DRS2 APIs Harvard Library Lab Project – 9/1/13 Progress Report

Spencer McEwen, Randy Stern, Chip Goines

This project has been planned as follows:

Phase	Planned Start date	Planned End date	Status
1. Planning and scoping	11/1/12	3/1/13	DONE
2. Technical design, including authentication/authorization for access to restricted DRS resources (BLOCKED fo	5/1/13	8/1/13	Progress made but not complete
3. Implementation –Implementation of search, metadata read, and content read APIs, including authentication /authorization for access to restricted DRS resources. Development work will utilize existing LTS development server. *NOTE: This estimate assumes Spencer does the API development work and we use the contractor to backfill on his DRS2 work.*	8/1/13	12/1/13	BLOCKED
4. Implementation of content delivery API that conforms to the International Image Interoperability Framework for the DRS Image Delivery Service.	6/1/13	9/1/13	In PROGRESS

PHASE 2- Due to limited availability of Spencer, some work has been done on Phase 2 in the past 3 months, but the majority of work on phase 3 will not be able to be completed before the ending of library lab funding in November. The Phase 2 status and work remaining is detailed below.

PHASE 4 - Significant progress was made on Phase 4. Chip Goines, LTS developer, has completed prototype implementation of a IIIF API for the existing DRS2 Image Delivery Service (IDS), based on the Luratech Image Content Server, and a proprietary API for delivering JPEG images from JPEG or JPEG2000 master images. He has been able to test with the Stanford University IIIF validation server, and the validation server reports that the code is now IIIF Level One compliant. His next step is to test whether the Level One compliance we have achieved will be adequate for integrating into the Multi-Image manuscript viewer being developed at Stanford as part of the Digital Medieval Manuscript Interoperability project.

Example URLS in the DRS QA environment (open only to Harvard) are:

http://idstest.lib.harvard.edu:9001/ids/iiif/760586/0,0,1200,1200/pct:50/full/native.jpg

http://idstest.lib.harvard.edu:9001/ids/iiif/760586/full/pct:50/full/native.jpg

http://idstest.lib.harvard.edu:9001/ids/iiif/760586/full/full/full/native.jpg

http://idstest.lib.harvard.edu:9001/ids/iiif/760586/full/pct:50/90/native.jpg

http://idstest.lib.harvard.edu:9001/ids/iiif/760586/0,0,500,500/pct:50/full/native.jpg

http://idstest.lib.harvard.edu:9001/ids/iiif/760586/0,0,500,500/full/full/native.jpg

http://idstest.lib.harvard.edu:9001/ids/iiif/760586/50,500,500,500/full/full/native.jpg

PHASE 2 – Details

A basic framework for a DRS2 API application now exists. Spencer's initial focus was on APIs for content owners. A prototype exists in which authentication and authorization work. If an owner of content wants to use the API they can talk to Spencer and he can create an API user and password for their application to use. They would only be able to access content within owner codes that they would normally be able to access in DRS2. Authorization is based on roles stored in our existing policy authentication database. From there they can use a basic set of object, file, and search API calls to retrieve and search metadata and download content files. XML or JSON is returned depending on the Accept HTTP header value. Using this API a content owner can get access to the full metadata and file contents of all their content in DRS2.

Spencer also started on a public version of the API that could be used to access public data in DRS2 (access flag = P) by non content owners. In the current prototype, object and file metadata can be retrieved without passing through the same authentication/authorization channels as content owners. Content access flags and other rights restrictions (embargo dates, secure storage, max image size, stream vs download) are also checked. While getting this working several questions were raised. More analysis is needed to determine which metadata fields should be exposed in the public API.

At this point, work had to end due to other pressing projects. Other things left to do include:
-implement rate limit throttling for authenticated and non authenticated requests -decide if current metadata format for authenticated requests is sufficient -implement addition API methods for object and file metadata access -implement addition API methods for public/non authenticated requests, implement a public search API.

Now that the basic framework of the application is established adding additional API methods should be relatively straightforward. With a few days work, the app could be made available for testing by external DRS2 content owners. The next immediate steps would be to finalize the authenticated API call and data response formats.