Electronic Records Modules
Electronic Records Committee
Congressional Papers Roundtable
Society of American Archivists

Create AIP with DataAccessioner and Bagger

Adriane Hanson
University of Georgia
ahanson@uga.edu

Date Published: **July 2016**

Module#: ERCM003

Create AIP with DataAccessioner and Bagger

May 17, 2016

Overview and Rationale

For preservation and to prove the files are authentic, we want the preservation masters for our digital files and the metadata about them to be stored together and maintained in a single package that makes it easy to verify that the contents are unchanged over time. This is the concept of an Archival Information Package (AIP) from the OAIS Reference Model.

This workflow allows staff to generate a basic AIP using free tools, the PREMIS metadata standard, and the BagIt specification. It is particularly useful for institutions without a digital asset management system that would help automate this process. By using standards that are common across the profession, it increases the likelihood that the AIP will be able to be ingested into any management system the institution acquires or develops at a later date.

This workflow is also designed with some flexibility so that different departments at the same institution can create compatible AIPs but still allow for variation between content types and department needs. Each department uses the same standards, the same top-level file directory structure, and creates a master metadata file. Each department can then make individual decisions about additional metadata to include and how to structure files within the "objects" folder in the bag.

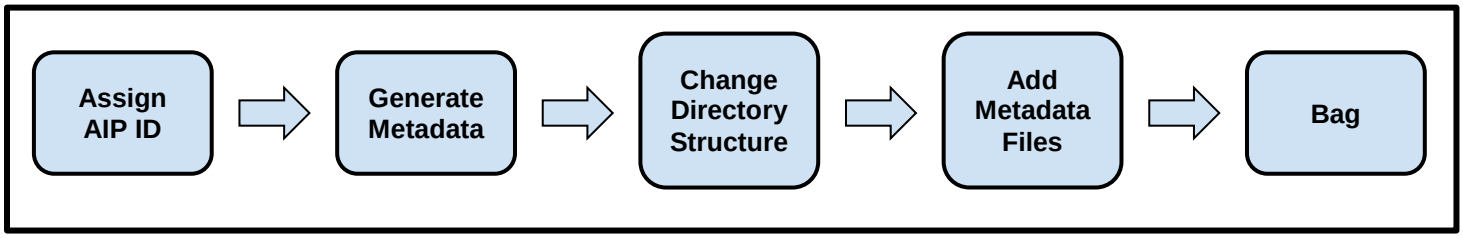
Procedural Assumptions

- All files to be included in an AIP are stored in a folder together
- PREMIS is the standard used for preservation metadata
- The files have already been scanned for viruses
- Appraisal has been completed

Hardware and Software Requirements

- DataAccessioner (free) to generate metadata about the files
- XML Blueprint for XML editor to transform a subset of the DataAccessioner metadata into a master metadata file
- Bagger (free) to package the AIP

