Spartan Archives Documentation and Procedures

Documentation

GeneralQuery (Researcher Interface)
Fedora Data Streams and Researchers
Fedora Objects and Data Streams
How Fedora Objects are used in the Spartan Archive
Java Programs used in the Spartan Archive
PDFPuller Documentation
SAControl Documentation
Spartan Archive Ingest Process
Spartan Archives Data Sources

Procedures

Adding a new Office or Collection
Changing Ingest Preferences
Dealing with Potential Problems
Ingesting a Records Collection
Manually Updating Data in Fedora
Updating Spartan Archives Parameters
Updating Spartan Archives Programs
Updating the Data Dictionary for a Collection

Special

Possible Enhancements to the Spartan Archive
Architectural Diagrams
**Documentation**

This section contains the descriptions of the programs and processes that comprise the Spartan Archive system. This includes the ingest process and the researcher interface and the administrative interface.

The researcher interface is covered in 2 sections: GeneralQuery and How Fedora Objects are used in the Spartan Archive. The first of these sections covers the Java/Javascript/JSP programs that provide access to the database search processes. The second covers the Fedora/xslt processes that display the contents of the archive and the PDF files.

Further information on how Fedora is used and what resources are provided are listed in Fedora Data Streams and Researchers, and Fedora Objects and Data Streams.

PDFPuller describes the programs that support the ingest of PDF files to the archive.

The Spartan Archive Ingest Process describes the ingest process and the Java programs involved in the process.

SAControl describes how the Administrative Interface is built and the programs that support it.

Spartan Archive Data Sources is a tool for front end developers telling them what data is available to them in creating web pages using JSP or AJAX technology.

Finally, Java Programs used in the Spartan Archives is an inventory of all of the Java programs involved. It contains documentation for the various utility and library functions used in the system.

**Procedures**

This section covers how to perform various tasks in the archive.

**Special**

This contains a list of proposed enhancements and diagrams of the server configuration and the overall architecture.
Presenting the Database Objects to Researchers

Overview

At the top levels all data presented to researcher comes directly from Fedora as described in other documents. However at the lowest level all record data in the Spartan Archive is stored in a database and presented via Java based servlets and Java Server Pages (JSP). This is illustrated below:
As shown above Academic Programs are presented in a different manner than other programs due to the nature of the information in the collection. This will be discussed further below.

This document will explain the design criteria used and the mechanisms employed to provide the data. Companion documents will describe the properties available to JSP web pages and provide guidance for adding additional record based collections.

Overview

The core of the applications are the two Java servlets: APData.java and GeneralQuery.java. Both of these servlets are essentially switches. After setting up the context of the request, they determine what is being asked for and delegate the fulfillment of the request to ordinary Java objects.

Database access utilizes the Tomcat Java application server’s database connection pooling facilities. These are defined in the context.xml file located in the META-INF directory of the application (See Listing 1 in the appendix).

Academic Programs

Academic Programs is a different sort of collection than the others. While all of the collections were originally presented in the form of paper publications, the others were basically lists of things or records. A student directory is a list of information about students, a course schedule is a list of information about courses that are offered in a semester. In contrast the Academic Programs document is a true document. While it is split into sections the only relevant fields in each section are the section titles and the section contents. In addition the document is organized as a hierarchy rather than a list. To deal with this structure the presentation as shown below:
On the left there is a hierarchical display of all of the titles in the document. When a researcher clicks on any of these titles the text associated with the title appears in the box labeled contents.

In addition there is a search field on the top that allows a researcher to find the text of any section by entering any fragment (word, part of word or phrase) found in the key words provided by the Office of the Registrar. A list of titles that match show up in the box to the right and a user can click on any of them to show the text. When this happens the titles on the left are scrolled to show context of the element found.

The content may contain links to websites outside of the archive. At this point all of those links point to their current versions. As a result the information found via those links may no longer be relevant, or the pages linked to may be dead. A future enhancement would be to convert the links to point to the Archive-it versions that were active at the time of publication.

Building the Page

This page is called from the Academic Programs collection page that was generated by Fedora. The URL is http://spartanarchive.msu.edu:8080/GeneralQuery/APEntry.html?name=Academic Programs&key=2011-2012

When the page is first entered the onload function (in populateTitles.js) is called which in turn does an asynchronous call to a Java servlet (APData.java).
That servlet uses the information passed to it to call functions in APReporting.java to build XML responses to return to the web page. The functions in APReporting in turn call APDataSource.java to extract the requested data.

Once the data is returned to the web page, the javascript program populates the titles box.

**Page Functions**

In order to make this useful and responsive, this portion of the application does not use JSP techniques. Rather, it depends on javascript and AJAX (Asynchronous Javascript And XML).

When a user clicks on a title, a javascript function does another asynchronous call to APData.java to return the requested content. On return the javascript program populates the content box and adjusts the title scroll position as needed.

When a user enters a value in the search box and clicks the search button a second javascript program (keywordSearch.js) does an asynchronous call to APData.java to find all of the titles that correspond to the keywords found. Clicking on one of the titles returned kicks off the same process as in the paragraph above.

APData.java responds to 3 different requests:

- **titles**: Returns an xml object with all titles for the year’s Academic Programs
- **content**: Returns an html object with the content associated with the requested title
- **keywordSearch**: Returns an xml object containing a list of all matching titles

In all cases APData calls the appropriate function in APReporting.java. These in turn call a function in APDataSource which extracts the data from the database and returns it as a list of Instance class objects. These are translated into an xml stream in APReporting.

**Classes Used**

<table>
<thead>
<tr>
<th>Class</th>
<th>Source File</th>
</tr>
</thead>
<tbody>
<tr>
<td>APData</td>
<td>APData.java</td>
</tr>
<tr>
<td>APReporting</td>
<td>APReporting.java</td>
</tr>
<tr>
<td>APDataSource</td>
<td>APDataSource.java</td>
</tr>
<tr>
<td>Instance</td>
<td></td>
</tr>
</tbody>
</table>

**JavaScript used:**

<table>
<thead>
<tr>
<th>JavaScript</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jquery.js</td>
<td>Used to support AJAX processing and tag manipulation</td>
</tr>
<tr>
<td>jxs.js</td>
<td>Used to support AJAX processing</td>
</tr>
</tbody>
</table>
Other Collections

All other collections are served by just two web pages: buildQuery.jsp and showSearchResults.jsp.

The Query Page

buildQuery.jsp is shown below. The first two columns consist of drop boxes that contain all of the data fields in the collection and all relevant search operations on those fields respectively. In the case of the potential search fields a definition pops up when a user holds the mouse over the name. The box on the right appears when the field has a limited set of non-obvious values. While choosing a value does not automatically populate the field, a user can cut and paste values from the list to the search field.

Building the page

The URL to load this page is similar to http://spartanarchive.msu.edu:8080/GeneralQuery/GeneralQuery?type=queryBuilder&name=CourseDescriptions&key=2011

In this URL name is always the short name of the collection (formed by removing the space so Course Descriptions becomes CourseDescriptions, Course Schedules
becomes CourseSchedules, etc.), key is the period in question. For semester based collections it is in the form semester id, last two digits of year. Fall Semester is FS, Spring SS and Summer US. So fall 2011 is FS11. If the user chose to search all instances of the collection the key is “all.”

Unlike Academic Programs the web page is not immediately loaded. Rather GeneralQuery.java gets control. Like APData, GeneralQuery is a Servlet that provides minimal process, routing most functionality to normal java classes (POJOs).

The type parameter of the URL tells GeneralQuery how to proceed. When it equals queryBuilder it:

- Calls FedoraAccess.java to return the list of field values and definitions. The values are used to population the field choices in the drop down box; the definitions support the popups related to the field names.
- Removes the Semester or Year field if only a single instance is requested
- Calls LookupLoader.java to load all relevant look up lists for the collection. LookupLoader sets request attributes that are used by the web page to populate the tables.
- Sets any remaining attributes and
- Loads the queryBuilder.jsp web page.

The page itself uses standard JSP tags to load the values into the fields select tag as options (the definitions are loaded as the option’s titles). The lookups are stored in invisible tables that are displayed when a user selects one of the associated fields. In addition a javascript program initializes the selections, the page name and sets up a series of options based on the type of the collection.

Page Functions

Users select fields and functions and provide the values for the fields desired. If a multipart search is required clicking on the And/Or buttons enables those features. Once the user is done and clicks on the Select button, the javascript program kicks in.

The javascript program uses the user’s selections to build a database query. (One caveat here: because HTTP uses the ‘%’ for different purposes than SQL the query is not entirely valid if the user chose the ‘contains’ option. This will be corrected on the back end.) In doing so it translates the English language field names to the database field names based on a set of properties in the collections options. Once that is done it transfers control back to the GeneralQuery Servlet.

In order to make this as generic as possible, the application uses a single “data source” class called Collection.java to execute the constructed query. It returns all rows found as arrays of String, where each entry in the array is a database column. All columns with the exception of the generated id are returned. The column titles are extracted from the collections data dictionary. (Asa result of this it is critical that the data dictionary
fields and the columns in the database are in the same order. Otherwise the displayed results make no sense.)

When row arrays are returned GeneralQuery sets several other attributes and calls the showSearchResults.jsp page.

The downloadable file is stored in the downloads folder within the GeneralQuery directory in the webapps directory of the fedora.ats.msu.edu server. Each file is named with a truncated version of the time it is created (in seconds). As each file is created any files more than a day old are deleted.

The Search Results Page

Once the data is extracted the showSearchResults.jsp builds the page using JSP tags. Two of the request attributes that are key to the generic nature of the page are ‘titles’ and ‘queryOutput.’ Titles is an array consisting of all of the field names belonging to the collection in the database. QueryOutput is an array contains each row that matched the user’s search criteria. (It is rather critical that these stay in sync.)

The resulting page will look like:

![Spartan Archive
An Electronic Records Archive at Michigan State University](image)

354 records returned
Download results

<table>
<thead>
<tr>
<th>Year</th>
<th>Code</th>
<th>Status</th>
<th>Start</th>
<th>End</th>
<th>Semesters</th>
<th>Course Title</th>
<th>Credits</th>
<th>Lectures</th>
<th>Lab</th>
<th>Lecture Credits</th>
<th>Lab Credits</th>
<th>Min Credits</th>
<th>Max Credits</th>
<th>Increment</th>
<th>Credit Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S593</td>
<td>0</td>
<td>Intermediate Algebra</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Survey of Calculus</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Survey of Calculus with Applications I</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Survey of Calculus with Applications II</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Calculus I</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Calculus II</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Calculus III</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Mathematical Investigation I</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Mathematical Investigation II</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Mathematical Investigation III</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Mathematical Investigation IV</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Mathematical Investigation V</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Mathematical Investigation VI</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Mathematical Investigation VII</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Mathematical Investigation VIII</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Mathematical Investigation IX</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Mathematical Investigation X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Mathematical Investigation XI</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Mathematical Investigation XII</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>MTH</td>
<td>Archived</td>
<td>F592</td>
<td>S598</td>
<td>0</td>
<td>Faculty directed study</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Classes Used:

<table>
<thead>
<tr>
<th>Class</th>
<th>Source File</th>
</tr>
</thead>
<tbody>
<tr>
<td>CourseDescription</td>
<td>CourseDescription.java</td>
</tr>
</tbody>
</table>
### Sub-Processes

#### Data Dictionary Process

There are 3 classes here: FedoraAccess, DataDictionaryProcessor and DataDictionary. Data dictionary data is stored in Fedora in the msu-uahc:DDCollectionName objects. (Current examples of CollectionName include CourseDescriptions, CourseSchedules, and StudentDirectory.) This data is stored in xml form in a data stream called 'source.'

FedoraAccess extracts that data stream from Fedora and sends it to DataDictionaryProcessor for conversion into a DataDictionary class. DataDictionaryProcessor does this using the jdom xml processing library.

The DataDictionary class is returned to buildQuery.jsp as a request attribute.

#### Classes Used:

<table>
<thead>
<tr>
<th>Class</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataDictionary</td>
<td>DataDictionary.java</td>
</tr>
<tr>
<td>DataDictionaryProcessor</td>
<td>DataDictionaryProcessor.java</td>
</tr>
<tr>
<td>DataSource</td>
<td>DataSource.java</td>
</tr>
</tbody>
</table>
Lookup Table Process

The Lookup Table process consists of two classes: LookupLoader and Lookup. LookupLoader takes the collection name as a parameter and sets request attributes for each table relevant to the collection.

The specific attributes are all lists of the lookup values stored as Lookup objects. These may consist of a single value or two values. In the latter case the first value is the one that goes into the search field while the second is a description of that value.

Classes Used:

<table>
<thead>
<tr>
<th>Class</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lookup</td>
<td>Lookup.java</td>
</tr>
<tr>
<td>LookupLoader</td>
<td>LookupLoader.java</td>
</tr>
</tbody>
</table>
Appendix

Listing 1:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Context antiJARLocking="true" path="/GeneralQuery">
  <Resource auth="Container" driverClassName="com.mysql.jdbc.Driver" maxActive="100"
           maxIdle="10" maxWait="10000" name="jdbc/CourseSchedules"
           password="xxxxxxxxxxxx" type="javax.sql.DataSource"
           url="jdbc:mysql://fedora-db.ats.msu.edu/courseschedules"
           username="spartanArchives" removeAbandoned="true" logAbandoned="true" autoReconnect="true"/>
  <Resource auth="Container" driverClassName="com.mysql.jdbc.Driver" maxActive="100"
           maxIdle="10" maxWait="10000" name="jdbc/Lookup"
           password="xxxxxxxxxxxxx" type="javax.sql.DataSource"
           url="jdbc:mysql://fedora-db.ats.msu.edu/lookup"
           username="spartanArchives" removeAbandoned="true" logAbandoned="true" autoReconnect="true"/>
  <Resource auth="Container" driverClassName="com.mysql.jdbc.Driver" maxActive="100"
           maxIdle="10" maxWait="10000" name="jdbc/AcadProg"
           password="xxxxxxxxxxxxx" type="javax.sql.DataSource"
           url="jdbc:mysql://fedora-db.ats.msu.edu/acadPrograms" username="spartanArchives"
           removeAbandoned="true" logAbandoned="true" autoReconnect="true"/>
  <Resource auth="Container" driverClassName="com.mysql.jdbc.Driver" maxActive="100"
           maxIdle="10" maxWait="10000" name="jdbc/StudentDirectory"
           password="xxxxxxxxxxxx" type="javax.sql.DataSource"
           url="jdbc:mysql://fedora-db.ats.msu.edu/studentdirectory"
           username="spartanArchives" removeAbandoned="true" logAbandoned="true" autoReconnect="true"/>
  <Resource auth="Container" driverClassName="com.mysql.jdbc.Driver" maxActive="100"
           maxIdle="10" maxWait="10000" name="jdbc/CourseDescriptions"
           password="xxxxxxxxxx" type="javax.sql.DataSource"
           url="jdbc:mysql://fedora-db.ats.msu.edu/coursedescriptions"
           username="spartanArchives" removeAbandoned="true" logAbandoned="true" autoReconnect="true"/>
</Context>

