Version 0.6

Content Management Interoperability Services

Unified Search Proposal

Versions

Version	Author	Date	Modifications
0.1	Gregory Melahn, IBM	02/09/2009	• N/A
0.2	Gregory Melahn, IBM	02/11/2009	• Added changeToken and nextChangeToken and more notes about how the events are ordered in the response set.
0.3	Gregory Melahn, IBM	02/17/2009	• Results of 2/17 review meeting
0.4	Gregory Melahn, IBM	02/24/2009	• Results of 2/24 review meeting
0.5	Gregory Melahn, IBM	03/02/2009	• Added REST and WS binding example
0.6	Gregory Melahn, IBM	03/02/2009	• Some cleanup

Table of Contents

INTRODUCTION

CMIS needs to allow repositories to expose what information inside the repository has changed in an efficient manner for applications of interest, like search crawlers, to facilitate incremental indexing of a repository.

In theory, a search crawler could index the content of a CMIS repository by using the navigation mechanisms already defined as part of the proposed specification. For example, a crawler engine could start at the root collection and, using the REST bindings, progressively navigate through the folders, get the document content and metadata, and index that content. It could use the CMIS date/time stamps to more efficiently do this by querying for documents modified since the last crawl.

But there are problems with this approach. First, there is no mechanism for knowing what has been deleted from the repository, so the indexed content would contain 'dead' references. Second, there is no standard way to get the access control information needed to filter the search results so the search consumer only sees the content (s) he is supposed to see. Third, each indexer would solve the crawling of the repository in a different way (for example, one could use query and one could use navigation) causing different performance and scalability characteristics that would be hard to control in such system. Finally, the cost of indexing an entire repository can be prohibitive for large content, or content that changes often, requiring support for incremental crawling and paging results.

STATUS

This document is a proposal for a modification to the draft CMIS specification.

OVERVIEW

The new service described in this proposal will allow search crawlers to efficiently navigate a CMIS repository.

DOMAIN MODEL

The following new services are proposed under Search / Discovery

getContentChanges

Add *String changeToken* and Boolean *changesIncomplete* to the output of *getRepositoryInfo*.

