), to name just a few) aren't implemented yet. You'll notice in the snippets above, I couldn't always use the interfaces but instead had to use implementation classes like APPContentManager and APPConnection, which felt bad. And I couldn't get the types or children tests to succeed against Alfresco. I had to run them against the test server that comes with Chemistry. That seems like a bad thing when both of those servers declare support for CMIS 1.0. The takeaway here is that if you are writing a Java application that's using CMIS, you may need to donate some time to the Chemistry project, or choose to go your own way.

Apache Chemistry TCK Client

There is a second, separate client that's part of the Apache Chemistry TCK. This may seem strange—Why wouldn't the TCK leverage the existing Chemistry client? The two clients developed independently of each other. Alfresco developed the TCK client and donated it soon after Day and Nuxeo formed the Chemistry project. It seems to me that the TCK client and the partially-implemented Chemistry client need to be merged. I don't know what the plan is for that, but I assume as Nuxeo makes their implementation read/write, the Chemistry client will get cleaned up. At that point the TCK can make the switch if it makes sense.

The TCK client is not production-ready either, but it does support read/write and it is being used. The TCK uses it to run compliance tests and others have used it successfully in their own test applications (see next section).



If you want to see the TCK in action, go to http://cmis.alfresco.com, scroll to the CMIS AtomPub TCK section, enable the "Trace Reqs/Responses" checkbox, and then click "Start TCK". It's actually kind of a handy little debugging tool.

Exploring Abdera, Chemistry, and the TCK Client on your own

If you want to play around with Java and CMIS, Alfresco has something cool for you. It's a Maven Archetype that gives you a couple of starter projects to work with. Here's how it works, assuming you have Apache Maven 2.06 or higher installed:

1. Go to a convenient working directory.
2. Invoke Maven, telling it to generate a new project using an archetype: `mvn archetype:generate -DarchetypeCatalog=http://maven.alfresco.com/nexus/content/repositories/releases`
3. Maven will give you a list of archetypes from the catalog to choose from. You're looking for "cmis-master-labs-archetype" which was

number 4 at the time of this writing. When prompted, type "4" and hit enter.
4. When prompted for a group ID, enter a package structure, like "com.optaros".
5. When prompted for the artifact ID, enter something like "cmis-article".
6. Take the default version and package.
7. Confirm the setup. Type "Y" and hit enter. Maven will set up a project under a directory with a name that matches the artifact ID.
8. If you are using your own test Alfresco server (instead of http://cmis.alfresco.com) edit the pom.xml file and change the "cmisServiceUrl" reference to your server (e.g., http://localhost:8080/alfresco/s/api/cmis).
9. Type "mvn install" and hit enter. Maven will download all of the dependencies the project needs, do a build, and run some tests.

If everything goes according to plan, you'll end up with two apps. One is a CMIS importer example and the other is a simple web application that runs CMIS queries against the repository.

To run the CMIS importer example:
1. Switch to the cmis-lab-atompub-binding directory
2. Type "mvn test" and hit enter.

The import is implemented as a JUnit test. The application uses a class called CMISClientUtils which wraps the Chemistry TCK client discussed in the previous section. When the tests complete successfully you'll have a new test folder in the root of your repository.

To run the CMIS query web application:
1. Switch to the cmis-query-webapp directory
2. Type "mvn jetty:run-exploded" and hit enter.
3. Use your browser to go to http://localhost:8081/cmis-query-webapp".

The project also includes a groovy console. The idea was to use groovy (http://groovy.codehaus.org/) scripts to make calls to the CMIS repository. Unfortunately, in the release I checked out there's a null pointer exception that keeps it from working.

For more information, including a link to the CMIS training lab material for which this archetype was originally developed, go to http://wiki.alfresco.com/wiki/CMIS_Maven_Toolkit.

Integrations based on CMIS

A number of integrations that rely on CMIS have already started to pop up. These might be useful in your projects or might at least make decent examples:

- Drupal CMIS API (http://drupal.org/project/cmis) (PHP)
- Drupal CMIS Alfresco (http://www.drupal.org/project/cmis_alfresco
-) (PHP)
- Drupal CMIS KnowledgeTree (http://drupal.org/project/cmis_knowledgetree) (PHP)
- CMIS Explorer (http://code.google.com/p/cmis-explorer/) (Flex)

- CMIS Spaces (http://code.google.com/p/cmisspaces/) (Flex)

See the Alfresco CMIS wiki page (http://wiki.alfresco.com/wiki/CMIS) for a list of other known clients and tools.

Conclusion

This article has given you an introduction to CMIS. We started with a brief overview of the specification and a brief listing of open source CMIS repositories. Then we jumped right in to the CMIS AtomPub Binding by using curl to send some GETs, POSTs, PUTs, and DELETEs against Alfresco. The examples showed how to do CRUD functions around folders, documents, and relationships, then moved on to CMIS query examples.

The last part focused on different libraries you can use to potentially make your life easier when working with CMIS, including Abdera, Chemistry, and the Chemistry TCK. If you've got Apache Maven installed, you can create a project using the "cmis-master-labs-archetype".

Hopefully this has inspired you to learn more about the CMIS spec and has sparked a few ideas around how you might incorporate CMIS into your next project.

Where to find more information

- The complete source code that accompanies this article is available from http://ecmarchitect.com at http://ecmarchitect.com/images/cmis-article-code.zip.
- If you are a developer who needs to get ramped up on the Alfresco platform, consider purchasing the Alfresco Developer Guide available at Packt Publishing (http://www.packtpub.com/alfresco-developer-guide/book), Amazon, and fine bookstores everywhere! ☺
- Richard McKnight has some introductory information on CMIS at his blog at http://www.oldschooltechie.com/blog/2009/11/23/introduction-cmis.
- The CMIS Technical Committee page at OASIS is at http://www.oasis-open.org/committees/cmis
- The CMIS specification can be downloaded as a PDF from http://docs.oasis-open.org/cmis/CMIS/v1.0/cd04/cmis-spec-v1.0.pdf
- The Apache Chemistry home page is http://incubator.apache.org/chemistry/.
- The Apache Abdera home page is http://abdera.apache.org/.

About the Author

Jeff Potts is Sr. Practice Director of the Enterprise Content Management Practice at Optaros, a leading Open Source and Next Generation Internet consultancy. Jeff has over fifteen years of experience implementing content management, collaboration, and other knowledge management technologies for a variety of Fortune 500 companies and is the author of the award-winning Alfresco Developer Guide (Packt, 2008). Jeff lives in Dallas, Texas with his wife and two kids. Read more at ecmarchitect.com.

Update History

- 12/7/2009: Updated the Maven archetype steps to reflect the new home of the Alfresco Maven repository.