See Fedora Objects for sample data dictionary listings.
```
Javascript Programs

Name: **cdInfo.js**  
Used by: queryBuilder.jsp  
Purpose: Provides configuration information for building searches on Course Descriptions  
Depends on: None

Name: **jquery.js**  
Used by: APEntry.html  
queryBuilder.jsp  
Purpose: Provides AJAX functionality and html tag manipulation.  
Depends on:

Name: **jxs.js**  
Used by: APEntry.html  
Purpose: Provides AJAX functionality  
Depends on:

Name: **keywordSearch.js**  
Used by: APEntry.html  
Purpose: Populates title results area based on the search term provided by the researcher.  
Depends on: jxs.js

Name: **populateTitles.js**  
Used by: APEntry.html  
Purpose: Populates Academic Program data in the page and responds to user title requests.  
Depends on: jquery.js

Name: **queryBuilder.js**  
Used by: queryBuilder.jsp  
Purpose: Initializes search fields, displays lookup lists and builds the database query  
Depends on: cdInfo  
  sdlInfo  
  socInfo

Name: **sdInfo.js**  
Used by: queryBuilder.jsp  
Purpose: Provides configuration information for building searches on Student Directory  
Depends on: None

Name: **showSearch.js**  
Used by: showSearchResults.jsp  
Purpose: Sets window size
Name: socInfo.js
Used by: queryBuilder.jsp
Purpose: Provides configuration information for building searches on Course Schedules
Depends on: None

Libraries Used

<table>
<thead>
<tr>
<th>Library</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>fedora-client-0.1.6-with-dependencies-bin.jar</td>
<td>Provides programmatic access to Fedora</td>
</tr>
<tr>
<td>jdom.jar</td>
<td>Provides facilities to read and write xml data streams</td>
</tr>
<tr>
<td>jstl.jar</td>
<td>Provides JSP functionality</td>
</tr>
<tr>
<td>log4j-1.2.16.jar</td>
<td>Provides logging functionality</td>
</tr>
<tr>
<td>mysql-connector-java-5.1.16-bin.jar</td>
<td>Provides access to the mysql database</td>
</tr>
<tr>
<td>standard.jar</td>
<td>Provides more JSP functionality</td>
</tr>
</tbody>
</table>
Java Programs

All Java classes are located in the datasource package.

Servlets:

Name: **APData.java**
Purpose: Populates title results area based on the search term provided by the researcher.
Uses:
- APReporting.java
- APDataSource.java

Attributes exported: None

Name: **GeneralQuery.java**
Purpose: Provides data (in the form of request attributes) for the search (query builder page) and the search results page.
Uses:
- CollectionSource.java
- DataDictionary.java
- DataDictionaryProcessor.java
- FedoraAccess.java
- Lookup.java
- LookupLoader.java

Classes

Name: **APReporting**
Purpose: Formats data extracted via APDataSource into xml format and writes to the Servlet’s output. The xml is used by javascript functions in APEntry.html to update the page.

Name: **APDataSource**
Purpose: Does SQL queries to supply Academic Program information to the system.

Name: **APDataSource.Instance**
Purpose: Internal class used to transport data from APDataSource to APReporting

Name: **CollectionSource**
Purpose: Massages any “like” queries, executes them and returns the results as a list of Arrays of Strings.

Name: **DataDictionary**
Purpose: List of Entry objects.
Name: **DataDictionary.Entry**  
Purpose: Internal class that organizes fields and definitions.

Name: **DataDictionaryProcessor**  
Purpose: Does jdom processing that converts the xml ‘source’ data stream from Fedora into a DataDictionary.

Name: **FedoraAccess**  
Purpose: Connects to Fedora and pulls a collections data dictionary in xml form from the appropriate ‘source’ data stream.

Name: **Lookup**  
Purpose: Stores a lookup table name (for example Subject Code) and a list of all of its possible values. In some cases the value and its meaning are concatenated together with a “:” separating them.

Name: **LookupLoader**  
Purpose: Based on the collection being searched, performs a series of SQL queries from dedicate lookup tables, storing each set of information in a Lookup Object. A List of LookupObjects is returned to the transaction as a request parameter.
How Fedora Data Streams are Presented to the Researcher

Researchers interact with Fedora in several ways:

- The main Spartan Archive page is generated from a Fedora data stream within the msu-uahc:SpartanArchive object via an XSL transformation.
- Each primary collection page is generated from the Fedora object corresponding to that collection.
- While the detail pages (for the data based collections) are generated by Java from a database, the data definition popups are generated from data streams associated with the appropriate instance of the collection.

As described in *Spartan Archive Fedora Objects and Data Streams: Summary and Examples* the msu-uahc:SpartanArchives object contains a data stream called ‘source.’ That data stream contains an xml representation of each office and the collections that belong to that office. This include the name of the collection, limited metadata and the expected schedule of deliveries. This is rendered via an xsl transform with a template located in the xsl data stream.

The main page contains links to each of collection’s main page. These pages are again generated through an xsl transform using the collections DublinCore data stream and the associated dcXSL data stream. There is only one copy of the dcXSL data stream for the data based collections and that is stored in msu-uahc:CDTransform. It is referenced by each collection object.

Each collection page contains links to instances of that collection. In the case of the data based collections these are generated via the Java servlet based backend and a set of jsp pages. The data definition popups are read from Fedora (by the servlet) and provided to the query page. The pages that these links address are all generated from Fedora data streams.

The data definition popups are obtained from the source data streams within each instance object. However there is only one copy of each. (The theory being that the data definitions which comprise the user guide are valid across all instances.) They are located in objects named msu-uahc:RG-CollectionName (where RG is the report group and CollectionName is the name of the collection. E.g. msu-uahc:UA.6.7-StudentDirectory), and msu-uahc:DDTransform. Each instance object references the data streams from these standard objects.
Spartan Archive Fedora Objects and Data Streams: Summary and Examples

This document contains a list of all objects and data stream types found in the Spartan Archive Fedora system. They are more completely explained in Fedora Objects and Data Streams.

**Spartan Archive Fedora Objects**

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
<th>Data Streams</th>
</tr>
</thead>
<tbody>
<tr>
<td>msu-uahc: SpartanArchive</td>
<td>Entry to the Spartan Archive as a whole</td>
<td>RELS-EXT source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>xsl defaultTypes</td>
</tr>
<tr>
<td>msu-uahc: RG-CollName</td>
<td>Collection object</td>
<td>RELS-EXT source</td>
</tr>
<tr>
<td></td>
<td>This is associated with its instances via the DublinCore data stream</td>
<td>DublinCore</td>
</tr>
<tr>
<td></td>
<td>relation element</td>
<td>dcXSL dataDictionary</td>
</tr>
<tr>
<td></td>
<td>Current collections are:</td>
<td>xsl defaults source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>source</td>
</tr>
<tr>
<td>msu-uahc: UA.6.7-AcadProg</td>
<td>Collection object for the Academic Programs collection</td>
<td>RELS-EXT source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DublinCore</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dcXSL dataDictionary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>xsl defaults source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>source</td>
</tr>
<tr>
<td>msu-uahc: UA.6.7-Courses</td>
<td>Collection object for the Schedule of Courses collection</td>
<td>RELS-EXT source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DublinCore</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dcXSL dataDictionary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>xsl defaults source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>source</td>
</tr>
<tr>
<td>msu-uahc: UA.6.7-CourseDs</td>
<td>Collection object for the Description of Courses collection</td>
<td>RELS-EXT source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DublinCore</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dcXSL dataDictionary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>xsl defaults source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>source</td>
</tr>
<tr>
<td>Name</td>
<td>Purpose</td>
<td>Data Streams</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>msu-uahc: UA.6.7-StudentDirectory</td>
<td>Collection object for the Student Directory collection</td>
<td>RELS-EXT DublinCore dcXSL dataDictionary xsl defaults source</td>
</tr>
<tr>
<td>msu-uahc: RG-ATAccessionNumber-XXX</td>
<td>Instance object for an instance of one of the above collections</td>
<td>RELS-EXT DublinCore dataDictionary source PDFFileN (1-#of PDFs) xsl premis</td>
</tr>
<tr>
<td></td>
<td>This is associated with its collection via the DublinCore relation element</td>
<td>(see note below)</td>
</tr>
<tr>
<td>msu-uahc: ex3CModel</td>
<td>The content model for the Registrars Office files used to support the display of the main page and the pdf pages</td>
<td>RELS-EXT</td>
</tr>
<tr>
<td>msu-uahc: ex3CSDef</td>
<td>Service definition object for displaying the main page and the pdf pages</td>
<td>RELS-EXT METHODMAP</td>
</tr>
<tr>
<td>msu-uahc: ex3CSDep</td>
<td>Service deployment object for displaying the main page and the pdf pages</td>
<td>RELS-EXT METHODMAP DSINPUTSPEC WSDL</td>
</tr>
<tr>
<td>demo: ex3CModel</td>
<td>The content model for the Registrars Office files used to support the display of the collection database search pages</td>
<td>RELS-EXT</td>
</tr>
<tr>
<td>demo: ex3CSDef</td>
<td>Service definition object for displaying the collection database search pages</td>
<td>RELS-EXT METHODMAP</td>
</tr>
<tr>
<td>demo: ex3CSDep</td>
<td>displaying the collection database search pages</td>
<td>RELS-EXT METHODMAP DSINPUTSPEC WSDL</td>
</tr>
<tr>
<td>Name</td>
<td>Purpose</td>
<td>Data Streams</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>msu-uahc: DDCollectionName</td>
<td>Objects that contain the data definitions for the collection</td>
<td>dataDictionary defaults</td>
</tr>
<tr>
<td>msu-uahc: DDAcademicPrograms</td>
<td>Object that contains the data definition for the Academic Programs collection</td>
<td>dataDictionary defaults</td>
</tr>
<tr>
<td>msu-uahc: DDCourseSchedules</td>
<td>Object that contains the data definition for the Schedule of Courses collection</td>
<td>dataDictionary defaults</td>
</tr>
<tr>
<td>msu-uahc: DDCourseDescription</td>
<td>Object that contains the data definition for the Description of Courses collection</td>
<td>dataDictionary defaults</td>
</tr>
<tr>
<td>msu-uahc: DDSStudentDirectory</td>
<td>Object that contains the data definition for the Student Directory collection</td>
<td>dataDictionary defaults</td>
</tr>
<tr>
<td>msu-uahc: CDTransform</td>
<td>Object that contains the stylesheet for transforming collection metadata</td>
<td>dcXSL</td>
</tr>
<tr>
<td>msu-uahc: DDTransform</td>
<td>Object that contains the stylesheet for transforming collection pdf lists</td>
<td>xsl</td>
</tr>
<tr>
<td>msu-uahc: LLTransform</td>
<td>Object that contains the stylesheet for transforming the PDF List (source) to a web page</td>
<td>xsl</td>
</tr>
</tbody>
</table>

Note: A collection instance can contain either references to database objects or PDFs. In the first case there will be no source data stream or PDFFileN data streams. Those are only relevant to PDF instances.
Spartan Archive Data Steams

This is a list of all of the fedora data streams used in the Spartan Archive system. Along with listing the location, name and purpose an example of each is included.

Object: msu-uahc:SpartanArchive:
Name: source
Purpose: Provides metadata on all collections within the archive
Example:
<collections>
  <office name="Office of the Registrar" shortName="RegistrarsOffice">
    <collection>
      <title>Academic Programs</title>
      <pid>msu-uahc:UA.6.7-AcademicPrograms</pid>
      <creator>MSU Office of the Registrar</creator>
      <lastUpdate>July 16, 2011</lastUpdate>
      <periodCovered>
        <start>2011-2012</start>
        <end/>
      </periodCovered>
      <description>Annual directory of descriptions of academic programs</description>
      <key>ap</key>
      <submissionSchedule>
        <submission period="year">
          <dueDate>7/15</dueDate>
          <alarmDate>8/15</alarmDate>
        </submission>
      </submissionSchedule>
    </collection>
    <collection>
      <title>Course Descriptions</title>
      <pid>msu-uahc:UA.6.7-CourseDescriptions</pid>
      <creator>MSU Office of the Registrar</creator>
      <lastUpdate>August 26, 2011</lastUpdate>
      <periodCovered>
        <start>1992</start>
        <end>2012</end>
      </periodCovered>
      <description>Annual listing of course descriptions</description>
      <key>cd</key>
      <submissionSchedule>
        <submission period="year">
          <dueDate>7/15</dueDate>
          <alarmDate>8/15</alarmDate>
        </submission>
      </submissionSchedule>
    </collection>
    <collection>
      <title>Course Schedules</title>
      <pid>msu-uahc:UA.6.7-CourseSchedules</pid>
      <creator>MSU Office of the Registrar</creator>
      <lastUpdate>August 26, 2011</lastUpdate>
      <periodCovered>
        <start>Fall 2003</start>
        <end>Fall 2011</end>
      </periodCovered>
      <description>Course information for each semester, including locations, instructors, credit hours, and special instructions</description>
      <key>soc</key>
  </collection>
</collections>
<submissionSchedule>
  <submission period="ss">
    <dueDate>1/15</dueDate>
    <alarmDate>2/15</alarmDate>
  </submission>
  <submission period="us">
    <dueDate>6/15</dueDate>
    <alarmDate>7/15</alarmDate>
  </submission>
  <submission period="fs">
    <dueDate>9/15</dueDate>
    <alarmDate>10/15</alarmDate>
  </submission>
</submissionSchedule>

<submissionSchedule>
  <submission period="ss">
    <dueDate>1/15</dueDate>
    <alarmDate>2/15</alarmDate>
  </submission>
  <submission period="us">
    <dueDate>6/15</dueDate>
    <alarmDate>7/15</alarmDate>
  </submission>
  <submission period="fs">
    <dueDate>9/15</dueDate>
    <alarmDate>10/15</alarmDate>
  </submission>
</submissionSchedule>

<collection>
  <title>Student Directory</title>
  <pid>msu-uahc:UA.6.7-StudentDirectory</pid>
  <creator>MSU Office of the Registrar</creator>
  <lastUpdate>August 26, 2011</lastUpdate>
  <periodCovered>
    <start>Spring 1994</start>
    <end>Fall 2011</end>
  </periodCovered>
  <description>Directory of local and permanent contact information for MSU students for each semester</description>
  <key>sd</key>
  <submissionSchedule>
    <submission period="ss">
      <dueDate>1/15</dueDate>
      <alarmDate>2/15</alarmDate>
    </submission>
    <submission period="us">
      <dueDate>6/15</dueDate>
      <alarmDate>7/15</alarmDate>
    </submission>
    <submission period="fs">
      <dueDate>9/15</dueDate>
      <alarmDate>10/15</alarmDate>
    </submission>
  </submissionSchedule>
</collection>
</collections>
Object: msu-uahc:SpartanArchive:
Name: xsl
Purpose: This is the stylesheet that transforms the source data stream into the main Spartan Archive page
Example:
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <html>
      <head>
        <title>Spartan Archives</title>
        <style type="text/css">
          body{
            font-family:Helvetica, Arial, sans-serif;
            font-size: 12px;
          }
          h3{
            font-size: 18px;
            font-weight: bold;
          }
          .title{
            text-align: center;
            padding-bottom:5px;
            font-size: 24px;
          }
          #content{
            margin-left:80px;
            padding-bottom: 10px;
          }
          #content h4{
            font-size:90%
          }
          .item{
            font-size: 120%;
            font-weight: bold;
            text-decoration: underline;
            padding-bottom: 5px;
          }
          .collection{
            padding-bottom:15px;
          }
          .key{
            font-weight: bold;
          }
          .value{
            font-weight: lighter;
          }
        </style>
      </head>
      <body>
        <div class="title">
          <div class="bigger">The Spartan Archives</div>
          <div>Michigan State University's Home for Digital Preservation</div>
        </div>
        <div id="content">
          <h3>Collections</h3>
          <xsl:apply-templates select="/collections/collection"></xsl:apply-templates>
        </div>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
**Object:** msu-uahc:SpartanArchive:

**Name:** defaultTypes

**Purpose:** This defines the configurable items available in the Spartan Archive System.

**Example:**

```xml
<metadataDefaultTypes>
  <type position="1">
    <name>Submission Name</name>
    <description>Name of the file that is submitted to the Archives</description>
  </type>
  <type position="2">
    <name>Collection Name</name>
    <description>The name of the collection</description>
  </type>
  <type position="3">
    <name>Collection Short Name</name>
    <description>This is the name that Fedora uses in its identifiers. (e.g., CourseSchedules rather than Schedule of Courses)</description>
  </type>
  <type position="4">
    <name>Record Group</name>
    <description>All identifiers in fedora will be of the type msu-uahc:Record Group-AT accession number. This defines the record group to be used. For example UA.6.7</description>
  </type>
  <type position="5">
    <name>Record Group Description</name>
    <description>The name associated with the record group, for example for UA.6.7 the description would be Office of the Registrar</description>
  </type>
  <type position="6">
    <name>Record Group Short Name</name>
    <description>The short name to be used in creating a storage space. For example the short name for UA.6.7 is RegistrarsOffice so records belonging to it would be stored in the directory UA.6.7_RegistrarsOffice</description>
  </type>
  <type position="7">
    <name>Publisher</name>
    <description>The name of the publisher, and the scheme from which it derives, associated with the collection. For example: Value is Michigan State University, Scheme is NACO</description>
  </type>
  <type position="8">
    <name>Language</name>
    <description>The language code to be used and the scheme from which it is derived. For example: Value is eng, Scheme is iso63903</description>
  </type>
  <type position="9">
    <name>Type</name>
    <description>The type of collection and the scheme from which it is derived. For example: Value is dataset, Scheme is DCMITYPE</description>
  </type>
  <type position="10">
    <name>Coverage</name>
    <description>The spatial or temporal area that the collection covers, its scheme and its qualifier. For example, Value is East Lansing (Mich.), Scheme is LCNAF, qualifier is spatial</description>
  </type>
</metadataDefaultTypes>
```
<type position="12">
  <name>Creator</name>
  <description>The fully qualified name of the organization that created the submission and the scheme from which it is derived. For example: Value is Michigan State University.</description>
</type>

<type position="13">
  <name>Subject</name>
  <description>The subject of the submission and the scheme from which it derived. There may be multiple subjects for a submission. For example: Value is Academic Programs.</description>
</type>

<type position="14">
  <name>Contributor</name>
  <description>The name of the person who actually contributed the submission and the scheme from which it derives. For example: Value is Doe, John. Scheme is NACO.</description>
</type>

<type position="15">
  <name>Format</name>
  <description>The mime type of the submission and the scheme from which it derives. For example: Value is text/xml, Scheme is IMT.</description>
</type>

<type position="16">
  <name>Relation</name>
  <description>The relationship between this item and another in the archive. The value is the object that this item is related to, the qualifier is the type of relationship. For example: Value is msu-uahc:AcademicPrograms, Qualifier is is part of.</description>
</type>

<type position="17">
  <name>Rights</name>
  <description>How this submission may be used. For example: Value is Use of this public domain resource is unrestricted.</description>
</type>

<type position="18">
  <name>Significant Properties</name>
  <description>These are values in the Premis metadata for information that the organization thinks is important, but is not otherwise included. Use Qualifier field for its type, and the value field for its value. For Example: Value is xml, Qualifier is structure.</description>
</type>
**Object:** msu-uahc:RG-Collection

**Name:** DublinCore

**Purpose:** This is the Dublin Core metadata data stream for the collection

**Example:**

