Version 0.2

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	2/17/2009	• Results of 2/17 review meeting

Table of Contents

INTRODUCTION

CMIS has introduced a capability that allows repositories to expose what information inside the repository has changed in an efficient manner for applications of interest, like search crawlers, to leverage 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 navigate a CMIS repository.

DOMAIN MODEL

The following new services are proposed under Search / Discovery

getContentChanges

Add *String changeToken* 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:complexType name="cmisChangedObjectType">
    <xs:complexContent>
       <xs:extension base="cmis:cmisObjectType">
         <xs:sequence>
            <xs:element name="type" type="cmis:enumTypeOfChanges" />
            <xs:element name=”change_time” type=” xs:dateTime/>
         </xs:sequence>
       </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

RepositoryCapability will be updated to include enumCapabilityChanges.

QUERY SERVICES

getContentChanges

Description	Gets a list of content changes. This service is intended to be used by search crawlers or other applications who 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. This provides the starting point for this service call to answer changed items. • (Optional) Int maxItems: max # of items to be returned. If not provided, then the repository decides the value. • (Optional) Boolean includeACL – includes ACL (provided ACL is part of CMIS)

	• (Optional) Boolean includeProperties – includes the properties for applicable objects • (Optional) Boolean includeFolders – include folders in the output set • (Optional) Boolean includeDocuments – include documents in the output set • (Optional) Boolean includeRelationships – include relationships in the output set • (Optional) Boolean includePolicies – include policies in the output set
Outputs	• Result set containing information about the content created, updated or deleted since the last service call. • Output as a set of CmisChangedObjectType • changeToken: a starting point for a subsequent service call. When used on a ubsequent 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).
Exceptions	• **Common Exceptions** • **InvalidArgumentException**: the changeToken is not recognizable (meaning for example it was not generated by this repository). This would likely indicate a development time error. • **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 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 • The result set will include folders, documents, relationships and/or policies, depending on the repository capabilities and the values of the includeFolders, includeDocuments, includeRelationships and includeProperties flags • 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 can specify a **RuntimeException** or **ConstraintViolationException** exception if this constraint is violated. • If the number of the items returned is less than maxItems, the consumer can assume there are no more entries to retrieve at this time but there will still be a nextChangeToken returned to resume getting changes at a later time. • If changeToken is not provided, then this is assumed to be a service call to retrieve the first maxItems items. • The order in which the CmisChangedObjectType instances appear in the output set are in 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.

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.

Other details of the REST binding are TBD

WEB SERVICES BINDING

Same as abstract capability