Version 0.1

Content Management Interoperability Services

Unified Search Proposal

Versions

Version	Author	Date	Modifications
0.1	Gregory Melahn, IBM	02/09/2009	• N/A

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

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>
      <xs:simpleType name="enumCapabilityChanges">
            <xs:restriction base="xs:string">
                  <xs:enumeration value="none" />
                  <xs:enumeration value="updates" />
                  <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) Int maxItems: max # of items to be returned • (Optional) Boolean includeACL – includes ACL (provided ACL is part of CMIS) • (Optional) Boolean includeProperties – includes the properties for applicable objects
Outputs	• Result set containing the content created, updated or deleted since the last crawl • Output as a set of CmisChangedObjectType
Exceptions	• Common Exceptions
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 • For created, updated or security changed objects the access rights are included in the returned list if includeACL is true • For deleted objects, properties or access rights are not returned • The content-stream is not returned so a search crawler for example 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 • This service is exposed via capabilities but it is a core part of the spec • Filing changes MAY be considered as changes to the folders or documents as a repository option • The order of the result set is that the most recently updates appear first. **DISCUSSION TOPIC**: This is so that crawlers can more efficient;y find the newest changes without having to page through the entire collection. However (for REST) if next_rels were stable then instead the crawler could remember the next_rel it last saw and start from there. This would mean the service could instead answer the results as oldest first. This would (arguably) be more efficient for crawlers.

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 crawler to trim the results based on read access 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