```xml
<dublinCore>
  <identifier>msu-uahc:UA.6.7-StudentDirectory</identifier>
  <title>Student Directory</title>
  <creator scheme="NACO">Michigan State University. Office of the Registrar</creator>
  <publisher scheme="NACO">Michigan State University</publisher>
  <subject scheme="NACO">Michigan State University. Office of the Registrar</subject>
  <description>Directory of local and permanent contact information for Michigan State University students. Updated each semester</description>
  <date qualifier="modified">2012-02-21</date>
  <coverage qualifier="spatial" scheme="LCNAF">East Lansing (Mich.)</coverage>
  <coverage qualifier="temporal">Spring Semester 1994 - Spring Semester 2012</coverage>
  <rights>Use of this public domain resource is unrestricted</rights>
  <relation qualifier="hasPart" msu-uahc:UA.6.7-A.2012.0019-001"label">Fall 2011</label>
  <key>FS11</key>
</relation>
  <relation qualifier="hasPart" msu-uahc:UA.6.7-A.2012.0021-001"label">Spring 2012</label>
  <key>SS12</key>
</relation>
</dublinCore>
```
**Object**: msu-uahc:RG-CollectionName:
**Name**: source
**Purpose**: This is a copy of the Dublin Core metadata data stream for the collection used to enable the display of the PDF links
**Example**: See msu-uahc:RG-CollectionName:DublinCore

**Object**: msu-uahc:RG-CollectionName:
**Name**: dcXSL
**Purpose**: This is the stylesheet that transforms the source data stream into the main Spartan Archive page
**Example**: see msu-uahc:CDTransform

**Object**: msu-uahc:RG-CollectionName:
**Name**: source
**Purpose**: This is the XML data definitions data stream for the collection
**Example**: see msu-uahc:DDCollectionName
**Object:** msu-uahc:RG-ATAccessionNumber-XXX  
**Name:** DublinCore  
**Purpose:** The Dublin Core metadata for this instance of the collection  
**Example:**

```xml
<dublinCore>
  <collectionname>Student Directory, Spring 2012</collectionname>
  <creator scheme="NACO">Michigan State University. Office of the Registrar</creator>
  <publisher scheme="NACO">Michigan State University</publisher>
  <subject>Student Records</subject>
  <subject scheme="NACO">Michigan State University. Office of the Registrar.</subject>
  <description>Directory of local and permanent contact information for MSU students for each semester</description>
  <date qualifier="issued(original)" scheme="W3CDTF">2012-01-27</date>
  <date qualifier="created(digital file)" scheme="W3CDTF">2012-01-27</date>
  <date qualifier="acquisitioned" scheme="W3CDTF">2012-02-03</date>
  <date qualifier="accessioned" scheme="W3CDTF">2012-02-21</date>
  <language scheme="iso63903">eng</language>
  <type scheme="DCMITYPE">dataset</type>
  <format>text/xml</format>
  <format qualifier="extent">25287312</format>
  <coverage qualifier="spatial" scheme="LCNAF">East Lansing (Mich.)</coverage>
  <coverage qualifier="temporal">Spring 2012</coverage>
  <rights>Use of this public domain resource is unrestricted.</rights>
  <relation>msu-uahc:StudentDirectory</relation>
</dublinCore>```
Object: msu-uahc:RG-ATAccessionNumber-XXX:
Name: premis
Purpose: This is Premis object metadata for this instance of a collection
Example:
<premis>
<object>
  <objectIdentifier>
    <objectIdentifierType>Spartan Archive Fedora</objectIdentifierType>
    <objectIdentifierValue>msu-uahc:UA.6.7-A.2012.0017-018</objectIdentifierValue>
  </objectIdentifier>
  <originalName>StudentDirectory.xml</originalName>
  <objectCategory>file</objectCategory>
  <significantProperties>
    <significantPropertiesType>structure</significantPropertiesType>
    <significantPropertiesValue>XML</significantPropertiesValue>
  </significantProperties>
  <significantProperties>
    <significantPropertiesType>content</significantPropertiesType>
    <significantPropertiesValue>data</significantPropertiesValue>
  </significantProperties>
  <objectCharacteristics>
    <compositionLevel>0</compositionLevel>
    <fixity>
      <messageDigestAlgorithm>MD5</messageDigestAlgorithm>
      <messageDigest>f7793b91e3b6f1fd84ab0be69c15dd8</messageDigest>
      <messageDigestOriginator>MSU-RO</messageDigestOriginator>
    </fixity>
    <size>29735103</size>
    <format>
      <name>text/xml</name>
    </format>
    <CreatingApplication>
      <creatingApplicationName>Custom Visual Basic Program</creatingApplicationName>
      <creatingApplicationVersion>1.0</creatingApplicationVersion>
      <dateCreatedByApplication>2012/06/22</dateCreatedByApplication>
    </CreatingApplication>
  </objectCharacteristics>
  <storage>
    <contentLocation>
      <contentLocationType>URI</contentLocationType>
      <contentLocationValue>data//records/UA.6.7_RegistrarsOffice/Serial_56_StudentDirectories/fs03</contentLocationValue>
    </contentLocation>
    <storageMedium>hard disk</storageMedium>
  </storage>
  <environment>
    <environmentCharacteristic>knownToWork</environmentCharacteristic>
    <environmentPurpose>render</environmentPurpose>
    <dependency>
      <dependencyName>XML Schema</dependencyName>
      <dependencyIdentifier>
        <dependencyIdentifierType>URI</dependencyIdentifierType>
        <dependencyIdentifierValue>schemas/studentDirectory.xsd</dependencyIdentifierValue>
      </dependencyIdentifier>
    </dependency>
    <software>
      <swName>Apache Tomcat</swName>
      <swVersion>7.022</swVersion>
    </software>
  </environment>
</object>
</premis>
<swType>Application Engine</swType>
</software>
<software>
  <swName>Fedora</swName>
  <swVersion>3.5</swVersion>
  <swType>Repository</swType>
</software>
<software>
  <swName>MySQL</swName>
  <swVersion>5.1</swVersion>
  <swType>Database Management System</swType>
</software>
<software>
  <swName>Apache Tomcat</swName>
  <swVersion>3.0</swVersion>
  <swType>Language</swType>
</software>
<software>
  <swName>Java</swName>
  <swVersion>6.025</swVersion>
  <swType>Language</swType>
</software>
</environment>

<relation>
  <relationType>structural</relationType>
  <relationshipSubType>is part of</relationshipSubType>
  <relatedObjectIdentification>msu-uahc:UA.6.7-StudentDirectory</relatedObjectIdentification>
</relation>

</object>
</premis>
Object: msu-uahc:RG-ATAccessionNumber-XXX:
Name: dataDictionary
Purpose: This is the XML data definitions data stream for the collection
Example: see msu-uahc:DDCollectionName
Object: msu-uahc:RG-ATAccessionNumber-XXX:
Name: source
Purpose: For those collections with PDF files this is an xml file listing all of the PDFs associated with this instance
Example:

```xml
<pdfList period="2000-2002" title="Academic Programs">
  <pdf>
    <link>http://spartanarchive.itservices.msu.edu:8080/fedora/objects/msu-uahc:UA.6.7-A.2012.0107-001/datastreams/PDFFile0/content</link>
    <size>28</size>
    <label><B>Table of Contents</B></label>
  </pdf>
  <pdf>
    <link>http://spartanarchive.itservices.msu.edu:8080/fedora/objects/msu-uahc:UA.6.7-A.2012.0107-001/datastreams/PDFFile1/content</link>
    <size>8</size>
    <label>Board of Trustees and Officers of the University</label>
  </pdf>
  <pdf>
    <link>http://spartanarchive.itservices.msu.edu:8080/fedora/objects/msu-uahc:UA.6.7-A.2012.0107-001/datastreams/PDFFile2/content</link>
    <size>48</size>
    <label>Mission Statement</label>
  </pdf>
  <pdf>
    <size>14</size>
    <label>University Organization</label>
  </pdf>
  <pdf>
    <size>8</size>
    <label>Academic Calendar 2000</label>
  </pdf>
  <pdf>
    <size>8</size>
    <label>Academic Calendar 2001</label>
  </pdf>
  <pdf>
    <size>99</size>
    <label>The Academic Program</label>
  </pdf>
  <pdf>
    <link>http://spartanarchive.itservices.msu.edu:8080/fedora/objects/msu-uahc:UA.6.7-A.2012.0107-001/datastreams/PDFFile7/content</link>
    <size>271</size>
    <label>Undergraduate Education</label>
  </pdf>
  <pdf>
    <link>http://spartanarchive.itservices.msu.edu:8080/fedora/objects/msu-uahc:UA.6.7-A.2012.0107-001/datastreams/PDFFile8/content</link>
    <size>274</size>
    <label>Graduate Education</label>
</pdfList>
```
General Information

The Colleges, Faculty, and Programs of Study

Faculty

The Honors College

College of Agriculture and Natural Resources

College of Arts and Letters

The Eli Broad College of Business and The Eli Broad Graduate School of Management

College of Communication Arts and Sciences

College of Education

College of Education
<label>College of Engineering</label>
</pdf>
<pdf>
<link>http://spartanarchive.itservices.msu.edu:8080/fedora/objects/msu-uahc:UA.6.7-A.2012.0107-001/datastreams/PDFFile19/content</link>
<size>275</size>
<label>College of Human Ecology</label>
</pdf>
<pdf>
<link>http://spartanarchive.itservices.msu.edu:8080/fedora/objects/msu-uahc:UA.6.7-A.2012.0107-001/datastreams/PDFFile20/content</link>
<size>185</size>
<label>College of Human Medicine</label>
</pdf>
<pdf>
<link>http://spartanarchive.itservices.msu.edu:8080/fedora/objects/msu-uahc:UA.6.7-A.2012.0107-001/datastreams/PDFFile21/content</link>
<size>244</size>
<label>James Madison College</label>
</pdf>
<pdf>
<link>http://spartanarchive.itservices.msu.edu:8080/fedora/objects/msu-uahc:UA.6.7-A.2012.0107-001/datastreams/PDFFile22/content</link>
<size>544</size>
<label>College of Natural Science</label>
</pdf>
<pdf>
<size>158</size>
<label>College of Nursing</label>
</pdf>
<pdf>
<link>http://spartanarchive.itservices.msu.edu:8080/fedora/objects/msu-uahc:UA.6.7-A.2012.0107-001/datastreams/PDFFile24/content</link>
<size>191</size>
<label>College of Osteopathic Medicine</label>
</pdf>
<pdf>
<link>http://spartanarchive.itservices.msu.edu:8080/fedora/objects/msu-uahc:UA.6.7-A.2012.0107-001/datastreams/PDFFile25/content</link>
<size>476</size>
<label>College of Social Science</label>
</pdf>
<pdf>
<link>http://spartanarchive.itservices.msu.edu:8080/fedora/objects/msu-uahc:UA.6.7-A.2012.0107-001/datastreams/PDFFile26/content</link>
<size>148</size>
<label>College of Veterinary Medicine</label>
</pdf>
</pdfList>
Object: msu-uahc:RG-ATAccessionNumber-XXX:
Name: xsl
Purpose: This is the stylesheet that controls the display of the PDF table of contents for this instance of the collection
Example: See msu-uahc:LLTransform
**Object:** msu-uahc:DDCollectionName:

**Name:** dataDictionary

**Purpose:** This is the XML data definitions data stream for the collection

**Example:**

```xml
<fieldDefinitions>
  <title>Student Directory</title>
  <date>2012-04-04</date>
  <fields>
    <field>
      <name>Semester</name>
      <definition>Semester for which this information is applicable. Semesters are abbreviated with 2 letters followed by 2 numbers. The first two letters correspond to the semester code (FS for Fall Semester, SS for Spring Semester, and US for Summer Semester), the two numbers are the last two digits of the year. For example, Spring 2008 would be SS08.</definition>
    </field>
    <field>
      <name>Student ID</name>
      <definition>ID assigned to student</definition>
    </field>
    <field>
      <name>Name</name>
      <definition>Student's first and last names as used at Michigan State University</definition>
    </field>
    <field>
      <name>Level</name>
      <definition>Student's general classification, taken from list to right</definition>
    </field>
    <field>
      <name>Major</name>
      <definition>Student's major course of study. Use abbreviation taken from list to right, such as ART EDUC for Art Education.</definition>
    </field>
    <field>
      <name>Class</name>
      <definition>Student's class, used in certain combinations with the level, taken from list to right</definition>
    </field>
    <field>
      <name>Local Line 1</name>
      <definition>First line of student's local address while attending the university, generally the street address</definition>
    </field>
    <field>
      <name>Local Line 2</name>
      <definition>Second line of student's local address while attending the university, generally supplemental address information such as post office box</definition>
    </field>
    <field>
      <name>Local City</name>
      <definition>City of student's local address while attending the university</definition>
    </field>
    <field>
      <name>Local State</name>
      <definition>State of student's local address while attending the university; two-letter US Postal Service format, such as MI for Michigan</definition>
    </field>
    <field>
```

```xml
```
<name>Local Zip</name>
<definition>Zip code at student's local address</definition>
</field>

<field>
<name>Local Phone</name>
<definition>Phone number student used while attending the university; only area code will be displayed</definition>
</field>

<field>
<name>MSU Email Address</name>
<definition>E-mail address assigned to the student by the university; may no longer be valid</definition>
</field>

<field>
<name>Permanent Line 1</name>
<definition>First line of student's permanent address while attending the university, generally street address</definition>
</field>

<field>
<name>Permanent Line 2</name>
<definition>Second line of student's permanent address while attending the university, generally supplemental address information such as post office box</definition>
</field>

<field>
<name>Permanent City</name>
<definition>City of student's permanent address while attending the university</definition>
</field>

<field>
<name>Permanent State</name>
<definition>State of student's permanent address while attending the university; two-letter US Postal Service format, such as MI for Michigan</definition>
</field>

<field>
<name>Permanent Zip</name>
<definition>Zip/postal code at student's permanent address</definition>
</field>

<field>
<name>Permanent Country</name>
<definition>Country of student's permanent address while attending the university</definition>
</field>

<field>
<name>Permanent Phone</name>
<definition>Phone number student provided as his/her permanent contact number; only area code will be displayed</definition>
</field>

