DIGITAL LIBRARIES
University Libraries
UNT

UNT Libraries AIP-to-DIP/ACP Conversion Workflow

Date: October 2015

Version: 1.0

Contributors:

Mark Phillips Assistant Dean for Digital Libraries
Hannah Tarver Department Head, Digital Projects Unit
Ana Krahmer Supervisor, Digital Newspaper Unit
Daniel Alemneh Supervisor, Digital Curation Unit
Laura Waugh Repository Librarian for Scholarly Works

UNT Libraries AIP-to-DIP/ACP Conversion Workflow

Introduction

The UNT Libraries has a formal definition of the required features and elements for an Archival Information Package (AIP) documented in the UNT Libraries OAIS Information Package Specification. That document, while describing what should be present in a valid Archival Information Package (AIP) and Dissemination Information Package (DIP), does not cover the process used for converting an AIP to a DIP. The goal of this document is to establish the process and workflow for the conversion of an AIP to a DIP for the UNT Libraries' Digital Collections. Historically the UNT Libraries has referred to its Dissemination Information Package (DIP) locally as the Access Content Package (ACP) and for the purpose of this document they are equivalent and can be used interchangeably.

Ingest Dropbox

Dropboxes define the steps involved in the ingest process of our digital content. They are responsible for creating the AIPs and DIPs for our repository, locally referred to as the Coda repository and Aubrey access systems. A dropbox encapsulates a series of steps that are executed, in sequential order, on the submitted digital objects. These steps are represented by folders in the dropbox that hold the input, output, and errors for each step. The management python scripts, such as makeAIP.py and makeACP.py, are used to convert a SIP to an AIP and an AIP to a DIP/ACP, respectivly.

The conversion process proceeds after the successful conversion from a submitted Submission Information Package (SIP) to an AIP that is covered in the document UNT Libraries SIP-to-AIP Conversion Workflow. As can be seen from the dropbox organization list below, this newly created AIP resides in the 3.ToACP folder, which is designated by Dropbox during ingest.

A dropbox is organized like this:

```
dropbox/
|-- 0.Staging/
|-- 1.ToAIP/
|-- 2.ToAIP-Error/
|-- 3.ToACP/
|-- 4.ToACP-Error/
|-- 5.ToArchive/
|-- 6.ToAubrey/
|-- 7.ToAubreySorted/
|-- 8.ToAubreySorted-Error/
|-- dropbox_config.py
|-- makeACP.py
|-- makeACPSort.py
|-- makeAIP.py
`-- moveToCODA.sh
```

Verification steps

The verification steps that makeACP.py executes before processing an AIP include the following:

1. Checks to see that the proper utilities and versions are installed on the ingest server.
2. Check the Namaste tag in the bag to make sure it is an AIP bag
3. Quickly validate the BagIt bag using the oxnum value in the bag-info.txt file.

Objects to convert to DIP are present from the previous SIP to AIP process and reside in the 3.ToACP folder, the makeACP.py script is exectuted, which walks through the three steps listed above to determine if a supplied AIP can be converted into a UNTL-DIP.

If makeACP.py fails on check 1 in the above list, the whole process stops, if this step passes then makeACP.py will start multiple sub-processes that allow for parallel processing of digital objects and therefore takes advantage of available processors on the ingest server. If there is an error in either step 2 or 3 in the list above, the process moves the object to the 4.ToACP-Error folder with a log file describing the error encountered and continues to process the next object in the queue. When makeACP.py is finished running it will let the operator know how many AIPs were processed, the number that were successfully converted into a DIP/ACP, and the number that failed.

AIP to DIP/ACP Conversion

Once the makeACP.py script verifies that the submitted AIP is valid and well formed, it executes a series of steps to complete the DIP creation. The steps include the following:

1. Retrieve instructions from coda_process.py
2. Based on the primary bitstream in each fileSet, create a web derivative file format for each fileSet.
3. If a metadata.xml record is not present generate a generic metadata.xml record.
4. Generate a UNTL METS Dissemination Information Package Profile XML document.

If all of these steps are completed successfully then a DIP/ACP is created in the 6.ToAubrey folder and the AIP that was processed is moved to the 5.ToArchive folder. If there were any errors encountered, the AIP is moved to the 4.ToACP-Error folder and the issue is logged with the object so it can be fixed and reprocessed in the future.

Example AIP and DIP/ACP

Input AIP

```
metapth1234/
|-- 0=UNTL_AIP_1.0
|-- bag-info.txt
|-- bagit.txt
|-- coda_directives.py
|-- data/
|   |-- data/
|   |   `-- 01_tif/
|   |       |-- 1999.001.001_01.tif
|   |       `-- 1999.001.001_02.tif
|   |-- metadata/
|   |   |-- 17711fdb-2e25-4566-bf3f-daa172a12190.jhove.xml
|   |   `-- 9639acae-397a-4c90-8851-52b6f04c4d8d.jhove.xml
|   |-- metadata.xml
|   `-- metapth1234.aip.mets.xml
`-- manifest-md5.txt
```

Resulting DIP/ACP

```
metapth1234/
|-- 0=UNTL_ACP_1.0
|-- bag-info.txt
|-- bagit.txt
|-- coda_directives.py
|-- data/
|   |-- metapth1234.mets.xml
|   |-- metapth1234.untl.xml
|   `-- web/
|       `-- 01_tif/
|           |-- 1999.001.001_01.jpg
|           |-- 1999.001.001_01.medium.jpg
|           |-- 1999.001.001_01.square.jpg
|           |-- 1999.001.001_01.thumbnail.jpg
|           |-- 1999.001.001_02.jpg
|           |-- 1999.001.001_02.medium.jpg
|           |-- 1999.001.001_02.square.jpg
|           `-- 1999.001.001_02.thumbnail.jpg
`-- manifest-md5.txt
```