SCHEMA ADDITIONS

```
<!--  Unified Search proposal -->
  <xs:simpleType name="enumTypeOfChanges">
     <xs:restriction base="xs:string">
        <!--  content with a new ID has been created -->
        <xs:enumeration value="created" />
        <!--  content with an existing ID has been modified -->
        <xs:enumeration value="updated" />
        <!--  content with an existing ID has been deleted -->
        <xs:enumeration value="deleted" />
        <!--  content with an existing ID has had its security policy changed
-->
        <xs:enumeration value="security" />
    </xs:restriction>
  <xs:simpleType name="enumCapabilityChanges">
     <xs:restriction base="xs:string">
        <xs:enumeration value="none" />
        <xs:enumeration value="includeACL" />
        <xs:enumeration value="includeProperties" />
        <xs:enumeration value="includeFolders" />
        <xs:enumeration value="includeDocuments" />
        <xs:enumeration value="includeRelationships" />
        <xs:enumeration value="includePolicies" />
        <xs:enumeration value="all" />
     </xs:restriction>
  </xs:simpleType>
  <xs:element name="changedObject" type="cmis:cmisChangedObjectType" />
  <xs:complexType name="cmisChangedObjectType">
    <xs:complexContent>
       <xs:extension base="cmis:cmisObjectType">
         <xs:sequence>
            <xs:element name="changeType" type="cmis:enumTypeOfChanges" />
            <xs:element name=”changeTime” type=” xs:DateTime” />
         </xs:sequence>
       </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

RepositoryCapability will be updated to include…

1. **enumCapabilityChanges**

2. a new Boolean flag, **changesIncomplete**, to signal that the getContentChanges service may not be able to answer all the changes that have occurred. For example, if a repository uses a transaction log as the basis for implementing getContentChanges and the transaction log has been truncated for space reasons, then the changesIncomplete flag should return true, otherwise false.

QUERY SERVICES

getContentChanges

Description	Gets a list of content changes. This service is intended to be used by search crawlers or other applications that need to efficiently understand what has changed in the repository.
Inputs	• ID repositoryId: Repository Id • (Optional) String changeToken: Opaque, verifiable, stable token generated by a previous getContentChanges service call or from getRepositoryInfo. This provides the starting point for this service call to answer changed items. • (Optional) Int maxItems: max number of items to be returned. • (Optional) Boolean includeACL: includes ACL (if ACL becomes part of CMIS) • (Optional) Boolean includeProperties: includes the properties for applicable objects
Outputs	• a set of CmisChangedObjectType resultSet: a set containing information about the content created, updated or deleted since the last service call. • String changeToken: a starting point for a subsequent service call. When used on a subsequent service call the set returned by that call will include the last item in the set returned on this service call (i.e. the sets will overlap by that item). • Boolean hasMoreItems: when true, signals that there are still more changes available
Exceptions	• **Common Exceptions** • **ExpiredChangeTokenException**: the changeToken has expired and cannot be used to locate the starting point for this service call. This can be caused by, for example, conditions where there are more changes than the repository can store for the length of time required by service consumers. This would indicate the need for some administrator action such as reconfiguring the crawling frequency or the amount of time that events are stored by the repository.
Notes	• It is a repository choice as to what type of content to include in the results (documents, folders, policies or relationships). The capabilities should be enumerated in getRepositoryInfo. • The results should be in a flat paged list • For created or updated objects the properties are included in the returned list if includeProperties is true and the repository capabilities support including properties in the change set • For created, updated or security changed objects the access rights are included in the returned list if includeACL is true and the repository capabilities support including ACL's in the change set • For deleted objects, properties or access rights are not returned • The content-stream is not returned so a search crawler would need to additionally fetch the document content using getContentStream to index that content • The definition of the authority needed to call this service is repository specific. A repository should throw a **PermissionDeniedException** exception if this constraint is violated. • If the input changeToken is not recognizable (meaning for example it was not generated by this repository), the repository should throw a **InvalidArgumentException** • If an input filter is provided that is not consistent with the capabilities of the repository (for example, if includeACL is provided, but the repository is not capable of returning access control information), the repository should throw a **InvalidArgumentException** • If changeToken is not provided, then this is assumed to be a service call to retrieve the first maxItems number of items. Note that the repositoryInfo flag, changesIncomplete, can be inspected to determine if getContentChanges is able to answer all the content in the repository. If changesIncomplete is true, then the caller should not expect that getContentChanges can answer all the content, but should

	instead rely on using a query if a complete list of content is needed. • The order in which the CmisChangedObjectType instances appear in the output set is the order in which the events happened, oldest first, for each instance of the content described by CmisChangedObjectType. For example, if an item was created at time t, updated at time t+1 and then deleted at time t+2, the order in which the events appear in the output set is *create at t*, *update at t+1*, *delete at t+2*, though these events would not necessarily be grouped together in a page of responses or even in the same response page. This is done so that the service consumer can process the events in the order they appear in the result set without having to remember what events it has already processed for an object. • A repository MAY treat a filing or unfiling operation as change to the document or the folder, or both. • If no "maxItems" value is provided, then the Repository will determine an appropriate number of items to return. How the Repository determines this value is repository-specific and opaque to CMIS. • The fact that less than maxItems items were returned does not, by itself, signify there are no more changes available. Rather, hasMoreItems should be inspected to determine whether there are more changes available.

REST BINDING

ATOM entries will be returned, one for each content item that has changed. The entry will also contain enough information to allow the search engine index the content so as to allow it to trim the results based on access rights, when a search is later executed.

A new collection of type 'changes' will be created. That collection will expose a feed of CmisChangedObjectTypes and will support paging.

getContentChanges

Description	Gets a list of content changes. This service is intended to be used by search crawlers or other applications that need to efficiently understand what has changed in the repository.
Arguments	Headers: None HTTP Arguments: changeToken, maxItems, includeProperties, includeACL

The client must POST to the CMIS *changes* collection like the example below.