<overview>Student contact information is controlled by FERPA privacy regulations. As such, students can choose not to provide contact information in which case the fields will be blank, or will not show. If the student chose to participate but opted out of publishing select fields, those fields will contain asterisks(*)</overview>
Object: msu-uahc:RG-Collections
Name: defaults
Purpose: These are the values that control ingest and metadata generation for this collection.
Example:
<defaults>
  <entry qualifier="" scheme="" value="Collection Short Name">StudentDirectory</entry>
  <entry qualifier="" scheme="" value="Contributor">Gadson, Damion</entry>
  <entry qualifier="" scheme="" value="Publisher">Michigan State University</entry>
  <entry qualifier="" scheme="" value="Rights">Use of this public domain resource is unrestricted</entry>
  <entry qualifier="" scheme="" value="Subject">course descriptions</entry>
  <entry qualifier="" scheme="" value="Submission Name">studentDirectory.xml</entry>
  <entry qualifier="" scheme="" value="Record Group Description">Office of the Registrar</entry>
  <entry qualifier="" scheme="" value="Record Group Short Name">RegistrarsOffice</entry>
  <entry qualifier="" scheme="" value="Type">dataset</entry>
  <entry qualifier="" scheme="" value="Creator">Michigan State University. Office of the Registrar</entry>
  <entry qualifier="" scheme="" value="Collection Name">Student Directory</entry>
  <entry qualifier="" scheme="" value="Record Group">UA.6.7</entry>
  <entry qualifier="" scheme="" value="Language">eng</entry>
  <entry qualifier="" scheme="" value="Coverage">East Lansing (Mich.) 48823</entry>
</defaults>
Object: msu-uahc:CDTransform:
Name: dcXSL
Purpose: This is the stylesheet that transforms the Dublin Core data stream a collection web page
Example:

```xml
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <xsl:variable name="targetURL">
      <xsl:choose>
        <xsl:when test="/dublinCore/title='Academic Programs'">/GeneralQuery/APEntry.html</xsl:when>
        <xsl:otherwise>/GeneralQuery/GeneralQuery?type=queryBuilder&amp;</xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <html>
      <head>
        <title>
          <xsl:apply-templates select="/dublinCore/title"/>
        </title>
        <link href="http://spartanarchive.itservices.msu.edu:8080/collection.css" rel="stylesheet" type="text/css"/>
      </head>
      <body>
        <div class="title">
          <h1>Spartan Archive</h1>
          <h3>An Electronic Records Archive at Michigan State University</h3>
        </div>
        <div style="font-size: 12px;font-weight: lighter;margin-left: 80px;margin-bottom:8px">
          <a href="/fedora/objects/msu-uahc:SpartanArchive/methods/msu-uahc:ex3SDef/getContent">Home</a>
        </div>
        <div id="content">
          <h3>Current Collection: <span class="value">
              <xsl:apply-templates select="/dublinCore/title"/>
            </span></h3>
          <div class="item">
            <span class="key">Creator: </span>
            <xsl:apply-templates select="/dublinCore/creator"/>
          </div>
          <div class="item">
            <span class="key">Last Updated: </span>
            <xsl:apply-templates select="/dublinCore/date[@qualifier='modified']"/>
          </div>
          <div class="item">
            <span class="key">Period Covered: </span>
            <xsl:apply-templates select="/dublinCore/coverage[@qualifier='temporal']"/>
          </div>
          <div class="desc">
            <xsl:apply-templates select="/dublinCore/description"/>
          </div>
          <div style="margin-top: 8px;margin-bottom: 4px; font-weight:bold">Usage Information:</div>
          <div style="margin-top: 8px;margin-bottom: 4px; font-weight:bold">Usage Information:</div>
          <div style="padding-right: 5px;">"</xsl:variable>
        </div>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```
Object: msu-uahc:DDTransform:
Name: xsl
Purpose: This is the stylesheet that produces the PDF link list for the collection
Example:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:variable name="targetURL">
    <xsl:choose>
      <xsl:when test="/dublinCore/title='Academic Programs'">
        /GeneralQuery/APEntry.html?
      </xsl:when>
      <xsl:otherwise>/GeneralQuery/GeneralQuery?type=queryBuilder&amp;
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <html>
    <head>
      <title>
        <xsl:apply-templates select="/dublinCore/title"/>
      </title>
      <link href="http://spartanarchive.itservices.msu.edu:8080/collection.css" rel="stylesheet" type="text/css"/>
    </head>
    <body>
      <div class="title">
        <h1>Spartan Archive</h1>
        <h3>An Electronic Records Archive at Michigan State University</h3>
      </div>
      <div style="font-size: 12px;font-weight: lighter;margin-left: 80px;margin-bottom:8px">
        <a href="/fedora/objects/msu-uahc:SpartanArchive/methods/msu-uahc:ex3SDef/getContent">Home</a>
      </div>
      <div id="content">
        <h3>Current Collection: <span class="value">
          <xsl:apply-templates select="/dublinCore/title"/>
        </span></h3>
        <div class="item">
          <span class="key">Creator: </span>
          <xsl:apply-templates select="/dublinCore/creator"/>
        </div>
        <div class="item">
          <span class="key">Last Updated: </span>
          <xsl:apply-templates select="/dublinCore/date[@qualifier='modified']"/>
        </div>
        <div class="item">
          <span class="key">Period Covered: </span>
          <xsl:apply-templates select="/dublinCore/coverage[@qualifier='temporal']"/>
        </div>
        <div class="description">
          <xsl:apply-templates select="/dublinCore/description"/>
        </div>
        <div style="margin-top: 8px;margin-bottom: 4px; font-weight:bold">Usage Information:</div>
        <xsl:variable name="CompressedTitle" select="concat(substring-before(/dublinCore/title, ' '), substring-after(/dublinCore/title, ' '))"/>
      </div>
    </body>
</xsl:stylesheet>
```
<div>
<p style="margin-left:80px">
<xsl:choose>
  <xsl:when test="/dublinCore/title!='Academic Programs'">
    <a href="{$targetURL}&amp;name={/dublinCore/title}&amp;key=all">Search the complete collection.</a> or choose a period below.
  </xsl:when>
  <xsl:otherwise>
    Choose a period below.
  </xsl:otherwise>
</xsl:choose>
</p>
<div class="instances">
<xsl:for-each select="dublinCore/relation">
  <xsl:if test="pdf='true'">
    <div>
      <span class="period">
        <xsl:value-of select="./label"/>
      </span>
      <span class="pdf">
        <a href="http://spartanarchive.itservices.msu.edu:8080/fedora/objects/{normalize-space(text())}/methods/msu-uahc:ex3SDef/getContent">PDF</a>
      </span>
    </div>
  </xsl:if>
</xsl:for-each>
</div>
</div>
</body>
</html>
Object: msu-uahc:LLTransform
Name: xsl
Purpose: This is the XSL stylesheet for transforming the PDF list (source) to a web page
Example:

```xml
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="pdfList">
    <html>
      <head>
        <title>Spartan Archive</title>
      </head>
      <body>
        <h1>
          <xsl:apply-templates select="@title"/> for <xsl:apply-templates select="@period"/>
        </h1>
        <xsl:apply-templates select="pdf"/>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="pdf">
    <div>
      <span style="display:inline-block;width:80px;font-size:75%; padding-right:5px">PDF</span>
      <xsl:value-of select="size"/>K
    </div>
  </xsl:template>
</xsl:stylesheet>
```
How Fedora Objects and Data Streams are Used in the Spartan Archive

Overview

Fedora is used to store all metadata for the Spartan Archive. This includes Premis and Dublin Core information for each collection and instance, along with data definitions and organizing information. Each object is identified by a permanent ID or PID. All Spartan Archive PIDs use the prefix--msu-uahc:.

Each object contains one or more data streams. Data streams are Fedora’s mechanism for storing data and metadata. Each object is required to contain a ‘dc’ data stream which is a limited-functionality Dublin Core metadata file containing identifier information for Fedora. Because it is limited in its functionality, many of the data objects in the Spartan Archives implementation of Fedora also contain a ‘Dublin Core’ data stream which has a richer set of metadata objects. Similarly objects representing instances of collections also contain a ‘Premis’ data stream which contains premis metadata for the object. Finally any Spartan Archive object which is expected to provide access to its data streams will contain one or more xsl stylesheet data streams which instruct the browser on how to display the data. (For the most part these style sheets are common to multiple objects and are stored in a single object in Fedora. They are then referenced in the individual Spartan Archives objects.)

Another data stream used in most Spartan Archive objects is the ‘RELS-EXT’ which is intended to document relationships between objects. At this time the only use for these data streams is to associate objects with their content model and related display methods.

Finally Fedora maintains a set of objects used for providing specialized access to data and metadata. These are the content models and the associated display methods. The Spartan Archive currently uses a single content model which supports xslt transformations. In the future additional content models will be added to support access to other types of digital objects.

The following diagram displays the relationship between the objects in the Spartan Archive Fedora system.
Spartan Archive Objects

The entry point to the Spartan Archives is the object msu-uahc:SpartanArchive. It contains 3 data streams: ‘source,’ ‘xsl,’ and ‘defaultTypes.’ The ‘source’ data stream contains information about each collection in the archive. This includes basic metadata, schedule information and some processing aids. (See the data stream listings for more detail.) The ‘xsl’ data stream contains the stylesheet needed to present the first page in the Spartan Archive access system (including links to the specific collections). The ‘defaultTypes’ data stream contains all of definitions of configurable items in the archive system. The actual configuration are maintained in each separate collection.
The Spartan Archive Welcome Page

The content models "GetContent" method generates the following page.

Michigan State University's Home for Digital Preservation
The Spartan Archives
Collection Objects:

Each Collection within the Archive has its own object identified by a PID with the form:

msu-uahc:RG-CollectionName

where RG is the record group. So for example the PID for the Academic Programs
collection is msu-uahc:UA.6.7-AcademicPrograms.

Each collection object has 6 data streams (along with the default ‘dc’ data stream).
Examples can be found in the Data Stream section. These are:

<table>
<thead>
<tr>
<th>Data Stream</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELS-EXT</td>
<td>Connects the object with the content model for displaying the metadata</td>
</tr>
<tr>
<td>source</td>
<td>These are the data definitions for the collection</td>
</tr>
<tr>
<td>xsl</td>
<td>The stylesheet for formatting the data definitions.</td>
</tr>
<tr>
<td>Dublin Core</td>
<td>The Dublin Core metadata for the collection</td>
</tr>
<tr>
<td>dcXSL</td>
<td>The stylesheet for formatting the Dublin Core. This stylesheet is used for all collections and is referenced here from: <a href="http://fedora.ats.msu.edu:8080/fedora/get/msu-uahc:CDTransform/dcXSL">http://fedora.ats.msu.edu:8080/fedora/get/msu-uahc:CDTransform/dcXSL</a> This is used to display the page for each collection including creating the links to the specific query pages. See examples below:</td>
</tr>
<tr>
<td>defaults</td>
<td>These contain the configuration items specific to this collection</td>
</tr>
</tbody>
</table>

Each collection page is generated using the ‘GetContent’ method of its content model, using the DublinCore xml data stream transformed according to its dcXSL stylesheet to produce a page like the one below.
Instance Objects

Each Instance within the Archive has its own object identified by a pid with the form:

\[
\text{msu-uahc:RG-ATAccessionNumber-XXX}
\]

where RG is the record group. ATAccessionNumber is the number assigned in Archivist’s toolkit for this submission and XXX is a sequential number beginning a 001. This number is used when more than one instance is accessioned at one time. As an example of that the twenty years of Schedule of Courses submissions were accessioned at once. As so each instance is differentiated by the suffix. Thus a sample instance object pid might be msu-uahc:UA.6.7-A.2012.0021-001.

Each instance object has 5 data streams (along with the default ‘dc’ data stream). Examples can be found in the Data Stream section. These are

<table>
<thead>
<tr>
<th>Data Stream</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELS-EXT</td>
<td>Connects the object with the content model for displaying the metadata</td>
</tr>
<tr>
<td>Premis</td>
<td>The premis metadata for the instance</td>
</tr>
<tr>
<td>Dublin Core</td>
<td>The Dublin Core metadata for the collection</td>
</tr>
</tbody>
</table>
Data Stream | Purpose
--- | ---
source | The data definitions for the collection. These not actually stored in each instance object. Rather they exist as data streams within a specific data definition object, and are referenced within the instance object by their Fedora url. For example: http://spartanarchive.msu.edu:8080/fedora/objects/msu-uahc:DDAcademicPrograms/datastreams/source/content
xsl | This is the stylesheet for the data definitions. It is referenced from http://fedora.ats.msu.edu:8080/fedora/get/msu-uahc:DDTransform/xsl

### Shared Objects

There a number of shared objects, that is secondary objects that support multiple primary objects. These can be classified into three groups:

- Content Models and Services
- Data Definitions
- Stylesheets.

#### Content Models and Services

Currently only one content model is used in Spartan Archive. Because the only current content in the Spartan Archive is the records based data from the Office of the Registrar, the content model only needs to support displaying the metadata for the collections and their instances. All other data is provided to researchers via the Java backend. These objects are:

<table>
<thead>
<tr>
<th>Object</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>msu-uahc:ex3CModel</td>
<td>Content model that supports applying a stylesheet against metadata (in xml form) to display a web page. The CModel object has a RELS-EXT data stream that ties it to the service definitions that it supports. In this case those are msu-uahc:ex3CSDef and demo:ex3CSDef.</td>
</tr>
<tr>
<td>Object</td>
<td>Purpose</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>msu-uahc:ex3CSDef</td>
<td>Service definition object that supports displaying the ‘source’ data streams that define the data definitions via the xsl stylesheet data source. SDef objects contain a MethodMap data stream that names the services provided via this definition. In this case it is getContent.</td>
</tr>
<tr>
<td>msu-uahc:ex3CSDep</td>
<td>Service deployment object that maps the data sources (in this case the ‘source’ and ‘xml’ data streams) to a web service that actually performs the conversion. SDep objects have a RELS-EXT data stream that ties the deployment back to both a SDef definition, and the content models that use the service. In addition they have a MethodMap data stream that does the actual mapping between the inputs and the web service, a DSINPUTSPEC data stream that defines the types of the input and a WSDL data stream that is required by the web service.</td>
</tr>
</tbody>
</table>

Data Definitions

Each collection has a data definition object that defines each data item in the collection. The object contains a single data stream ‘source’ which is an xml representation of the data dictionary. This data stream is referenced in each instance object. (See the table in Instance Objects above for an example.) These definitions are used for two purposes in the system. First they provide the the list of fields available for query and second they are used to popup definitions for each of those fields.

Data Definition objects have PIDs of the form msu-uahc:DDCollectionName. So the Academic Programs collection data definition object would have a PID of msu-uahc:DDAcademicPrograms.

Stylesheets

In order to have a consistent presentation for collections, stylesheets are shared across the Spartan Archive system.

The collection definition stylesheet is contained in msu-uahc:CDTransform. This has a single data stream called ‘dcXSL.’ This data stream transforms a collections Dublin
Core metadata into the collection webpage. This data stream is referenced from each collection object.
Java programs used in the Spartan Archive System

There are approximately 20 Java programs in use in the Spartan Archive system. These include large web applications, batch ingest programs, libraries and several special purpose utility programs. The large programs and the batch ingest programs have been documented elsewhere. This document will list and describe all of the programs, referring the reader to other documentation as needed.

Web Applications:

GeneralQuery documented in Presenting Database Objects to Researchers.doc. This is the web application that provides access to the Spartan Archives for researchers. It works in conjunction with the Fedora only front end of the Spartan Archives application (Fedora Objects and Data Streams.doc).

SAControl documented in SAControl Documentation.doc. This provides the interface for Administrators to control the ingest process and to update information that controls the Spartan Archives processing. Administrator use of the program is documented in the Procedures section of the Spartan Archive directory.

PDFPuller is documented in PDFPuller Documentation.doc. This application provides a way for administrators to ingest PDFs for the Academic Programs and Course Descriptions collection into Fedora and the Digital Vault. Administrator use is documented in the Procedures section of the Spartan Archive directory.

In each of these applications the web pages are presented using a combination of AJAX and JSP techniques. These display the data that is extracted from the backend data base and Fedora data stores in an intelligible way. The data that they use is generated by the Java backend programs listed above and delivered to the web pages as xml (for AJAX) pages, and request attributes (for JSP) pages. This data is documented in Spartan Archives Data Sources.doc.

Batch Ingest Programs

These programs are documented in Spartan Archives Ingest Process.doc. There purpose is to discover new submissions, validate them, and ingest them into Fedora and the digital vault. The specific programs are listed below in order of use. All of these will send emails to the administrator when events of special interest occur.

GetIngestFiles. This discovers the new submissions, validates them for integrity, moves them to a quarantine area and sets a status marker to indicate that the submission is ready for submission. At its conclusion the program emails the administrator to notify him or her that a new file has been delivered. The administrator will use SAControl (described above) to authorize the file’s ingest.

The programs in the second half of the ingest process are run from a shell script runPrograms.sh found in archives/bin.
Ingest2. This copies the submission to the digital vault, creates a Fedora Object for the submission and updates the collection and repository objects. At the end Ingest2 updates the status marker.