Workflow Overview

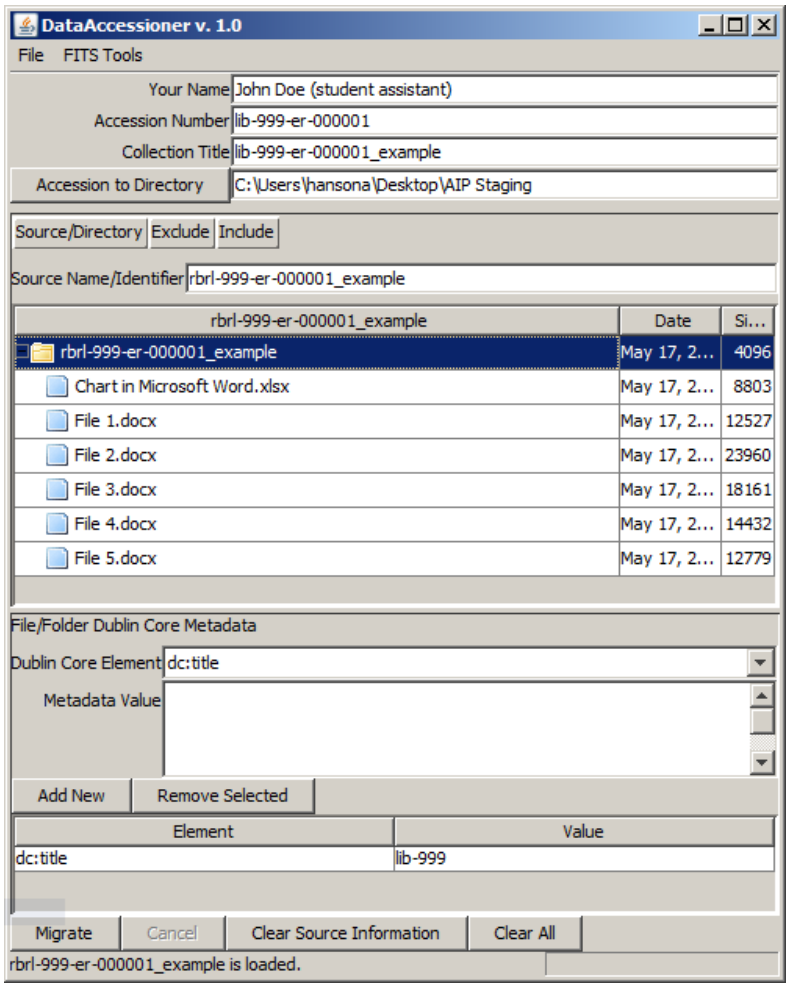


Workflow

1. Assign identifier to the AIP, which includes the collection number, "er" to indicate it is electronic records, and a sequential number. Example: lib-999-er-000001 (the 1st AIP in collection LIB-999)

2. Generate metadata about the files using DataAccessioner. The program must copy the files to another location in order to generate this metadata. This also ensures that a copy of the preservation masters remains available in case anything goes wrong while creating the AIP. Fill out the following fields in DataAccessioner:
    a. **Your Name:** also include position title if not regular staff (i.e. student assistant)
    b. **Accession Number:** AIP-ID
    c. **Collection Title:** AIP-ID and name (i.e. lib-999-er-000001_example)
    d. **Access to Directory:** AIP Staging Folder (where you want to copy files to)
    e. **Source/Directory:** the folder containing the files for the AIP (files to copy)
    f. **Source name/Identifier:** AIP-ID and name (same as collection title)
    g. **Dublin Core: Title.** Collection number (i.e. lib-999)

DataAccessioner ready to copy



3. Reorganize the file directory structure of the copies made by DataAccessioner. The purpose of this step is to create a consistent structure that works for several departments at our institution. This will allow us all to submit our AIPs to an institutional preservation storage system that is currently in development.
    a. Rename the top level folder to add _bag to the end (i.e. lib-999-er-000001_bag)
    b. Rename the folder with the files (i.e. lib-999-er-000001_example) to "objects"
    c. Create a folder named "metadata" at the same level as the objects folder
    d. Add _da to the name of the DataAccessioner output file ((i.e. lib-999-er-000001_da.xml) and move it into the metadata folder

4. Add additional metadata files to the metadata folder. Files should be saved as XML or in plain text format to allow them to be searchable in a future preservation system.
    a. Create the master metadata file. This is a minimum amount of information that all departments are required to provide and will be ingested into our preservation storage system in a structured way. All other metadata will only be keyword searchable.
        i. Open the xml generated by DataAccessioner in XML Blueprint
        ii. Run the XSLT stylesheet to create the master file
        iii. Save in the metadata folder. File naming convention is AIP-ID_master.xml (i.e. lib-999-er-000001_master.xml)
    b. Convert the deed of gift to a text file and save in the metadata folder. File naming convention is Collection-ID_dg.xml (i.e. lib-999_dg.xml)
    c. Convert any other relevant metadata files to text files and save in the metadata folder. File naming convention is AIP-ID_code for metadata type. Could include preservation information, rights information, etc.

