Interoperability and Collection of Preservation Metadata for Digital Repository Content

Matt Cordial, Tom Habing, Bill Ingram, Robert Manaster
University of Illinois Urbana-Champaign
cordial;thabing;wingram2;manaster@uiuc.edu
Outline

• Conceptual Overview
  – Conceptual diagrams
  – Framework architecture

• Components
  – METS Profiles (The Hub)
  – Content processing, transformation, and metadata generation (Spokes)
  – Facilitating repository ingestion and dissemination (The Handoff)

• Conclusion
  – Finding more information
A Birds-eye View
For $N$ different repositories that need to interoperate, this model reduces the complexity from $N^2(N-1)$ to $2^N$.

This simple idea is the rationale for many different standards that aim to promote interoperability.
The Essentials

• Extensible METS profile
• Repository specific processing and transformation utilities
• Java API for local integration and extensibility
  – Apache XML Beans
• Dissemination/Submission Web-service
  – RESTful
Functional Overview

METS Profiles

Technical Metadata Augmentation
- TechMD Augmenter
- JHOVE
- XSLT

Processing and Transformation
- METS Construction
- Descriptive Metadata Augmentation
- Bitstream Verification
- Profile Validation

Handoff
- Web Service Client
- Web Service

Repository

from hub to hub
METS Profiles

- Non-prescriptive in regards to structure or file formats
- Intended to overlay other profiles which specify case-specific needs (i.e. web captures)
- PREMIS
- MODS
  - Must conform to the DLF Aquifer profile
- File-format specific technical metadata
  - MIX, VIDEOMD, AUDIOMD, others as appropriate
Functional Overview

METS Profiles

Technical Metadata Augmentation
- TechMD Augmenter
- JHOVE

Processing and Transformation
- METS Construction
- Descriptive Metadata Augmentation
- Bitstream Verification
- Profile Validation

Handoff
- Web Service Client
- Web Service

Repository

to hub

from hub
Technical Metadata Generation/Augmentation

- JHOVE Output + Custom XSLT
- Java “Applicators” for specific technical metadata schemas
  - MIX
  - TEXTMD
  - AUDIOMD
  - PREMIS
- Class hierarchy to support new Applicators
Functional Overview

METS Profiles

Technical Metadata Augmentation

- TechMD Augmenter
- JHOVE
- XSLT

Processing and Transformation

- METS Construction
- Descriptive Metadata Augmentation
- Bitstream Verification
- Profile Validation

Handoff

- Web Service Client
- Web Service

Repository

from hub to hub
To-Hub Processing

Data Store / DIPs

Generate/collect provenance metadata

Extract format-specific technical metadata

Generate/collect digital provenance metadata

Embed links to digital items

Model structure of the item

Embed native metadata

Transform/enrich native metadata

Hub

<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<metadata ID="MDO">
  <xref ref-type="text" xlink:href="http://www.loc.gov/"/>
  <xref ref-type="description" xlink:href="http://www.lib.msu.edu/contactus"/>
  <xref ref-type="publisher" xlink:href="http://www.lib.msu.edu/contactus"/>
  <xref ref-type="subject" xlink:href="http://www.lib.msu.edu/contactus"/>
  <xref ref-type="title" xlink:href="http://www.lib.msu.edu/contactus"/>
</metadata>
From-Hub Processing

Generate provenance metadata

Hub

Transform hub metadata to repository-compatible metadata

Assemble into packages for repository ingest

Add the METS file as an item in the submission package

SIPs

metadata.xml

hubMets.xml

Generate provenance metadata

Transform hub metadata to repository-compatible metadata

Assemble into packages for repository ingest

Add the METS file as an item in the submission package

hubMets.xml

metadata.xml

SIPs
Functional Overview

METS Profiles

Technical Metadata Augmentation

Processing and Transformation

Handoff

Repository

METS Profiles

TechMD Augmenter

JHOVE

XSLT

METS Construction

Descriptive Metadata Augmentation

Bitstream Verification

Profile Validation

Web Service

Web Service Client

to hub

from hub
• Packages usable by a repository’s native ingestion routines

• REST Web-Service
  – Client integrated into processing workflow
  – DSpace, EPrints, and others in the next year
  – Specification and API to create service for other repository systems
Client submits a GET request to LRCRUD service for a specific item.

Service calls the native DSpace dissemination routine.

Service receives the output from the dissemination, creates a header file, and adds both the header file and the disseminated content to a zip-file.

Service returns a zip-file containing the package to the client.
**Create stub record**

2) Client issues a POST request to LRCRUD specifying “where” to create the record (e.g. communities or collections) if needed

3) LRCRUD calls the native Fedora creation routine

4) Fedora supplies LRCRUD with the ID for the newly created record

5) LRCRUD responds to the client with an HTTP 201 “Created” message and returns the ID in the Location: header

**Upload and ingest the item**

8) Client issues a PUT request to LRCRUD to replace the package identified by the URI. The entity body of the request must contain the zip-file containing the package to be ingested.

9) LRCRUD unpacks the files and calls the native Fedora ingestion routine.

10) Fedora tells LRCRUD that ingestion was successful

11) LRCRUD responds to the client with an HTTP 204 “No Content” message indicating that the request was successful.
More Information

Open Source Code:
http://sourceforge.net/projects/echodep

LRCRUD Service Specification:
http://dlgrainger.uiuc.edu/echodep/hns/LRCRUDS.htm

METS Profiles:
Generic -
http://www.loc.gov/standards/mets/profiles/00000015.xml

Web Capture -
http://www.loc.gov/standards/mets/profiles/00000016.xml

Java API Documentation (Javadoc):
http://echodep.sourceforge.net/javadoc/index.html

Project Web Site
http://ndiipp.uiuc.edu/
Questions?

Matt Cordial
cordial@uiuc.edu