A subset of the batch ingest programs are the database load programs. These are:

- AcadXMLToDatabase: loads the Academic Programs Database
- CDXMLToDatabase: loads the Course Descriptions Database
- SDXMLToDatabase: loads the Student Directory Database
- SOCMXMLToDatabase: loads the Schedule of Courses database
- CreateLookupLists: extracts data from the above databases to use as aids to researchers who are building queries.

**Libraries**

These are programs that are used as components of other programs. They are listed below in alphabetical order.

EventStatus. This is documented in *Spartan Archives Ingest Process.doc*. This manages all access and update to the submissionstatus and saevents tables in the spartanarchive database. These tables store the status and events that provide information to administrators and control the processing of the ingest files.

fedoraAccess. This is documented in *Spartan Archives Ingest Process.doc*. This manages all access and update to the Fedora system for those programs that use it.

FedoraDefaults. This is documented in *Spartan Archives Ingest Process.doc*. This manages the access and update to all of the parameters that control the processing of ingest files and the creation of metadata (Dublin Core and Premis).

Mail. This provides a convenient way for the batch ingest and validation (see below) programs to notify administrators that an event of interest has happened. This works by creating a subprocess that calls the Linux mailx program.

**Utility Programs**

These are programs that either provide a specific production purpose unrelated to the above programs or can be run to deal with specific issues.

Two of these programs are concerned with verifying the integrity of the ingested files. They both use an MD5 utility from [http://twmacinta.com/myjava/fast_md5.php](http://twmacinta.com/myjava/fast_md5.php) to calculate checksums that are then compared to previously calculated and stored values. If any problems are found they use the Mail library to notify administrators.

ValidateAccessions. This program validates files that were ingested manually using the Duke Data Accessioner program. This creates manifest xml files that among other things documents the location and checksums of all of the files ingested. This file is used to locate the ingested files. The program
recalculates the MD5 checksum for each file and compares it to the stored value. Any error is logged internally and a single email is sent to the administrator to say that there is either no problem, or provides a list of all problems found.

ValidateFedoraFiles. This program validates the files that were automatically ingested as part of the ingest process described above. Each of these ingests resulted in the creation of a Premis data stream in Fedora. This program walks all of these datastreams, calculates new checksums and compares them to the value stored in Premis.

Other utility programs are used in an as needed basis. These include the following:

SetPreferences. There are a number of preferences that help to control the ingest process relating to locations and logins (see Changing Ingest Preferences.docx). SetPreferences will add or modify these preferences. Using the report option it will display all of the preferences saved.

CreateDBFromCSV. This program takes a spreadsheet (data template.csv) as input and creates a database in fedora-db.ats.msu.edu, and collection records in Fedora for a new collection. This is used to create a new collection for the archive. In addition to this a new database load program must be written (based on existing load programs) and added to runPrograms.sh.

FixPremis. This is used to make batch changes to the Premis data streams in Fedora. Anytime there is a need to correct or modify the data stream in an identifiable class of Fedora objects, this program can be used. A new function must be written to correct the specific problem and logic may have to be added to restrict the action to the specified class of objects. There are several functions in the program that can be used as samples to guide the creation of these functions.

validateSchema. This program can be used to investigate problems with an ingest file. It uses the xml schema to verify that the file is what is expected.
PDFPuller Documentation

Overview

The Office of the Registrar publishes PDF versions of the Academic Programs and the Description of Courses documents independently of providing the database dumps of their records to the Archive. In addition these PDFs are not sent to the Archives, rather the Archives must pull them from the web site in which they are published. Finally rather than a single PDF file file, each document consists of dozens or hundreds of individual PDFs.

In order to avoid having the Archive Administrator pull each of these PDFs and add them to Fedora individually, the PDFPuller program was developed to automate the process.

The PDF Puller is implemented as a web application. To use the application the Administrator chooses which collection is desired and enters the identification of the PDF site. (This must match exactly what the Registrar has posted on its site. The URLs in question are: Academic Programs- http://www.reg.msu.edu/UCC/APYearIndex.asp and for Course Descriptions- http://www.reg.msu.edu/UCC/DescYearIndex.asp) A single Java Servlet is used to coordinate between the services requested by the administrator and the Java classes that service the requests. This is illustrated below:
The web pages are pure Java Server Pages (JSP).

Overview

The core of the application is the Java Servlet PDFSource.java. Like all of the Servlets in the Spartan Archive application this class serves as a traffic director. It determines request and routes it to the appropriate working class.

Requests come to PDFSource via urls of the form:

http://fedora.ats.msu.edu:8080/PDFPuller/PDFPuller?type=getPDFs&collection=Academic+Programs&period=2010-2011

<table>
<thead>
<tr>
<th>Request Type</th>
<th>Page Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>getPDFs</td>
<td>Spartan Archives Current Submissions</td>
</tr>
<tr>
<td>updateFedora</td>
<td>Spartan Archive Event Log</td>
</tr>
</tbody>
</table>

Individual Pages

PDFPuller 1

This page is initially built as shown below as a pure HTML form. Its only purpose is to collect the information on the type of collection to be pulled and the period for which it pulled.
The name entered for the set of PDFs that are pulled must match that used at the Office of the Registrars site (listed above) exactly since this program uses those names to determine what to pull.

When the administrator has entered the information and clicked on the submit button, PDF source is sent a request like:

http://fedora.ats.msu.edu:8080/PDFPuller/PDFPuller?type=getPDFs&collection=Academic+Programs&period=2010-2011

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>This defines the request</td>
</tr>
<tr>
<td>collection</td>
<td>This is the kind of collection PDFs requested. Currently this can only be Academic Programs or Course Descriptions</td>
</tr>
<tr>
<td>period</td>
<td>This is the name of the collection link at the Registrar’s Office web site. It will be massaged into a regular form and used in building the fedora files and determining the location of the PDFs in the digital vault.</td>
</tr>
</tbody>
</table>

When PDFSource receives this request it calls PDFProcessor to open the Registrar’s Office web page, determine the link to the pdf files, read that page and return a list of all
of the names of the PDFs and links to the individual files. This list is used to set the pdf request attribute in the page that is returned.

Because the web pages used do not use a uniform layout, parsing the pages can be difficult. To overcome this problem PDFPuller uses a utility class HtmlReader to read in the page and provide access to it in a more useable logical line by logical line form.

At the end PDFSource causes displayPDFList.jsp to be displayed with the list of PDFs found.

Web Page used: displayPDFList.jsp
Database/Tables used: None
Javascript files: None,
Java classes used:

<table>
<thead>
<tr>
<th>Class</th>
<th>Called via</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDFSource.java</td>
<td>All interactions</td>
<td>Sets the pdf request attribute with the list of links from PDFProcessor</td>
</tr>
<tr>
<td>PDFProcessor.java</td>
<td>type=getPDFs</td>
<td>Retrieves a list of all of the linked pdf files for the collection and period requested</td>
</tr>
<tr>
<td>HtmlReader.java</td>
<td>type=getPDFs</td>
<td>This is a utility class that reads the contents of a web page and returns it in strictly defined lines that are easy to use in PDFProcessor.</td>
</tr>
</tbody>
</table>

PDFPuller 2

This page is built as shown below as via JSP. This gives the administrator the chance to review some of the PDFs that will be used to ensure that they are the correct files. If so he or she will determine the appropriate Accession Number from Archivist Toolkit, enter it with an appropriate extension (for example if the Accession Number is A.2012.0343 the value to enter is A.2012.0343-001) and press submit to ingest the files. To do this it sends a request like

http://fedora.ats.msu.edu:8080/PDFPuller/PDFPuller

via a POST call.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>This defines the request</td>
</tr>
<tr>
<td>collection</td>
<td>This is the kind of collection PDFs requested. Currently this can only be Academic Programs or Course Descriptions</td>
</tr>
<tr>
<td>newPeriod</td>
<td>This is the name of the collection link at the Registrar’s Office web site. It will be massaged into a regular form and used in building the fedora files and determining the location of the PDFs in the digital vault.</td>
</tr>
<tr>
<td>newPid</td>
<td>This is core of the identification number that will be used to create the Fedora object. To create that pid the program will append a “msu-uahc:UA.6.7-” to the front of the parameter</td>
</tr>
<tr>
<td>links</td>
<td>This is the list of pdf names and links that were displayed on the page.</td>
</tr>
</tbody>
</table>

When PDFSource receives this request it massages the period to put into a standard form (either year or year-year in full 4 digit form), sets the location in the digital vault to
store the pdfs, then asks PDFProcessor to pull the PDFs and store them appropriately. It then calls FedoraBuilder to create the fedora objects and data streams for the PDFs.

Finally it collects information about what was done and presents it to the user via resultPage.jsp.

Web Page used: resultPage.jsp  
Database/Tables used: None  
Javascript files: None  
Java classes used:

<table>
<thead>
<tr>
<th>Class</th>
<th>Called via</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDFSource.java</td>
<td>All interactions</td>
<td>Calls PDFProcessor to store the pdf files and FedoraBuilder to create the fedora objects and data streams</td>
</tr>
<tr>
<td>PDFProcessor.java</td>
<td>type=updateFedora</td>
<td>Takes the list of links from PDFSource, pulls the pdfs, stores them in the digital vault and returns a list of Files for use by FedoraBuilder</td>
</tr>
<tr>
<td>FedoraBuilder.java</td>
<td>type=updateFedora</td>
<td>Builds the metadata files for the Fedora PDF object (including Dublin Core and Premis) then calls FedoraAccess to build the object and add the data streams (metadata and PDFs)</td>
</tr>
<tr>
<td>FedoraAccess/java</td>
<td>type=updateFedora</td>
<td>This is a general purpose utility class located in the FedoraAccess library. It is used here to add an object to Fedora and update its data streams.</td>
</tr>
</tbody>
</table>
Spartan Archive Ingest Process

Overview

There are 7 steps to the Spartan Archive ingest process:

1. Handle drop off files. In this step the system identifies any files found in the drop off area. If they are known files, the system will validate their contents (XML validation) and verify checksums. In any case they are moved from the drop off area to the quarantine area. An archivist is notified via email that the files exist.

2. The archivist uses the Spartan Archive Administrative Interface to determine the fate of the submitted file. He or she may reject the file, pend the file or approve it. If the file is approved, the archivist will provide an Accession Number and add the accession to Archivist Toolkit. (To support this last activity the Administrative Interface provides an Archivist Toolkit import file for download.)

3. If the file has been approved, appropriate metadata is extracted and a new object referring to that file is added to Fedora. The file is then copied to its final location in the records directory.

4. Any new submissions found will be loaded to the Spartan Archive database.

5. Once the file is acted on by the Archivist in step 2, files in the drop off area will be deleted.

6. Rejected files will be deleted from quarantine, along with any files that have been moved to the records directory and added to the database.

7. Notify the archive of any overdue submissions.

There is a separate directory for each office that provides files to the archive in the drop off area. Only that office and the archive have the permissions to write to or read from the directory. Currently offices use SFTP to send files to this directory. In the future it is expected that there will be a more user-friendly web interface to this process.

Controlling the process.

Communication between the steps and the archivists are handled via 3 mechanisms. First archivists are notified by email whenever the state changes. In addition there a set of events and status's are maintained in a database. The events are used to give the archivist more detailed information about what has happened to a submission than the email can reasonably provide. The statuses give the current status of each submission at the end of each step. Statuses provide information to the archivist and direct the actions of the ingest process.

In addition there are two sets of preferences: One is general and is maintained in the file system, the other is related to specific collections and is maintained in Fedora. The general set of preferences controls file locations, database access and email addresses. The collection specific preferences control metadata construction and collection specific locations.
Email

Email is sent to the archivists when:

- A new submission is found in the drop off area and copied to the quarantine area.
- When any problems are discovered in the process of loading Fedora, moving the submission to the records area, or in updating the database.
- When the submission has successfully completed the ingest process.
- When a submission is overdue.

The addresses to send the email are maintained in the file system preferences in the format address1, address2, ...

Events

Events are maintained in the spartanarchive database on MySQL. At each step of the ingest events are added. As such they provide the history for the ingest. For example when a submission is validated an event is added to document the success or failure of the activity. These events are displayed to the archivist in the Administrative Interface in two places. When dealing with a submission, all events associated with that submission are displayed when the submission is selected. In addition a second page shows all of the events to date.

Statuses

Statuses are also maintained in the spartanarchive database on MySQL. There is a single status for each submission. This status is used to communicate the state of the submission at any time. Statuses include:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
<th>Possible Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown file</td>
<td>A file was found in the drop off area that could not be identified as a submission from this office</td>
<td>Reject: to delete the file. Pend: to further investigate the submission. Contact the office to identify the file. If legitimate ask them to rename it appropriately and then reject the current file.</td>
</tr>
<tr>
<td>Status</td>
<td>Meaning</td>
<td>Possible Actions</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>In quarantine</td>
<td>This is a known file.</td>
<td>Check the events to ensure that it is valid and complete. Approve to move on to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the next steps of the ingest process.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pend to further investigate.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reject delete the file without ingest.</td>
</tr>
<tr>
<td>Approved for Ingest</td>
<td>The submission has been approved and will go on to the</td>
<td>Once the investigation has been completed either Approve or Reject. The ingest</td>
</tr>
<tr>
<td></td>
<td>next step of the process.</td>
<td>process will ignore this submission until that time.</td>
</tr>
<tr>
<td>Pended</td>
<td>The submission has been pended for investigation</td>
<td></td>
</tr>
<tr>
<td>Reject</td>
<td>The submission has been rejected</td>
<td>The ingest process will delete all copies of this submission.</td>
</tr>
<tr>
<td>Added to Fedora</td>
<td>Ingest2 has successfully added the submission to Fedora</td>
<td>The database loads programs read this to kick off the load process.</td>
</tr>
<tr>
<td></td>
<td>and moved the submission to the records area.</td>
<td></td>
</tr>
<tr>
<td>Added to the Database. Ingest</td>
<td>All ingest processes are complete and the collection</td>
<td></td>
</tr>
<tr>
<td>Complete</td>
<td>instance is ready for researcher access.</td>
<td></td>
</tr>
</tbody>
</table>

**Preferences**

General preferences are used to provide standard information to the ingest programs. They are stored in `/home/archives/.java/.userPrefs/prefs.xml`. These are used in the Java programs to identify file locations, database locations and access, and who should be receiving email notifications. Sample values are:
Collection specific preferences are located in Fedora objects. `msu-uahc:SpartanArchive` contains the XML data stream “source.” Each `msu-uahc:DDCollectionName` (where `CollectionName` is the actual collection name) object contains an XML data stream called “defaults.”

**msu-uahc:SpartanArchive:source**

<table>
<thead>
<tr>
<th>Element</th>
<th>Parent</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>office</td>
<td>None</td>
<td>The office that owns the collection. The shortName attribute is the name of the directory in which the office drops its files</td>
</tr>
<tr>
<td>collection</td>
<td>office</td>
<td>Each office has one or more collection elements. Examples include Academic Programs, and Student Directories</td>
</tr>
<tr>
<td>title</td>
<td>collection</td>
<td>The name of the collection provided by the office. This is used by GetIngestFiles to identify the submissions that appear in the drop off area</td>
</tr>
<tr>
<td>pid</td>
<td>collection</td>
<td>This is Fedora’s identifier for the collection object</td>
</tr>
<tr>
<td>creator</td>
<td>collection</td>
<td>The office that created the file.</td>
</tr>
<tr>
<td>Element</td>
<td>Parent</td>
<td>Meaning</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>lastUpdate</td>
<td>collection</td>
<td>The last time this collection or one of its elements was updated</td>
</tr>
<tr>
<td>periodCovered</td>
<td>collection</td>
<td>The period that this collection covers</td>
</tr>
<tr>
<td>start</td>
<td>periodCovered</td>
<td>The start of the period, for current collection this is a semester (Fall 2000) or a year (1992)</td>
</tr>
<tr>
<td>end</td>
<td>periodCovered</td>
<td>The end of the period. See above for format.</td>
</tr>
<tr>
<td>description</td>
<td>collection</td>
<td>The description of the collection. Used in building the metadata in Ingest2.</td>
</tr>
<tr>
<td>key</td>
<td>collection</td>
<td>The key to this collection instance used by the access system</td>
</tr>
<tr>
<td>submissionSchedule</td>
<td>collection</td>
<td>The schedule for delivery for instances of this collection.</td>
</tr>
<tr>
<td>submission</td>
<td>submissionSchedule</td>
<td>An instance of the schedule. For annual instances there will only be one of these. For semester based instances there will be one for each semester. The attribute period identifies each.</td>
</tr>
<tr>
<td>dueDate</td>
<td>submissionSchedule</td>
<td>When the instance is first expected.</td>
</tr>
<tr>
<td>alarmDate</td>
<td>submissionSchedule</td>
<td>When to notify the archive if the submission has not yet arrived.</td>
</tr>
</tbody>
</table>
There is one entry element for each default. Each has 3 attributes:
- value: The name of the entry
- qualifier: A qualifier
- scheme: The naming scheme

In addition each element has a value which gives the meaning of the element.
Programs Used:

Ingest Programs

GetIngestFiles:

Reads each offices directory in the drop off area looking for new files. When they are found the program moves the file to the quarantine area and determines if it is a known file, i.e. does the name match a known collection for this office.

