Integrating DataUp with DataVerse

Purpose:

The purpose of this document is to introduce at a high level some of the concepts leading to DataUp integration with DataVerse. In addition it can also be used as a pointer to some of the incomplete implementation for this integration that is being checked in, so that another developer can review the incomplete changes and continue on towards finishing the work.

Quick Introduction to DataVerse:

The test deployment of the DataVerse solution that we can use for development purposes is available here –

The first step is to navigate to the above link and create an account. Once you have an account, then you can log into your account and create a dataverse. A [Dataverse](http://thedata.org/book/features) is a container for research data studies, customized and managed by its owner. Inside your dataverse you can then create Studies which can essentially be understood as sub-containers within your container.

Talking in terms of DataUp, a Dataverse Repository that we create in DataUp will not actually be a new DataVerse, but it will be a Study (sub-container) created within the top-level container or DataVerse that we have manually created.

The following link has a quick reference to the DataVerse Api set -

<http://thedata.harvard.edu/guides/dataverse-api-main.html#data-deposit-api>

Implementation details of DataVerse Integration Code (Incomplete Code):

* Adding a BaseRepository for DataVerse:

To accomplish this we have to add a base repository row for DataVerse in the BaseRepository table in the db.

* DataVerse configuration:

To communicate with the DataVerse server we need some key configuration information that essentially includes (but is not limited to) ServerName, DataVerse Root, UserName, Password.

To store this configuration info the Web.config for WebAPI gets a new ConfigurationSection called DataVerse. This section can be extended to get more config values in the future.

* DataVerse Repository Adapter:

The interface IRepositoryAdapter is extended to create a new Repository Adapter for dataverse. This adapter implements (pseudo implementation available at the moment) methods for creating a new Study and uploading files to a Study. We ideally would also like to have methods for checking for the presence of a file in a repository and downloading a file from a repository, however the DataVerse API set doesn’t seem to support these APIs. We’ll have to check with the DataVerse dev team to get clarifications about these missing apis.

The other challenge with the implementation of this adapter is to figure out the C# binding for calling SWORD based DataVerse APIs. The API documentation lists the CURL commands for interacting with the Server however using Curl commands from the DataUp code is a very poor implementation choice and that approach should only be taken as the last resort. I tried to HttpWebRequest class in the .Net Framework but didn’t get success with that.

P.S.: I checked with Philip Durbin in the DataVerse dev team about the availability of the c# binding for SWORD and he didn’t seem to know the answer. He suggested that we develop our own C# binding, but that is a sizable project by itself. This in my mind is the biggest blocker in terms of wrapping up this project.

* Creating a Study in DataVerse:

As explained earlier a Study in a DataVerse is a sub-container that we create below the root container. The files can only be uploaded to the Studies we create below the root.

When Creating a Study we have to author an xml called atom-entry-study.xml that contains metadata elements for the Study and pass it to the SWORD call to create the Study. A Study that we create is private by default. A Study becomes public when a file is uploaded to the Study and subsequently published.

* Uploading a File to a DataVerse Study:

A Study in DataVerse is a versioned entity (like source control). A file upload operation on dataverse has to be followed with a Publish operation so the file becomes public. To package the metadata for the file along with the data file being uploaded we leverage the zipped file technique in place for Merritt.

* DataVerse File Service:

The class DataVerseFileService extends FileServiceProvider and encapsulates the functionality of the packaging a file and calling the right adapter for post a file to DataVerse.