5. Bag the contents of the AIP (files and metadata) using Bagger
    a. Click "Create Bag in Place"
    b. Click the "Browse" button next to "Select Data" and select the folder with all of the files (lib-999-er-000001_bag in our example) and click "ok"
    c. The bag will be created
    d. Click "Save Bag"
    e. Click "Validate Bag" and then "Is Bag Complete" to verify the bag is correct
    f. If desired, additional metadata can be added to the bag using the Bag-Info tab

File Directory Structure Throughout the Process

Structure created by DataAccessioner (Step 2)

```
lib-999-er-000001
    libl-999-er-000001_example
        Chart in Microsoft Word.xlsx
        File 1.docx
        File 2.docx
        File 3.docx
        File 4.docx
        File 5.docx
lib-999-er-000001.xml
```

Reorganized structure (Step 3)

```
lib-999-er-000001_bag
    metadata
        lib-999-er-000001_da.xml
    objects
        Chart in Microsoft Word.xlsx
        File 1.docx
        File 2.docx
        File 3.docx
        File 4.docx
        File 5.docx
```

Structure of the completed bag (Step 5)

```
lib-999-er-000001_bag
    data
        metadata
            Lib-999_dg.xml
            Lib-999-er-000001_da.xml
            Lib-999-er-000001_master.xml
        objects
            Chart in Microsoft Word.xlsx
            File 1.docx
            File 2.docx
            File 3.docx
            File 4.docx
            File 5.docx
    bag-info.txt
    bagit.txt
    manifest-md5.txt
    tagmanifest-mdf.txt
```