I. If so,
   A. It writes an event stating that the file is found
   B. It attempts to validate the file’s format using XML validation
   C. It attempts to verify the checksum for the file.
   D. It writes an event on the conclusion of each of the above steps that states the results of each activity.
   E. Finally the status record is written containing the current status (“In quarantine”), and various pieces of information about the submission that will be of use further down the ingest process.

II. Otherwise
   A. It writes an event stating that the file is found
   B. Writes a status stating that the file is unknown.

III. Removes any files from the drop off area that have a status of “Rejected,” “Added to Fedora” or “Ingest Complete.”

Java Files:

<table>
<thead>
<tr>
<th>Name</th>
<th>Package</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetIngestFiles.java</td>
<td>getingestfiles</td>
<td>Controls the process</td>
</tr>
<tr>
<td>ComputeMD5.java</td>
<td>computemd5</td>
<td>Calculates checksum for passed in file.</td>
</tr>
<tr>
<td>FedoraAccess.java</td>
<td>fedora</td>
<td>Extracts office and collection information from Fedora</td>
</tr>
<tr>
<td>CollectionsProcessor.java</td>
<td>fedora</td>
<td>Static class, used to parse the XML retrieved from fedora</td>
</tr>
<tr>
<td>Coll.java</td>
<td>fedora</td>
<td>Data class. Used to hold title and last update information</td>
</tr>
<tr>
<td>FileMetadataExtractor.java</td>
<td>metadata</td>
<td>Extracts metadata from the XML input file</td>
</tr>
<tr>
<td>Name</td>
<td>Package</td>
<td>Purpose</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FileMetadata.java</td>
<td>metadata</td>
<td>Data class. Used to hold metadata.</td>
</tr>
<tr>
<td>SchemaValidator.java</td>
<td>validateschema</td>
<td></td>
</tr>
</tbody>
</table>

**Libraries:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>fedora-client-0.1.6-with-deps.jar</td>
<td><a href="https://github.com/mediashelf/fedora-client">https://github.com/mediashelf/fedora-client</a></td>
<td>Provides programmatic access to fedora objects and data streams</td>
</tr>
<tr>
<td>jdom.jar</td>
<td><a href="http://www.jdom.org/">http://www.jdom.org/</a></td>
<td>Provides XML parsing support</td>
</tr>
<tr>
<td>log4j-1.2.16.jar</td>
<td><a href="http://logging.apache.org/log4j/">http://logging.apache.org/log4j/</a></td>
<td>Provides logging support</td>
</tr>
<tr>
<td>mysql-connector-java-5.1.16-bin.jar</td>
<td><a href="http://dev.mysql.com/downloads/connector/j/">http://dev.mysql.com/downloads/connector/j/</a></td>
<td>Provides programmatic access to the mysql database</td>
</tr>
<tr>
<td>EventStatus.jar</td>
<td>EventStatus Project</td>
<td>Manages the update and access to Spartan Archive events and statuses</td>
</tr>
<tr>
<td>Mail.jar</td>
<td>Mail Project</td>
<td>Interface to mailx to send email notifications to archivists</td>
</tr>
</tbody>
</table>

**External Resources:**

Fedora (msu-uahc:SpartanArchive:source)  
mailx program on the servers
**Ingest2**

I. For each submission that is marked as “Approved for Ingest:"
   A. Extracts additional metadata from the file
   B. Using this and the defaults from Fedora creates DublinCore and Premis metadata.
   C. Creates a new Fedora data object for this submission, and updates the parent collection object in Fedora.
   D. Moves submission from quarantine to the records area
   E. If all is successful updates the status to “Added to Fedora”

II. Any submissions marked for Rejection, or that have been moved to the records area and added to the database are deleted.

III. Creates new events for each activity, noting the success or failure of the activity

Java Files:

<table>
<thead>
<tr>
<th>Name</th>
<th>Package</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingest2.java</td>
<td>ingest2</td>
<td>Controls the process</td>
</tr>
<tr>
<td>FileMetadata.java</td>
<td>ingest2</td>
<td>Data class. Used to hold metadata.</td>
</tr>
<tr>
<td>DCBuilder.java</td>
<td>metadata.dc</td>
<td>Converts the defaults from Fedora and the submission file into a DublinCore XML stream</td>
</tr>
<tr>
<td>DCElement.java</td>
<td>metadata.dc</td>
<td>Stores DublinCore data element and converts it to XML</td>
</tr>
<tr>
<td>PremisBuilder.java</td>
<td>metadata.premis</td>
<td>Converts a PremisData object into an XML data stream.</td>
</tr>
<tr>
<td>PremisData.java</td>
<td>metadata.premis</td>
<td>Data class. Contains data needed to create Premis XML data stream.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“Significant properties” occur multiply so are in a contained data class</td>
</tr>
<tr>
<td>Property.java</td>
<td>metadata.premis</td>
<td>Data class. Used for “significant properties”</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td><strong>Source</strong></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>fedora-client-0.1.6-with-dependencies.jar</td>
<td><a href="https://github.com/mediashelf/fedora-client">https://github.com/mediashelf/fedora-client</a></td>
<td>Provides programmatic access to fedora objects and data streams</td>
</tr>
<tr>
<td>EventStatus.jar</td>
<td>EventStatus Project</td>
<td>Manages the update and access to Spartan Archive events and statuses</td>
</tr>
<tr>
<td>Mail.jar</td>
<td>Mail Project</td>
<td>Interface to mailx to send email notifications to archivists</td>
</tr>
<tr>
<td>FedoraAccess.jar</td>
<td>FedoraAccess Project</td>
<td>Interface to Fedora</td>
</tr>
<tr>
<td>FedoraDefaults.jar</td>
<td>FedoraDefaults Project</td>
<td>Interface to defaults stored in Fedora</td>
</tr>
</tbody>
</table>
Database Load Programs

**AcadXMLToDatabase**

Converts the XML submission format into 4 database tables: section (containing the section title and HTML content), sectiontosection (containing links between the levels of the Academic Programs document), keywords (keywords linked to the section) and searchContent (section content in plain text form for searching).

The program will only run when there is an academicPrograms.xml submission that has the status “Added to Fedora.”

At the successful conclusion the program updates the status to “Added to Database. Ingest Complete”

An event record is added for either success or failure and the archivists are notified by email for both situations.

**Java Files:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Package</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main.java</td>
<td>acadxmltodatabase</td>
<td>Converts XML submission source and calls Database to load the tables.</td>
</tr>
<tr>
<td>Database.java</td>
<td>acadxmltodatabase</td>
<td>Manages database access</td>
</tr>
</tbody>
</table>

**Libraries:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventStatus.jar</td>
<td>EventStatus Project</td>
<td>Manages the update and access to Spartan Archive events and statuses</td>
</tr>
<tr>
<td>Mail.jar</td>
<td>Mail Project</td>
<td>Interface to mailx to send email notifications to archivists</td>
</tr>
</tbody>
</table>
External Resources:

fedora-db.ats.msu.edu database server

PCXMLToDatabase

Converts the Course Descriptions XML submission format into database tables: queryTable (a flat representation of the Course Description items) and yearsOffered (containing the year and a link to the course when it is offered.)

This submission is complicated in that only changes are sent, not the entire new version of the course descriptions. As such older versions are “archived” and each has a starting date and a potential end date. This program works through the complication to create a single row for each course for each year it is offered.

The program will only run when there is an courseDescriptions.xml submission that has the status “Added to Fedora.”

At the successful conclusion the program updates the status to “Added to Database. Ingest Complete”

An event record is added for either success or failure and the archivists are notified by email for both situations.

Some of the SQL queries are contained in a properties file: queryStrings.properties

Java Files:

<table>
<thead>
<tr>
<th>Name</th>
<th>Package</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCXMLToDatabase.java</td>
<td>pcxmldatabase</td>
<td>Converts XML submission source and calls Database to load the tables.</td>
</tr>
<tr>
<td>Database.java</td>
<td>pcxmldatabase</td>
<td>Manages database access</td>
</tr>
<tr>
<td>Course.java</td>
<td>pcxmldatabase</td>
<td>Data class. Contains course information from the XML file.</td>
</tr>
</tbody>
</table>

Libraries:

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>jdom.jar</td>
<td><a href="http://www.jdom.org/">http://www.jdom.org/</a></td>
<td>Provides XML parsing support</td>
</tr>
</tbody>
</table>
mysql-connector-java-5.1.16-bin.jar | http://dev.mysql.com/downloads/connector/j/ | Provides programmatic access to the mysql database
---|---|---
EventStatus.jar | EventStatus Project | Manages the update and access to Spartan Archive events and statuses
Mail.jar | Mail Project | Interface to mailx to send email notifications to archivists

External Resources:

fedora-db.ats.msu.edu database server

SDXMLToDatabase

Converts the Student Directories XML submission format into a database table: queryTable (a flat representation of the Student Directory items)

The program will only run when there is a studentDirectory.xml submission that has the status “Added to Fedora."

At the successful conclusion the program updates the status to “Added to Database. Ingest Complete”

An event record is added for either success or failure and the archivists are notified by email for both situations.

Java Files:

<table>
<thead>
<tr>
<th>Name</th>
<th>Package</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDXMLToDatabase.java</td>
<td>sdxmltodatabase</td>
<td>Converts XML submission source and calls Database to load the tables.</td>
</tr>
<tr>
<td>Database.java</td>
<td>sdxmltodatabase</td>
<td>Manages database access</td>
</tr>
<tr>
<td>StudentRecord.java</td>
<td>sdxmltodatabase</td>
<td>Data class. Contains student directory information from the XML file.</td>
</tr>
</tbody>
</table>
Libraries:

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>jdom.jar</td>
<td><a href="http://www.jdom.org/">http://www.jdom.org/</a></td>
<td>Provides XML parsing support</td>
</tr>
<tr>
<td>mysql-connector-java-5.1.16-bin.jar</td>
<td><a href="http://dev.mysql.com/downloads/connector/j/">http://dev.mysql.com/downloads/connector/j/</a></td>
<td>Provides programmatic access to the mysql database</td>
</tr>
<tr>
<td>EventStatus.jar</td>
<td>EventStatus Project</td>
<td>Manages the update and access to Spartan Archive events and statuses</td>
</tr>
<tr>
<td>Mail.jar</td>
<td>Mail Project</td>
<td>Interface to mailix to send email notifications to archivists</td>
</tr>
</tbody>
</table>

External Resources:

fedora-db.ats.msu.edu database server

SOCXMLToDatabase

Converts the Course Schedules XML submission format into a database table: queryTable (a flat representation of the Student Directory items)

The program will only run when there is a courseSchedule.xml submission that has the status “Added to Fedora.”

At the successful conclusion the program updates the status to “Added to Database. Ingest Complete”

An event record is added for either success or failure and the archivists are notified by email for both situations.

Some of the SQL queries are contained in a properties file: queryStrings.properties
Java Files:

<table>
<thead>
<tr>
<th>Name</th>
<th>Package</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCXMLToDatabase.java</td>
<td>loadDatabase</td>
<td>Converts XML submission source and calls Database to load the tables.</td>
</tr>
<tr>
<td>Database.java</td>
<td>loadDatabase</td>
<td>Manages database access</td>
</tr>
<tr>
<td>Section.java</td>
<td>loadDatabase</td>
<td>Data class. Contains course information from the XML file.</td>
</tr>
</tbody>
</table>

Libraries:

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>jdom.jar</td>
<td><a href="http://www.jdom.org/">http://www.jdom.org/</a></td>
<td>Provides XML parsing support</td>
</tr>
<tr>
<td>mysql-connector-java-5.1.16-bin.jar</td>
<td><a href="http://dev.mysql.com/downloads/connector/j/">http://dev.mysql.com/downloads/connector/j/</a></td>
<td>Provides programmatic access to the mysql database</td>
</tr>
<tr>
<td>EventStatus.jar</td>
<td>EventStatus Project</td>
<td>Manages the update and access to Spartan Archive events and statuses</td>
</tr>
<tr>
<td>Mail.jar</td>
<td>Mail Project</td>
<td>Interface to mailx to send email notifications to archivists</td>
</tr>
</tbody>
</table>

External Resources:

fedora-db.ats.msu.edu database server
Shared Local Libraries

**EventStatus**

Supports access and updates to Spartan Archives events and statuses.

Java Files:

<table>
<thead>
<tr>
<th>Name</th>
<th>Package</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventManager.java</td>
<td>eventsStatus</td>
<td>Adds new events to the database</td>
</tr>
<tr>
<td>StatusManager.java</td>
<td>eventsStatus</td>
<td>Manages access to statuses, and updates statuses</td>
</tr>
<tr>
<td>Submission.java</td>
<td>eventsStatus</td>
<td>Data class. Contains status record information. Client programs use this to accumulate data to update the status.</td>
</tr>
</tbody>
</table>

**FedoraAccess**

Provides access and update to Fedora data objects and streams. It extracts collection data from msu-uahc:SpartanArchive. It adds and update collection and instance specific data to Fedora.

Java Files:

<table>
<thead>
<tr>
<th>Name</th>
<th>Package</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>FedoraAccess.java</td>
<td>fedora</td>
<td>Manages access and update to Fedora. Implements defaults.Access from FedoraDefaults</td>
</tr>
<tr>
<td>Collection.java</td>
<td>fedora</td>
<td>Data class. Manages collection data</td>
</tr>
<tr>
<td>Instance.java</td>
<td>fedora</td>
<td>Data class. Manages data for collection instances</td>
</tr>
</tbody>
</table>
### Libraries

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>fedora-client-0.1.6-with-dependencies.jar</td>
<td><a href="https://github.com/mediashelf/fedora-client">https://github.com/mediashelf/fedora-client</a></td>
<td>Provides programmatic access to fedora objects and data streams</td>
</tr>
<tr>
<td>jdom.jar</td>
<td><a href="http://www.jdom.org/">http://www.jdom.org/</a></td>
<td>Provides XML parsing support</td>
</tr>
<tr>
<td>log4j-1.2.16.jar</td>
<td><a href="http://logging.apache.org/log4j/">http://logging.apache.org/log4j/</a></td>
<td>Provides logging support</td>
</tr>
<tr>
<td>FedoraDefaults</td>
<td>FedoraDefaults Project</td>
<td>Provides the Access Interface</td>
</tr>
</tbody>
</table>

### External Resources:

Fedora (msu-uahc:SpartanArchive:source)

FedoraDefaults

### Java Files:

<table>
<thead>
<tr>
<th>Name</th>
<th>Package</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>FedoraDefaults.java</td>
<td>defaults</td>
<td></td>
</tr>
<tr>
<td>Access.java</td>
<td>defaults</td>
<td>Interface that clients must provide to access Fedora.</td>
</tr>
<tr>
<td>DefaultsProcessor.java</td>
<td>defaults</td>
<td>Static class that converts Fedora XML data streams to and from Java class format</td>
</tr>
<tr>
<td>DefaultTypes.java</td>
<td>defaults</td>
<td>The possible default types. Not used in the ingest process.</td>
</tr>
<tr>
<td>Parameter.java</td>
<td>defaults</td>
<td>Data class containing all parameters associated with a given type. Thus this has one or more Parameter.</td>
</tr>
<tr>
<td>Parameters.java</td>
<td>defaults</td>
<td>Data class containing a single parameters data</td>
</tr>
</tbody>
</table>

### Libraries

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>fedora-client-0.1.6-with-dependencies.jar</td>
<td><a href="https://github.com/mediashelf/fedora-client">https://github.com/mediashelf/fedora-client</a></td>
<td>Provides programmatic access to fedora objects and data streams</td>
</tr>
<tr>
<td>dependencies.jar</td>
<td>client</td>
<td>access to fedora objects and data streams</td>
</tr>
<tr>
<td>------------------</td>
<td>--------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>jdom.jar</td>
<td><a href="http://www.jdom.org/">http://www.jdom.org/</a></td>
<td>Provides XML parsing support</td>
</tr>
</tbody>
</table>

External Resources:

Fedora (msu-uahc:SpartanArchive:source)
Data Exported from the Spartan Archive Servlets

In order to add or modify functionality in the Spartan Archive it is necessary to understand what data is provided by the current Spartan Archive Java backend system. Because the current user interface is built using defined data sources, it may be modified without changing the Java system running on the server. Similarly, additional or expanded data sources can be developed on the backend without affecting current web pages.