```
POST /cmis/main HTTP/1.1
Host: example.org
User-Agent: Thingio/1.0
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Type: application/cmisrequest+xml;type=changes
```

```
Content-Length: 0
Accept: application/atom+xml;type=feed
```

Response:

```
HTTP/1.1 200 OK
Date: Fri, 29 March 2009 12:12:11 GMT
Content-Type: application/atom+xml;type=feed
Content-Length: nnn
Location: http://cmis.example.org/cmis/changes

<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:cmis="http://www.cmis.org/CMIS/1.0">
   <title>Changes</title>
   <link href="http://cmis.example.org/cmis/changes"/>
   <updated>2009-03-29T11:11:11Z</updated>
   <author>
       <name>CMIS Repository</name>
   </author>
   <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
   <cmis:changeToken>2da49c60-e389-31f9-c83D-0003434550ce7</cmis:changeToken>
   <cmis:hasMoreItems>false</cmis:hasMoreItems>
   <entry>
      <author>
        <name>John Doe</name>
      </author>
      <content type="text">See Stream</content>
      <id>Fd33394D-DC43-24BA-8326-AC454F683B7A</id>
      <published>2009-03-09T13:11:11Z</published>
      <summary type="html">&lt;p&gt;This is a document summary&lt;/p&gt;&lt;</summary>
      <title type="text">Foobar</title>
      <updated>2009-03-09T13:11:11Z </updated>
      <!-- a created object  -->
      <cmis:changedObject>
          <cmis:properties>
             <cmis:propertyId name="ObjectId">
               <cmis:value>{F4C339BD-DC4B-44BA-8426-A5C54D693B7A}</cmis:value>
             </cmis:propertyId>
             <cmis:propertyUri name="uri">
                 <cmis:value>http://cmis.example.com:8080/uri0101</cmis:value>
             </cmis:propertyUri>
             <cmis:propertyId name="ObjectTypeId">
                <cmis:value>document</cmis:value>
             </cmis:propertyId>
             <cmis:propertyString name="CreatedBy">
                <cmis:value>John Doe</cmis:value>
             </cmis:propertyString>
             <cmis:propertyDateTime cmis:name="CreationDate">
                <cmis:value>2009-03-09T13:11:11Z</cmis:value>
             </cmis:propertyDateTime>
             <cmis:propertyString cmis:name="LastModifiedBy">
                <cmis:value>John Doe</cmis:value>
             </cmis:propertyString>
             <cmis:propertyDateTime name="LastModificationDate">
                <cmis:value>2009-03-09T13:11:11Z</cmis:value>
             </cmis:propertyDateTime>
             <!-- . . . other properties from the object removed for brevity -->
           </cmis:properties>
         <cmis:changeType>created</cmis:changeType>
         <cmis:changeTime>2009-03-09T13:11:11Z</cmis:changeTime>
      </cmis:changedObject>
      <!-- a deleted object  -->
      <cmis:changedObject>
          <cmis:properties>
             <cmis:propertyId name="ObjectId">
               <cmis:value>{F4C339BD-DC4B-44BA-8426-A5C54D693B7A}</cmis:value>
             </cmis:propertyId>
           </cmis:properties>
         <cmis:changeType>deleted</cmis:changeType>
         <cmis:changeTime>2009-03-09T13:11:12Z</cmis:changeTime>
      </cmis:changedObject>
      <!-- an updated object  -->
      <cmis:changedObject>
          <cmis:properties>
```

```
            <cmis:propertyId name="ObjectId">
              <cmis:value>{F4C339BD-DC4B-44BA-8426-A5C54D693B7A}</cmis:value>
            </cmis:propertyId>
            <cmis:propertyUri name="uri">
                <cmis:value>http://cmis.example.com:8080/uri0101</cmis:value>
            </cmis:propertyUri>
            <cmis:propertyId name="ObjectTypeId">
               <cmis:value>document</cmis:value>
            </cmis:propertyId>
            <cmis:propertyString name="CreatedBy">
               <cmis:value>John Doe</cmis:value>
            </cmis:propertyString>
            <cmis:propertyDateTime cmis:name="CreationDate">
               <cmis:value>2009-03-09T13:11:11Z</cmis:value>
            </cmis:propertyDateTime>
            <cmis:propertyString cmis:name="LastModifiedBy">
               <cmis:value>John Doe</cmis:value>
            </cmis:propertyString>
            <cmis:propertyDateTime name="LastModificationDate">
               <cmis:value>2009-03-09T13:11:12Z</cmis:value>
            </cmis:propertyDateTime>
            <!-- . . . other properties from the object removed for brevity -->
          </cmis:properties>
        <cmis:changeType>updated</cmis:changeType>
        <cmis:changeTime>2009-03-09T13:11:12Z</cmis:changeTime>
     </cmis:changedObject>
  </entry>
</feed>
```

WEB SERVICES BINDING

getContentChanges

Description	Gets a list of content changes. This service is intended to be used by search crawlers or other applications that need to efficiently understand what has changed in the repository