XSLT to create master metadata file from DataAccessioner output

Note: the grey text between <!-- --> are comments. For each piece, they explain what information is being pulled from the DataAccessioner output, the PREMIS being generated, and if it is required or optional in our institution's implementation. The rules for optional portions are within comments. To make them work, the --> should be moved to the end of the explanation so that the rules are outside the comment and will be acted upon.

```
<xsl:stylesheet
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0"
 exclude-result-prefixes="xs"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:premis="info:lc/xmlns/premis-v2">

<xsl:template match="/">

<master>

<!-- gets AIP title from dc:title field in DataAccessioner XML (mandatory). -->

<dc:title><xsl:value-of select="//dc:title"/></dc:title>

<premis:object>

<!-- gets AIP identifier from accession number attribute in DataAccessioner XML (mandatory). PREMIS 1.1 -->
<premis:objectIdentifier>
        <premis:objectIdentifierType>Locally defined identifier</premis:objectIdentifierType>
        <premis:objectIdentifierValue><xsl:value-of select="//*[local-name() =
'accession']/@number"/></premis:objectIdentifierValue>
</premis:objectIdentifier>

<!-- inserts default object category of aggregate (mandatory). Change if needed using controlled vocabulary.
PREMIS 1.2 -->
<premis:objectCategory>aggregate</premis:objectCategory>

<!-- inserts tags for preservation level (optional). PREMIS 1.3
<premis:preservationLevel>
        <premis:preservationLevelType>NEEDS-DATA</premis:preservationLevelType>
        <premis:preservationLevelValue>NEEDS-DATA</premis:preservationLevelValue>
        <premis:preservationLevelRole>NEEDS-DATA</premis:preservationLevelRole>
        <premis:preservationLevelRationale>NEEDS-DATA</premis:preservationLevelRationale>
        <premis:preservationLevelDateAssigned>NEEDS-DATA</premis:preservationLevelDateAssigned>
</premis:preservationLevel> -->

<!-- starts PREMIS 1.5-->
```

```
<premis:objectCharacteristics>

	<!-- gets every composition level from DataAccessioner XML (optional) and removes duplicate values.
PREMIS 1.5.1 -->
	<xsl:for-each select="//premis:compositionLevel[not(.=preceding::premis:compositionLevel)]">
		<xsl:copy-of copy-namespaces="no" select="."/>
	</xsl:for-each>

	<!-- gets every size from DataAccessioner XML and adds the values to give total size of AIP in bytes
(optional). PREMIS 1.5.3 -->
	<premis:size><xsl:value-of select="format-number(sum(//premis:size),'#')"/></premis:size>

	<!-- gets every format from DataAccessioner XML and removes duplicates based on format designation
(name and version). (mandatory). PREMIS 1.5.4-->
	<xsl:for-each
select="//premis:format[not(./premis:formatDesignation=preceding::premis:format/premis:formatDesignation)]">
		<xsl:copy-of copy-namespaces="no" select="."/>
	</xsl:for-each>

	<!-- inserts tags for creating application (optional). PREMIS 1.5.5
	<premis:creatingApplication>
		<premis:creatingApplicationName>NEEDS-DATA</premis:creatingApplicationName>
		<premis:creatingApplicationVersion>NEEDS-DATA</premis:creatingApplicationVersion>
		<premis:dateCreatedByApplication>NEEDS-DATA</premis:dateCreatedByApplication>
		<premis:creatingApplicationExtension>NEEDS-DATA</premis:creatingApplicationExtension>
	</premis:creatingApplication> -->

 <!-- inserts tags for inhibitors (mandatory if applicable). PREMIS 1.5.6
	<premis:inhibitors>
		<premis:inhibitorType>NEEDS-DATA</premis:inhibitorType>
		<premis:inhibitorTarget>NEEDS-DATA</premis:inhibitorTarget>
		<premis:inhibitorKey>NEEDS-DATA</premis:inhibitorKey>
	</premis:inhibitors> -->

</premis:objectCharacteristics>

<!-- gets collection ID from collection name attribute in DataAccessioner XML (mandatory). PREMIS 1.13 -->
<premis:relationship>
	<premis:relationshipType>structural</premis:relationshipType>
	<premis:relationshipSubType>Is Member Of</premis:relationshipSubType>
	<premis:relatedObjectIdentifier>
		<premis:relatedObjectIdentifierType>Object identifier
value</premis:relatedObjectIdentifierType>
		<premis:relatedObjectIdentifierValue><xsl:value-of select="//*[local-name() =
'collection']/@name"/></premis:relatedObjectIdentifierValue>
	</premis:relatedObjectIdentifier>
```

```
</premis:relationship>

<!-- inserts default repository (mandatory). PREMIS 1.13 -->
<premis:relationship>
	<premis:relationshipType>structural</premis:relationshipType>
	<premis:relationshipSubType>Is Member Of</premis:relationshipSubType>
	<premis:relatedObjectIdentifier>
		<premis:relatedObjectIdentifierType>Object identifier
value</premis:relatedObjectIdentifierType>
		<premis:relatedObjectIdentifierValue>RBRL</premis:relatedObjectIdentifierValue>
	</premis:relatedObjectIdentifier>
</premis:relationship>

<!-- inserts tags for additional relationships (optional). PREMIS 1.13
<premis:relationship>
	<premis:relationshipType>NEEDS-DATA</premis:relationshipType>
	<premis:relationshipSubType>NEEDS-DATA</premis:relationshipSubType>
	<premis:relatedObjectIdentifier>
		<premis:relatedObjectIdentifierType>Locally defined
identifier</premis:relatedObjectIdentifierType>
		<premis:relatedObjectIdentifierValue>NEEDS-DATA</premis:relatedObjectIdentifierValue>
		<premis:relatedObjectSequence>NEEDS-DATA</premis:relatedObjectSequence>
	</premis:relatedObjectIdentifier>
</premis:relationship> -->

</premis:object>
</master>

</xsl:template>

</xsl:stylesheet>
```