There are currently two Servlets in use in the Spartan Archive researchers’ interface. These are:

APData used to support the display of Academic Programs information to researchers. This provides its data feeds in the form of XML data streams.

GeneralQuery supports all other collections of records. It provides data as Servlet Attributes to JSP based pages.

Details of each data set are provided below (Note all urls are relative to /GeneralQuery/:)

APData (Academic Programs):

XML data streams:

<table>
<thead>
<tr>
<th>Data stream</th>
<th>Called by</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>/APData?type=titles</td>
</tr>
<tr>
<td>HTML Content of section record</td>
<td>/APData?type=content</td>
</tr>
<tr>
<td></td>
<td>/APData?type=keywordSearch</td>
</tr>
</tbody>
</table>

GeneralQuery:

Attributes:
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Contains</th>
<th>Called by</th>
</tr>
</thead>
<tbody>
<tr>
<td>lookups</td>
<td>List of Lookup class. Each instance of Lookup contains a title identifying the field for which the values belong, and a List of Strings. These Strings may refer to a single value, or may have two values (a title and definition) separated by a colon (&quot;:&quot;). Available fields are: title: which returns the field name values: which returns a List of the possible values of the field</td>
<td>/GeneralQuery?type=queryBuilder&amp;name=CollectionName&amp;key=period CollectionName is the name of the collection, e.g. StudentDirectory Period is either all (for all periods) or the short name. For semesters this is FSYY, SSYY, or USYY (YY is last two digits of the year). For years it is the 4 digit year</td>
</tr>
<tr>
<td>datadictionary</td>
<td>A DataDictionary object containing all of the fields and definitions for the collection. Available fields are: period: either semester or year usageNotes: any applicable help text entries: List of Entry objects. Entry objects have 2 available fields: field and definition size: Number of entries titles: Array of strings containing the fields</td>
<td>/GeneralQuery?type=queryBuilder&amp;name=CollectionName&amp;key=period</td>
</tr>
<tr>
<td>titles</td>
<td>Array of String containing the field names</td>
<td></td>
</tr>
<tr>
<td>downloadPath</td>
<td>Relative url of the location where the downloaded results are stored</td>
<td></td>
</tr>
<tr>
<td>indexField</td>
<td>The period (year or semester)</td>
<td>/GeneralQuery?type=queryBuilder&amp;name=CollectionName&amp;key=period</td>
</tr>
<tr>
<td>Attribute</td>
<td>Contains</td>
<td>Called by</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>shortName</td>
<td>String with the name of the collection with all spaces removed</td>
<td>/GeneralQuery?type=queryBuilder&amp;name=CollectionName&amp;key=period And /GeneralQuery?type=showResults&amp;name=CollectionName&amp;query=queryString&amp;key=period</td>
</tr>
<tr>
<td>queryOutput</td>
<td>List of Array of Strings. Each array contains one row of the returned results. Each string is the value of field in that row.</td>
<td>/GeneralQuery?type=showResults&amp;name=CollectionName&amp;query=queryString&amp;key=period</td>
</tr>
<tr>
<td>recordCount</td>
<td>Total number of rows returned</td>
<td>/GeneralQuery?type=showResults&amp;name=CollectionName&amp;query=queryString&amp;key=period</td>
</tr>
<tr>
<td>query</td>
<td>Slightly modified version of the SQL query used to return the results</td>
<td>/GeneralQuery?type=showResults&amp;name=CollectionName&amp;query=queryString&amp;key=period</td>
</tr>
<tr>
<td>type</td>
<td>The name parameter with a space between the two parts (e.g. StudentDirectory is returned as Student Directory)</td>
<td>/GeneralQuery?type=showResults&amp;name=CollectionName&amp;query=queryString&amp;key=period</td>
</tr>
</tbody>
</table>
Adding new offices and records collections to the Spartan Archive

Offices and records collections are physically represented in the Spartan Archives in these places:

- **msu-uahc:SpartanArchives:source.** This Fedora data stream (source is the data stream within the msu-uahc:SpartanArchives object) contains entries for each office in the archive and each collection supplied by that office.

- **msu-uahc:RG-CollectionName.** There is one Fedora object for each collection identified by its Record Group (RG) and name. An example of this is msu-uahc:UA.6.7-AcademicPrograms. This object contains a dublinCore data stream describing the collection and listing all of the instances belonging to the collection.

- **A directory in the dropoff AFS area named with the short name of the office.** (The short name is often just the office name with the spaces removed, but can be different.) This is where the office drops its submission files.

- **A directory in the data vault (/home/archives/data/records/) named after its official designation.** This is identified by a name like RG-OfficeShortName (for example UA.6.7-Registrarsoffice).

- **A directory within the offices data vault identified with the archives official designation for the records collection instances** (for example SERIAL_108_Academic_Programs).

- **A schema file named CollectionShortName.xsd** (for example academicPrograms.xsd). This should be modeled after the existing schemas (found in /home/archives/schemas) using for element names the database names generated as described below.

- **A database in the spartanarchive-db.itservices.** It will have the short name of the collection with a single table called querytable.

- **A javascript file in GeneralQuery that has a translation between the field names as shown on the screen and the field names in the database.**

Some of these will have to be handled manually. Others can be done automatically. The automatic process will add the office and collection information to the msu-uahc:SpartanArchive source data stream, it will create a new Fedora object for the collection (including the data dictionary) and will create the collection database (which will be empty).

This still leaves a number of activities that will have to be done by others. IT Services will be responsible for creating the drop off area and the Office and collection directories in the data vault. The archives developer will have to build and implement a program to load the office’s submission file to the new database. (The new program can be modeled on SDXMLToDatabase without much trouble. Preferably the developer can modify the process so that a single load program can handle all normal records collections.) In addition the archives developer will have to create the javascript file and add it to the queryBuilder.jsp page. (Again the better solution would be to find a way to do this automatically and eliminate the collection specific javascript files.) Finally the developer will have to create an xml schema for the collection.
Automatic process
This begins with the data template.xlsx spreadsheet (located at S:Projects/Spartan Archive/Programs).
When you open the spreadsheet you’ll see the following:

Use this spreadsheet to collect the information required for the office and collection. Each field has a
tooltip that explains its purpose. (One point of clarification. The Submission Name field has a #VALUE
warning. This will be automatically created when the Collection Short Name is entered. Or it can be
entered explicitly.)
The database will be created with the fields in the same order as entered here. The names of the generated database fields will simply be the field names entered with spaces removed. (The submitting office should use the same, i.e. the database field names, for element names in their submission.)

Once this is filled out, and all parties agree to the result, save the spreadsheet as a .csv file. Then run CreateDBFromCSV (found in S:\Projects\SpartanArchives\Programs) as follows:

java –jar CreateDBFromCSV.jar nameOfFile.csv   (where nameOfFile.csv is the name you gave it when you saved the file.)

This will update Fedora with all needed objects and data streams and create the database for eventual loading.
Changing Ingest Preferences

Server, locations, user ids and passwords change from time to time. The problem is that they are used in multiple programs. In order to minimize the hassle involved in changing any of these values they are stored in a Java preference file on each server. They are only accessible to the user archives.

To change any of these value log in to either server (fedora.ats.msu.eu, or archives-storage.ats.msu.edu) as archives. CD to the archives/bin directory and run

```
java –jar SetPreferences.jar entryName::value entryName::value
```

repeat the entry::value pair as often as needed to set each preference.

As an example to change the password for the fedora user run

```
java –jar SetPreferences.jar fedoraP::newpassword
```

To get a current list of the preferences run

```
java –jar SetPreferences.jar report
```

The table below shows all of the current preference and their meanings

<table>
<thead>
<tr>
<th>Entry</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>fedoraP</td>
<td>Password for the fedora user account</td>
</tr>
<tr>
<td>fedoraURL</td>
<td>URL for fedora</td>
</tr>
<tr>
<td>fedoraUser</td>
<td>User account for fedora</td>
</tr>
<tr>
<td>quarantine</td>
<td>The location of the quarantine area</td>
</tr>
<tr>
<td>repository</td>
<td>The server holding the data vault</td>
</tr>
<tr>
<td>sadb</td>
<td>The address of the database where the events and statuses are stored</td>
</tr>
<tr>
<td>sadbP</td>
<td>The password for the database user account</td>
</tr>
<tr>
<td>sadbUser</td>
<td>The database user account</td>
</tr>
<tr>
<td>schemaLocation</td>
<td>Location of the XML schemas</td>
</tr>
</tbody>
</table>
Potential Problems

Ingest:

We’ve run into x types of problems in the ingest:

Submission does not validate.

This has been caused by issues such as:

- **Odd (non-ascii) characters in the submission.** We’ve asked the Registrars Office to use <?xml version="1.0" encoding="ISO-8859-1"?>. This tells the validating program that these characters are legitimate. If this repeats, check to see if the first line in the submission contains this string. If not replace the xml entry (<?xml version="1.0"?>) with that value.

- **Invalid string in the root element.** In the Student Directory file, the attribute: xmlns:xsi=http://www.w3.org/2001/XMLSchema causes the file to fail validation. The Registrars Office has been asked to remove it, but if it is still there, you’ll have to remove the phrase from the root element manually. This can be done with a text processor.

- **Mandatory element missing or element is not cvc-pattern-valid or facet-valid.** The xml schemas have been pared down to what I believe is the minimum necessary. However it is possible that new combinations will appear. If that occurs you’ll need to modify the schemas. They are located in the archives-storage server in archives/schema. Each one has the same name as the submission file with an .xsd suffix rather than an .xml suffix.

If the message is mandatory element missing, find the line in the schema that addresses the element and add the attribute minOccurs = "0".

If a non compatible message is delivered then change the appropriate line in the schema from a custom type to a generic one such as type="xs:string". There are a number of examples of this in the studentDirectory.xsd schema. If the non-compatible item is the year or semester attribute in the root, then the file is bad and should be returned for correction.

If any of these occur you will then have to recalculate the checksum. You can do this with the fciv program from Microsoft (available from http://support.microsoft.com/kb/841290). You do this with the following line (the example uses courseDescriptions, but will work with any other submission substituted):

```
fciv courseDescriptions.xml > courseDescriptions.xml.sum
```

Replace the original checksum file with your new one before attempting to re-ingest the file.

Database Load Issues

There are two likely problems with the database load:

- Data is too long for the field. This will occur when one of the text field that comes in is longer than the space allocated for it on the database. I’ve tried to be more than generous with the space, but the departments have surprised me before. This problem will be obvious in the
administrative interface. When you check Manage Submissions the status will be Added to Fedora, not Added to Database, IngestComplete. When you click on the message the Event listing will show the error.

To resolve this you need to open Navicat, select fedora-db then the database with the problem. Right click on querytable (unless this is academic programs in that case use the table from the error message) and chose design table. Then increase the size of the field appropriately. You’ll need to delete any records from the partial update and re-run the load. That will be described below.

- Alphabetic data in a numeric field. Again this will be seen in the Manage Submission page. This can happen if the Registrars Office changes the format of a field. To correct this open Navicat, select fedora-db, then the database in question. Right click on the table (see above) and chose design table. Find the field and change its type to char with size 12. Then delete any partial update and rerun the database load. (see below)

To delete a partial update contact Felicia and ask to have the database restored to prior to the update. If that fails, open Navicat, chose fedora-db and the database in question. For non Academic Program databases, double click on querytable. This will show you a listing of all of the records in the database, 1000 records at a time. At the bottom right is a button that looks like ->]. Click on that button. This will take you to the end of the table. Verify that the Semester field matches that of the partial update.

At this point back up the database by going back to the main Navicat screen, and with the database open click on Backup at the top of the screen and then New Backup (in line below). This will bring up a dialog box. Click on Start. Close the box when it is complete.

To delete the partial update, go back to the main Navicat screen and with the database open click on the Query button at the top and then New Query in the line below. In the page that appears type

Delete from querytable where semester = “[semester code as shown in the listing above]”

For example to delete the partially updated records for Spring Semester 2013 you would type

Delete from querytable where semester =”SS13”

To perform the delete click on the Run button in the icon list below the menu.

If it is successful you will see a message like:

[SQL] delete from querytable
Affected rows: 24
Time: 0.031ms

To rerun the load log on to fedora.ats.msu.edu as archives. Open the bin directory then type
./runPrograms.sh

and press Enter. This will rerun any tasks that are incomplete.

Query System Hangs

If this should happen you’ll need to refresh the web application. To do this log into fedora.ats.msu.edu as a user with sudo privileges. Open the /usr/local/opt/apache-tomcat-7.0.22/webapps directory. Then type

    sudo touch GeneralQuery.war

This will cause the application to be reloaded from its compressed form. This will normally clear any problems. If any problems persist you’ll need to work with Felicia to resolve them. For what it’s worth this has not yet been necessary.
Ingesting a Records Collection

When an office drops off a new submission in its dropoff area, the Spartan Archive administrator will get an email stating that the file is here and has been moved to quarantine. When that occurs the administrator should open the Manage Submissions page (http://fedora.ats.msu.edu:8080/SAControl/ControlSource?type=submissions).

The new files will have a status of In Quarantine. To approve the ingest click on the line with the new file. This will enable the Actions for Submission box on the right.
Before continuing open Archivist Toolkit and get the next Accession Number available. (To do this double click on the word Accessions on the left. When the accessions appear type A.YYYY where YYYY is the current year in the filter field at the top right. This will bring up a list of all of the accessions or the year. The new accession will be A.YYYY.XXXX where XXXX is the next available sequential number. In the example above the new accession number would be A.2012.00117) **Do not create a new accession at this time.**
In that Actions for Submission box click on Accept for Admission. Then enter the new accession number in the field provided. Click on Accept.

The status for the file should change to Approved for Ingest. In addition below the Accept button there will be a link to download an import file for Archivist Toolkit. Left click (or command click on a Mac) the link to download the atimport.xml file to your system.
In Archivist Toolkit, click on Import/Import Accessions (XML) menu item (at the top left).

Navigate to the directory in which you saved the atimport.xml file. Select the file and then click the Import button. A dialog box will appear. If there are no errors, you are done.

If there is an error message saying that this is a duplicate accession someone may have used the number before you. In that case you should get a new number and repeat the approval process. So long as it is all done on the same day, there should be no problems.
If there is any other error—investigate and correct any problems found. As a last resort you can print the atimport.xml file and manually create the accession.

If you do not want to approve the file you can select either pend or reject the file. Pending only serves to remind you to investigate.

If the file says that it is In FTP, there was a problem with the file. Click on the file and study the events listed below it. They should show any problems. Use the procedure in the Potential Problems document to resolve any issues.
Manually updating data in Fedora

Fedora can be accessed directly through a web interface (http://spartanarchive.msu.edu:8080/fedora/admin). A user id and password are required to see specifics or make changes. These will be requested at the time they are required.

Click on the Search button on the top left and enter msu-uahc:* in the search field. Click on the search button to see all objects in the Spartan Archive. Choose the object desired. Metadata is stored in objects as follows:

Dublin Core data streams for collections is saved in objects named like: msu-uahc:RG-Collection Name, for example:

- msu-uahc:UA.6.7-AcademicPrograms
- msu-uahc:UA.6.7-CourseDescriptions
- msu-uahc:UA.6.7-CourseSchedules
- msu-uahc:UA.6.7-StudentDirectories

Dublin Core and Premis data streams for collection instances as saved in objects named like: msu-uahc:RG-Accession Number-xxx, for example: msu-uahc:UA.6.7-A2012.0016.001.

Each metadata data stream is stored as xml. To edit the xml for an object choose the object from the list and double-click it. Choose the desired data stream from the list on the bottom and double-click it. Finally click on the Edit button. This will bring up a window with the text of the data stream.

Make any changes you want and then click on save changes to finish. (If you don’t see the button, expand the size of the overall window.) If you have made any changes that make it an invalid xml file you will be given an error message. Correct the error and resave.