```
<cmis:getContentChangesResponse>
  <!-- a created object  -->
  <cmis:changedObject>
    <cmis:properties>
        <cmis:propertyId cmis:name="ObjectId">
          <cmis:value> {F4C339BD-DC4B-44BA-8426-A5C54D693B7A}</cmis:value>
          </cmis:propertyId>
        <cmis:propertyUri cmis:name="Uri">
          <cmis:value>http://cmis.example.com:8080/uri0101</cmis:value>
          </cmis:propertyUri>
        <cmis:propertyId cmis:name="ObjectTypeId">
          <cmis:value>document</cmis:value>
          </cmis:propertyId>
         <cmis:propertyString name="CreatedBy">
           <cmis:value>John Doe</cmis:value>
         </cmis:propertyString>
         <cmis:propertyDateTime cmis:name="CreationDate">
           <cmis:value>2009-03-09T13:11:11Z</cmis:value>
```

```
        </cmis:propertyDateTime>
        <cmis:propertyString cmis:name="LastModifiedBy">
           <cmis:value>John Doe</cmis:value>
        </cmis:propertyString>
        <cmis:propertyDateTime name="LastModificationDate">
          <cmis:value>2009-03-09T13:11:11Z</cmis:value>
        </cmis:propertyDateTime>
        <!-- . . . other properties from the object removed for brevity -->
    </cmis:properties>
    <cmis:changeType>created</cmis:changeType>
    <cmis:changeTime>2009-03-09T13:11:12Z</cmis:changeTime>
  </cmis:changedObject>
  <!-- a deleted object  -->
  <cmis:changedObject>
      <cmis:properties>
         <cmis:propertyId name="ObjectId">
            <cmis:value>{F4C339BD-DC4B-44BA-8426-A5C54D693B7A}</cmis:value>
         </cmis:propertyId>
      </cmis:properties>
      <cmis:changeType>deleted</cmis:changeType>
      <cmis:changeTime>2009-03-09T13:11:12Z</cmis:changeTime>
  </cmis:changedObject>
  <!-- an updated object  -->
  <cmis:changedObject>
     <cmis:properties>
       <cmis:propertyId name="ObjectId">
          <cmis:value>{F4C339BD-DC4B-44BA-8426-A5C54D693B7A}</cmis:value>
       </cmis:propertyId>
       <cmis:propertyUri name="uri">
          <cmis:value>http://cmis.example.com:8080/uri0101</cmis:value>
       </cmis:propertyUri>
       <cmis:propertyId name="ObjectTypeId">
          <cmis:value>document</cmis:value>
       </cmis:propertyId>
       <cmis:propertyString name="CreatedBy">
          <cmis:value>John Doe</cmis:value>
       </cmis:propertyString>
       <cmis:propertyDateTime cmis:name="CreationDate">
          <cmis:value>2009-03-09T13:11:11Z</cmis:value>
       </cmis:propertyDateTime>
          <cmis:propertyString cmis:name="LastModifiedBy">
       <cmis:value>John Doe</cmis:value>
          </cmis:propertyString>
       <cmis:propertyDateTime name="LastModificationDate">
          <cmis:value>2009-03-09T13:11:12Z</cmis:value>
       </cmis:propertyDateTime>
     <!-- . . . other properties from the object removed for brevity -->
     </cmis:properties>
     <cmis:changeType>updated</cmis:changeType>
     <cmis:changeTime>2009-03-09T13:11:12Z</cmis:changeTime>
  </cmis:changedObject>
  <cmis:changeToken>2da49c60-e389-31f9-c83D-0003434550ce7</cmis:changeToken>
  <hasMoreItems>false</hasMoreItems>
</cmis:getContentChangesResponse>
```