Ingest

Ingest

Basics

Name:	**Ingest**
Version:	1.00 (2016-10-25)
Status:	production
Specification:	Merritt Ingest Service
Download:	Not available
More information:	Curation home page

The Ingest micro-service provides a means to add new digital content into the curation environment for active management by the Program. Using terms defined by the Open Archival Information System (OAIS) reference model, the Ingest service accepts Submission Information Packages (SIPs) and converts them into Archival Information Packages (AIPs). This process may involve the use of other micro-services: the Transformation service to transcode objects into normative forms defined by internal standards; the Characterization service to produce relevant descriptive information for management in the Inventory service; and the Storage service to manage object files.

The Ingest micro-service defines the following conceptual entities:

- Batch. A set of jobs.
- Job. The processing of a single digital object.
- Profile. An indication of the type of digital object being ingested.
- Handler. A specific ingest sub-task.
- Notification. The final status resulting from an ingest.

A Ruby library for submitting objects to the Merritt ingest system has been developed; please see the bitbucket page http://bitbucket.org/merritt/mrt-ingest-ruby/ for more information.

API

Programmatic submission to the Ingest service can be made using the API. As described in the Ingest specification document, the request arguments are:

Argument	**Value**
filename	(optional) The name of the file
file	The file itself, i.e. its contents
type	Valid values: • file • container • object-manifest • batch-manifest • container-batch-manifest • single-file-batch-manifest
profile	The submission profile, which we will provide to the submitter
primaryIdentifier	(optional) ARK identifier, if known
localIdentifier	(optional) local identifier, if known

digestType	(optional) valid values: • adler-32 • crc-32 • md2 • md5 • sha-1 • sha-256 • sha-384 • sha-512
digestValue	(optional) digest value, hex-encoded string
creator	(optional) creator
title	(optional) title
date	(optional) date
note	(optional) descriptive note
responseForm	(optional) valid values: • anvl • csv • json • turtle • xhtml • xml

All of the enumerated values (file type, digest type, response form) are case-insensitive.

Sample cURL

curl --silent -u user:password \
-F "file=@ucsf_etd_200609.checkm" \
-F "type=container-batch-manifest" \
-F "submitter=username" \
-F "responseForm=xml" \
-F "profile=merritt_demo_content" \
-F "localIdentifier=local-ID-test" \
https: //merritt-stage.cdlib.org/object/update

Using the secure https connection with curl may require additional certificates. Please contact UC3 with any questions.

METS feeder

The steps to submitting content using METS:

1) First, create a METS document that points to all of the component parts of a digital object, using one of the profiles described in the CDL Guidelines for Digital Objects. The METS documents and all of the digital object components should be on a web-accessible server. If you need to open up a firewall, we can give you the addresses of the machines that need to access the files.

2) Create a "manifest," which is a just simple list of all of the URLs for METS documents. (This list must be on a web-accessible server also.) The METS manifest is a text file with line breaks between each METS document, looking like this:

http: //URL/subdir/metsfile1.xml
http: //URL/subdir/metsfile2.xml
etc.

3) Then send the URL of the manifest to the feeder. The URL should have this format:

http://<feederURL>/mets?userID=<userID>&authCode=<authCode>&accessGroupID=<accessGroupID>&manifestURL=<manifestURL>

with these elements:

1. feederURL (for Merritt stage, this is merritt-stage.cdlib.org/feeder-mets/; for production, merritt.cdlib.org/feeder-mets/
2. userID (login)
3. authCode (password)
4. accessGroupID (collection name--we can let you know the collection name) (nb–without the _content suffix)
5. fillin (Optional. if "false", this will suppress fixity check of object components by feeder, speeding up the submission. The implicit default is &fillin=true)
6. manifestURL (with URL encoding replacing ":" (%3a) "/" (%2f) etc)

To fill in some of the variables, it would look like this (no line breaks):

curl "http: //merritt.cdlib.org/feeder-mets/mets/?userID=yourLogin&authCode=yourAuthCode&accessGroupID=xxxxx&manifestURL=http%3a%2f%URL%2fsubdirectory%2fFilename"

4) We will retrieve the list and ingest the METS documents.

Polling status

Basics

Provide status of a Merritt Ingest submission during and after processing. Status responses include:

- PENDING: Queued for Ingest service
- CONSUMED: Currently in process
- COMPLETED: Successfully processed
- FAILED: Unsuccessfully processed

Polling REST API

Summary
/istatus/bid/BID/job
/istatus/bid/BID/jobfull
/istatus/bid/BID/primary
/istatus/bid/BID/primaryfull
/istatus/bid/BID/local
/istatus/bid/BID/localfull

Single object
/istatus/bid/BID/job/JID
/istatus/bid/BID/jobfull/JID
/istatus/bid/BID/primary/PID
/istatus/bid/BID/primaryfull/PID
/istatus/bid/BID/local/LID
/istatus/bid/BID/localfull/LID

- Other requests are passed directly through to couchDB. See: http://wiki.apache.org/couchdb/HTTP_Document_API
- Job, Primary and Local IDs (JID, PID, LID) must be JSON URL encoded and double quoted. For example, %22localid001%22 for localid001.

Authentication

Supports HTTP Basic. Credentials provided by Merritt team.