Close the browser window to exit.
Updating Spartan Archives Parameters

Background

Parameters in the Spartan Archive System serve two purposes. First they provide standard values for premis and Dublin Core metadata. So by modifying the values of the parameters you can control the contents of future premis and Dublin Core data streams. Second they are used in the control and processing of the Spartan Archive ingest and query systems.

The table below lists all of the parameter types available along with sample values and a guide to where the parameters are used. The possible parameters are controlled by the defaultTypes data stream in the msu-uahc:SpartanArchive Fedora object. You can add new types by editing that data stream. You should not remove or change any of the existing parameter types as that may cause the Ingest or query system to break. (Changes will need to be coordinated with changes to the codes.)

<table>
<thead>
<tr>
<th>Entry Name</th>
<th>Value</th>
<th>Scheme</th>
<th>Qualifer</th>
<th>Used In</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submission Name</td>
<td>courseDescriptions.xml</td>
<td></td>
<td></td>
<td>Ingest process, Premis</td>
</tr>
<tr>
<td>Collection Name</td>
<td>Description of Courses</td>
<td></td>
<td>Dublin Core</td>
<td></td>
</tr>
<tr>
<td>Collection Short Name</td>
<td>CourseDescriptions</td>
<td></td>
<td>Ingest, Query System</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Annual listing of course descriptions</td>
<td></td>
<td>Dublin Core</td>
<td></td>
</tr>
<tr>
<td>Record Group</td>
<td>UA.6.7</td>
<td></td>
<td></td>
<td>Ingest, Premis</td>
</tr>
<tr>
<td>Record Group Description</td>
<td>Office of the Registrar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record Group Short Name</td>
<td>RegistrarsOffice</td>
<td></td>
<td>Ingest, Premis</td>
<td></td>
</tr>
<tr>
<td>Publisher</td>
<td>Michigan State University</td>
<td>NACO</td>
<td>Dublin Core</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>eng</td>
<td>Iso63903</td>
<td></td>
<td>Dublin Core</td>
</tr>
<tr>
<td>Type</td>
<td>dataset</td>
<td>DCMITYPE</td>
<td></td>
<td>Dublin Core, Premis</td>
</tr>
<tr>
<td>Coverage</td>
<td>East Lansing (Mich.) 48823</td>
<td>LCNAF</td>
<td>Spatial</td>
<td>Dublin Core</td>
</tr>
<tr>
<td>Creator</td>
<td>Michigan State University. Office of the Registrar</td>
<td>NACO</td>
<td></td>
<td>Dublin Core</td>
</tr>
<tr>
<td>Subject</td>
<td>course descriptions</td>
<td></td>
<td></td>
<td>Dublin Core</td>
</tr>
<tr>
<td>Contributor</td>
<td>Kristin Schuette</td>
<td>NACO</td>
<td></td>
<td>Dublin Core</td>
</tr>
</tbody>
</table>
**Format** | text/xml | **IMT** | **Dublin Core Premis**
---|---|---|---
**Relation** | msu-uahc:UA.6.7-CourseDescriptions | isPartOf | Dublin Core
**Rights** | Use of this public domain resource is unrestricted | | Dublin Core
**Significant Properties** | | | Premis

Most of the metadata terms are obvious, but some of these control processing so more explanation is needed.

Submission Name is used in the initial Ingest process to recognize known files. In the example the Registrars Office must name its course descriptions submissions courseDescriptions.xml. No other name will be recognized.

Collection Name is used in creating the import file for Archivist Toolkit.

Collection Short Name is used to identify the Fedora collection level object (in the sample it would msu-uahc:UA.6.7-CourseDescriptions, and the Data Definitions/Parameters object for the collection msu-uahc:DDCourseDescriptions).

Record Group is used to build the Fedora Permanent ID (pid) and to help determine where the file should be stored on disk.

Record Group Short Name is used to help determine where the file is stored on disk. It cannot contain spaces.

Relation is used to connect instances of a collection to the collection as a whole. The Dublin Core data stream contains a `<relation qualifier="isPartOf">` entry pointing at its collection. Each collection has a set of `<relation qualifier="hasPart">` entries pointing at its instances.

**Maintaining Parameters**


To display the parameters for editing choose the office and the collection as shown below:
This lists all of the parameters. To change a parameter just over write what is there. To add a new parameter value (for parameters like subject that support multiple values) click on the + button in the row. This will open a new row with the same entry name. To remove a parameter clear the Value, Scheme and Qualifier fields.

When you are done click the Update Parameters button. (Note: because of caching if you return to this page in the next few hours you may see the older values. They have actually been changed.)
Updating Spartan Archives Programs

When a new program has been developed or an older program has been updated you can put it into production as follows:

**Batch programs.**

Batch programs are loaded as .jar files. You can generate these, along with any .jar files they depend on, with any IDE or via the command line. When they are ready to be deployed ftp them to

For GetIngestFiles.jar: archives-storage.ats.msu.edu, user archives, directory bin

For all others: fedora.ats.msu.edu, user archives, directory bin

Support .jar files need to be put in the bin/lib directory.

**Web Applications**

Web applications are deployed as .war files. These are a little more difficult as none of our users has write access to the web apps directory. To deploy these ftp them to a user with sudo privileges on fedora.ats.msu.edu. Then sign on to fedora.ats.msu.edu as that user and open the web apps directory at (/usr/local/opt/apache-tomcat-7.0.22/webapps). Then type

    sudo cp ~/WebAppName.war ./ then Enter

For example

    sudo cp ~/GeneralQuery.war ./

This will copy the web application to the directory. Tomcat will automatically expand this into a directory of the same name. You can tell when this has happened if you type

    ls -l then Enter

When the times on the .war file and the directory match the web application has been deployed.

If this does not happen open the logs directory (/usr/local/opt/apache-tomcat-7.0.22/logs) and type

    tail –n100 catalina.[today’s date in year.month.day form].log

For example

    tail –n100 catalina.2012.07.05.log

This will give you a list of the most recent entries made by Tomcat. Work with Felicia to interpret the results and correct the problem.
Updating the Data Dictionary for a Collection

The data dictionary for a collection serves three roles. First it is used to provide popup messages on the search screen. These serve to let researchers know what is in the field that they may want to search on. Second it provides a place to put general messages to researchers about a collection. Finally, and most critically to the functionality of the Spartan Archive, it controls the generic search process.

In order to keep things flexible and to minimize the effort it will take to add new collections to the Archive, the search results do not differ between collections. The results are put out in the same order as they are stored. And the titles are pulled from the data definitions. As a result if the definitions order differs from the storage order the results will not make sense. The process for adding new collections will enforce this synchronization, but it is important not to change the order after the fact.

Updating the data definitions are done on the data definitions page. To get to this open the Spartan Archives Configuration Page (http://fedora.ats.msu.edu:8080/SACtrl/ControlSource?type=offices).

Initially this shows a list of the offices providing collections. Click on the office you are interested in and then chose the collection you want to update.
This will show the parameters used with the collection. In addition at the bottom of the page the Data Dictionary button will be enabled. Click on this button to bring up the data dictionary page.
Update any of these fields as required. Then click the Update Definitions button to complete the process.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interdepartment With</td>
<td>If offered through more than one department, will contain department codes of other dep</td>
</tr>
<tr>
<td>Administered By</td>
<td>If administered through more than one department, will contain the department code of</td>
</tr>
<tr>
<td>Course Title</td>
<td>Name of the course, such as Calculus</td>
</tr>
<tr>
<td>Course Description</td>
<td>Description of course contents</td>
</tr>
<tr>
<td>Semesters Offered</td>
<td>Rule for semesters for which this course will be offered, such as ANP 854 offered Fall o</td>
</tr>
<tr>
<td>Credits</td>
<td>Credit value of the course Lab plus lecture credits, do not necessarily add up to the act,</td>
</tr>
<tr>
<td>Lab Credit</td>
<td>Number of class hours per week in laboratory</td>
</tr>
<tr>
<td>Lecture Credit</td>
<td>Number of class hours per week in lecture/recitation/discussion</td>
</tr>
<tr>
<td>Variable Minimum Credit</td>
<td>For courses with variable credit value, minimum amount of credit that can be awarded</td>
</tr>
<tr>
<td>Variable Maximum Credit</td>
<td>For courses with variable credit value, maximum amount of credit that can be awarded</td>
</tr>
<tr>
<td>Variable Increment</td>
<td>For courses with variable credit value, size of the steps between possible credit values</td>
</tr>
<tr>
<td>Reenrollment Credit</td>
<td>Maximum amount of credit that can be earned from this course if taken multiple times; p</td>
</tr>
<tr>
<td>Grading Option</td>
<td>Identifies course with pass/no pass grading scheme</td>
</tr>
<tr>
<td>Restrictions</td>
<td>Limitations on student access to the course, such availability only to seniors or to stud</td>
</tr>
<tr>
<td>Recommended Background</td>
<td>Prior academic work, experience, or other qualifications recommended but not required</td>
</tr>
<tr>
<td>Prerequisite</td>
<td>Courses that must be completed prior or concurrently with the course</td>
</tr>
<tr>
<td>Semester Alias</td>
<td>Another course identified as the equivalent of this course</td>
</tr>
<tr>
<td>Associated Course</td>
<td>Course associated with this course</td>
</tr>
<tr>
<td>Corequisite Course</td>
<td>Course that must be completed concurrently with this course</td>
</tr>
<tr>
<td>College Code</td>
<td>Numerical identifier for the college offering the course, such as 16 for the College of Err</td>
</tr>
<tr>
<td>Department Code</td>
<td>Numerical identifier for the department offering the course such as 24 for the Departine</td>
</tr>
<tr>
<td>College Code Description</td>
<td>Full name of the college</td>
</tr>
<tr>
<td>Department Code Description</td>
<td>Full name of the department</td>
</tr>
<tr>
<td>Subject Code Description</td>
<td>Full name of the subject area</td>
</tr>
</tbody>
</table>
Possible Enhancements to the Spartan Archives

Process changes

Show unknown files in the submissions list and allow an administrator to change it to a known name.

Currently if an office misnames a submission there will be an event generated and an email sent to the Archives administrator. The only recourse here is to contact the office and have it renamed and resent. It would be more effective to allow the administrator to rename the file directly once it has been identified. This would require changes to the ingest process and the Administrative Console.

Create a generic database load

Currently each database is loaded with its own program. Originally this was due to the fact that I was breaking the data down to third normal form. This is useful, but not so much for our purposes. As a result I’ve greatly simplified them so that (with the exception of Academic Programs) they all build one large record for each relevant item. It seems likely that the non-Academic Programs loads could be consolidated into a single load program with some ingenuity. If that was done it would not only reduce the maintenance on the current loads, but could greatly simplify the process of adding new collections.

In the current process the Fedora structure and the database structure can be created automatically. However, a separate database load needs to be written. If we had a generic load that would eliminate that step. This would also make the Spartan Archive structure much more portable to other institutions.

Create a method to automatically ingest Duke Data Accessioner files to Fedora

DDA is being used to ingest digital files to the digital vault. But this is not getting them into Fedora for organizational or presentation purposes. DDA produces XML files with all, or most of what is needed to create a Fedora object. Entering all of this by hand is problematic. It would be possible to create a Java/JDom program to automatically extract the DDA data and create a Fedora object and associated data streams.

Create/find a process to checksum/validate Archivelt files

Archivelt web files take up the great majority of files in the digital vault. However they are not checksummed and cannot be validated. A way needs to be determined to ensure the integrity of these files.

Add virus checking to ingest

Because of the uncertainty of where ingest was going to take place and the uncertainty of what tool to use for the purpose, virus checking was not implemented in the ingest process. Because all of the files currently handled are pure text files, this was not a major concern. However it is still a useful function that should be added. To do so a virus checker that can be run on CentOS should be obtained. There is
already a function in GetIngestFiles.java that can be used to call the tool. It is currently disabled but once a tool is selected it can be added.

Changes to the Query System

**Add definition pop-ups on the query results table.**

Currently definition pop-ups are enabled on the query page. This is helpful, but there is no current way to show definitions on the results page. This should be added.

**Add full text search as a query method**

The current system has limited searching capabilities. There is a keyword search for Academic Programs and a logical query mechanism for the other collections. However being able to do Google style searches on the text would have obvious value. This could most readily be done using Solr and Lucene. It appears that there are others in ITS using these tools so there may be some support there. In addition Fedora has several facilities to integrate this kind of search. You can see how some people are using it by reviewing past postings to the Fedora mailing list.

**Modify the links in the Academic Programs display to point to historic sites**

When you view the Academic Programs information in the Spartan Archives there are a number of web links shown. Currently those links are pointing to the original locations. As time goes on the information in those pages changes and links may disappear. This provides a less than optimum experience for researchers. It would be better if those links pointed to the original pages (rather than the pages currently linked.) This should be possible for the majority of the links that point to MSU web sites. Ed has been working with Archivelt to archive MSU’s web sites. A program can be written to change the links in the current database to point to the pages in Archivelt instead. Doing this would let the researcher know exactly what was in the link at the time that the Academic Programs document was produced.

Technical Changes

**Eliminate redundant code**

There are a number of places where similar functionality has been implemented redundantly. Some of this was inadvertent; some was simply the most expedient way to do something at the time. However, this means that maintaining the system will be that much harder. For example since the system is built around Fedora there are multiple places where Fedora interaction takes place. There was some attempt to centralize this in a single FedoraAccess library, but immediate difficulties arose that caused me to put that off. Thus there is a FedoraAccess library used by PDFPuller, Ingest2, and the new collection process. However there are also FedoraAccess objects in GetIngestFiles, GeneralQuery and SAControl. Time spent rationalizing and centralizing the code will be repaid with easier maintenance down the line.
Similarly Ingest2 and PullPDF both create Dublin Core and Premis data streams. These processes should be consolidated.

**Add the database locations to the Ingest Preferences**

The ingest process uses a set of preferences (located in /home/archives/.java/.userPrefs/prefs.xml) to keep track of the information required to access Fedora, send mail, and a number of other things. However the database load programs have the location of the databases, the user id and password hard coded into the programs. This makes them more difficult to test and makes it harder to make changes when one of these values changes. These three values database location, database user id and database password should be added to the preferences. Each of the load programs already use preferences for some of their processing so there are examples of how to do this. Adding preferences is described in S:\Projects\Spartan Archive\Procedures\Changing Ingest Preferences.docx.

**Add Serial value to parameters**

Currently the Archive’s Serial name (for example Academic Programs is 108) is hard coded into the Ingest2 program. That will make adding new collections that much harder because the code will have be changed, recompiled and deployed to add a new serial. If it is added to the parameters this can be done on-line by an archivist.

**Create a way to test on a test server**

Currently code is developed on Windows and deployed in Linux. This is generally workable, but several cases have occurred where the differences between the two systems has created problems that could not be readily tracked down. When debugging in Window, I use NetBeans to walk through the code as it is being processed. On Linux I can’t do that because NetBeans won’t run without a graphical environment, which means that I am restricted to reviewing logs and print lines. If the test system is created with an X11 interface, NetBeans will run and full debugging can be done there.

**Create new Fedora content models for JPG, MPEG, etc.**

A Fedora content model is basically a way of packaging web services for use in presenting data from Fedora objects in more appropriate formats. For example the content model used for the current collections packages the Saxon XSL Translator to reformat the xml data streams of Spartan Archives collections into more usable web pages. For photographs a content model can support the display of a single digital format in thumbnail, medium resolution or high resolution form. Many other opportunities exist as well that the Archive should research.

**Modify the load programs to be fully generic**

Currently the load programs (with the exception of Academic Programs) are very similar. Each takes the xml file and maps it into a 1 table database. This is particularly useful for the style of queries that the Spartan Archive supports. They are all very similar but there are still 3 of them. Continuing to develop the archive this way requires a new load program for every records based collection. This adds
development costs and hurts it portability. Instead the archives should modify the process to have a single load program that can use the data dictionary developed for the collection and the generated database design.

**Convert batch process to web apps and integrate into a content model**

This was suggested by Aaron Collie and makes a lot of sense. Doing so would greatly enhance the portability of the project. Doing this requires reviewing the existing design. Do we stick with batch processing to discover and ingest files, or have it kicked off when an administrator opens the application? Both seem doable (there is an open source project called quartz which allows scheduling J2EE applications like this one at [www.quartz-scheduler.org](http://www.quartz-scheduler.org)).

**Put a limit on the number of events and submissions shown**

Currently the “Manage Submissions” page uses all of the submissions and associated events that have ever occurred. This could be distracting. Consider limiting the displayed submissions to a time limit (6 months? 2 weeks?) depending on the volume of ingest.