CMIS Unified Search

Design Discussion - 17 February, 2009

Attendees:

Al Brown, IBM
Paul Goetz, SAP
Ethan Gur-esh, Microsoft
Laurent Hasson, IBM
Gregory Melahn, IBM
John Newton, Alfresco

Topics:

1. John asked whether one of the use cases we intend to support with this service would include a use case as he saw recently with a customer that has deployed TIBCO. They would want to trigger a workflow from a content change and so would be looking for more of a push than a pull model. It was agreed that at the face-to-face meeting, this was not one of the use cases we were explicitly targeting with this service. None the less, if we could support it with this service, that would still be useful. There were several issues identified with trying to support that use case with this service (a) performance of the system with potentially many push operations (b) the related fact that typical user activity that would cause content changes that were actually interesting to search engines (the targeted use case) would include bursts of changes to a single content item that could be skipped or consolidated in a periodic pull model. We decided that a service to push events really should be a separate service.
2. There was some discussion on how a search engine does an initial crawl of the repository. One approach would be call the service with no *changeToken* in which the repository could answer back all of the items in the repository. We decided that the way a search crawler would do an initial crawl would be to do a full query. A *changeToken* would then be required for incremental crawls from that point on. This token could either need to be available as a new service or as something that could be retrieved from the *getRepositoryInfo* service. We decided on the latter approach. So, in summary, the way a search crawler would behave to do a full crawl…
    a. call *getRepositoryInfo* and save changeToken from that service
    b. call the Query service with a queryString like "SELECT * FROM DOCUMENTS)
    c. call *getContentChanges* with the changeToken from Step (a).

3. Ethan pointed out there was an inconsistency in *enumCapabilityChanges* wrt the input parameters for *getContentChanges*
4. The meaning of maxItems was discussed again and it was agreed it would remain a repository choice on the default value.

5. We need to state how a repository is supposed to treat filing or unfiling operations on folder contents on whether they are answered as changes. The simplest approach is to say that a repository MAY treat a filing or unfiling action as a change to the folder or to the document or both, though those kinds of changes would typically not cause a search engine to want to re-index the folder or the document.
6. There was some discussion about only answering the properties of a document that have actually changed. For example, if only the property *customer_number* on a document changed then the entry would only have the document id and the customer_number in it, and not the rest of the properties. This was not considered practical in version one since not all repositories could efficiently answer that information based on the journal or query being used to implement the service
7. There was some discussion about building in additional filter criteria into the service and this was also considered not practical for version one.
8. Ethan pointed out that we need to state whether the object identified by the *changeToken* that is answered by the *getContentChanges* service is included in the set answered on the next *getContentChanges* service call. We decided that it should be (i.e. at time t0 with *changeToken* c0, the service answers *changeToken* c1 and set0 which includes x as the last item in the set, and at time t1 with *changeToken* c1, the service answers set1 which includes x as the first item in the set)
9. No one liked *nextChangeToken* so we will just use the name *changeToken* for input and output.
10. Ethan pointed out that using a *RuntimeException* to signal an expired change token is overloading that exception, and it would be better to use a new exception or the *ConstraintViolationException*. Greg did not want to define too many exceptions but the consensus was to define a new exception for this case.
11. There was discussion about whether we should not treat relationships and policies as changes. The consensus was that this should remain a repository choice but we should make this another repository capability answered by getRepositoryInfo.

Action Items:

1. Greg to update the document
2. Next meeting scheduled for week of 02/23/2009. Greg will send out suggestions for a specific date time (will try for 2/24).