Object Analysis

in the Merritt Digital Preservation Repository

Document version: 0.5
Last modified: 2024-06-25

About

In-depth curation across the wide range of collections being preserved in the Merritt repository is a task that stands to benefit from the assistance of an automated process capable of providing regular reports to depositors with key insights into object and collection composition.

Enabling the generation of such reports based on object analysis can provide a baseline status for existing objects and collections whose establishment may or may not predate current depositors at the variety of UC Libraries and affiliated memory organizations that use Merritt.

The Merritt team has built a prototype which analyzes all versions of digital objects in the system on a per-collection basis. Its output is presented using Amazon OpenSearch dashboards. The analysis process consists of three primary steps: Object Build, Analyze, and Test.

In the **Build** step, all available data describing an object is gathered from the system's Inventory database and then saved to a JSON file using a custom schema. The same data is then run through the **Analyze** and **Test** steps whose results are then used to update the JSON. Data from fields in the JSON file are then added to an OpenSearch index to create a rich map of relationships across fields for display and exploration in dashboards – for example, how many objects in a collection contain both a recognized metadata sidecar file, along with multiple high-resolution images. Or, which objects in the collection include a complete record of object-level metadata noting Title, Creator, Date and a Local Identifier.

This document provides an explanation of the object Analysis and Test output as it is presented in OpenSearch, with a goal of enabling depositors to request specific information about the objects in their collections for use as a basis for potential preservation decisions and actions.

It also documents sample analysis requests as a means to get started, along with information about how the service may be tailored to provide results that may be more pertinent to campus collections and their overall holdings in Merritt.

I. Object Analysis and Test Definitions

Each of the sections that follow record the potential output of either the Analysis or Test steps that are executed against digital objects.

Object Classification: Analysis

During the **Analysis** step, objects are classified into the following buckets.

Common terms

- Content file: Any file in an object that was introduced by the Producer as part of the Submission Information Package (SIP), according to the OAIS model. Each object (AIP) in Merritt is represented by both Producer files and System files, the latter of which are automatically generated by the repository for tracking purposes.
- Derivative: A lower resolution file derived from the high resolution preservation/data file, often saved in formats that introduce lossy compression; purposed for access.
- MIME types: A two-part identifier for file formats and format contents transmitted on the Internet. Their purpose is somewhat similar to file extensions in that they identify the intended data format. Also commonly known as a media type.
- Preservation file: The primary, high-resolution data file stored in an object.

Object Classification Buckets

complex_object
A complex object: A container file is found in an object, or the object contains several preservation files of multiple mime types.

has_multi_digital_files_with_derivatives
The object contains multiple preservation files of the same MIME type. Derivative files are also present.

has_multi_digital_files
The object contains multiple preservation files of the same MIME type.

has_digital_file_with_derivatives
Preservation file and derivative files present in an object.

has_derivatives_only
Only derivative content files present in an object.

has_single_digital_file
Only one identifiable content file is present in the object.

has_no_content
No identifiable content files were found in the object. Recognized metadata files may be present.

Metadata Classification: Analysis

During the **Analysis** step, the metadata files in objects are classified into the following buckets.

Common terms

- Metadata sidecar file: A descriptive metadata file associated with an object that includes information about the object's content as well as collection context, and/or a preservation metadata file that contains technical metadata and metadata regarding the preservation process.
- Merritt metadata file: A uniquely named Electronic Resource Citation text file (mrt-erc.txt) that contains object-level metadata: Title, Creator, Date and one or more Local Identifier entries.

Metadata Classification Buckets

has_bag_metadata_file
A BagIt metadata file derived from a bagged object is present. i.e. bag-info.txt

has_common_metadata_file
A "Common" Merritt metadata file is present in the Producer directory of the AIP and DIP. Recognized files include: mets.xml, mets.txt, mrt-dc.xml, cdlmeta.tar.gz.

Note that the repository also automatically generates a mrt-erc.txt file that is stored as a System file in the AIP and DIP. It contains the same facets of metadata which may be updated by subsequent depositor-submitted mrt-erc.txt files.

has_etd_metadata_file
The object contains a metadata sidecar file for an ETD submission. The naming convention in use for this sidecar file is consistent across all ETD collections in the system, for example: Smith_ucmerced_0000D_00000_DATA.xml

has_nuxeo_style_metadata_file
An XML metadata sidecar file for a Nuxeo submission is present in the object. The naming convention in use for this file is consistent across all Nuxeo-to-Merritt direct deposit collections. The file name incorporates a unique Nuxeo-provided UUID.

has_metadata_with_secondary
The object contains an identifiable, primary metadata sidecar file, as well as another metadata file, such as a depositor-submitted Merritt ERC file (mrt-erc.txt).

has_single_metadata_file
The object contains an identifiable, primary metadata sidecar file, such as the aforementioned BagIt, ETD, or Nuxeo style metadata files, or mets.xml, mets.txt, mrt-dc.xml, or cdlmeta.tar.gz.

has_multi_metadata
The object contains multiple, potential metadata sidecar files.

has_secondary_metadata_only
Files containing metadata that would not be classified as a sidecar file are present in the object. A sidecar file is assumed to have richer metadata than a Merritt ERC file (mrt-erc.txt).

has_no_sidecar_metadata
The object contains no identifiable metadata file (apart from the System mrt-erc) and instead contains only preservation data files and/or derivatives.

Object Analysis: Tests

During the **Test** step, data collected about the object thus far, including format information, file extension and classification results is run through a series of tests. These tests are purposed to provide additional insight into the contents and composition of an object. Once completed, each test results in one of four statuses: Pass, Info, Warn and Fail.

At a high level, these statuses imply:

- Pass: An object is in an optimal state
- Info: Object state is not ideal, yet action does not necessarily need to be taken
- Warn: Action recommended; may be a recurring issue
- Fail: Action should be taken

Test Status Definitions – Specific criteria for each

Pass (all criteria met)

- The MIME type of all files in the object are recognized as being sustainable, or are indicated as a Preferred or Acceptable format according to the Library of Congress Recommended Formats Statement (LOC RFS).
- The extensions of all files in the object are according to extensions typically associated with the MIME type of the file at hand.

- Through classification, an object is observed to include multiple preservation files with or without derivatives, a single preservation file with or without derivatives, or simply has a single preservation file.
- Through classification, an object is observed to include a single, recognized “Common” metadata file, a bag-info.txt file, an ETD metadata file, a Nuxeo-style metadata file, or one of these along with a secondary metadata file. e.g. mets.xml + mrt-erc.txt
- The object incorporates all object-level metadata: Title, Creator, Date, Local Identifier

Info (any single criterion met)

- The MIME type tests noted in the Pass section, above, are successful.
- Through classification, an object is considered to be a Complex object that includes a container file (e.g. .zip or .tar) or preservation files of multiple MIME types.
- Through classification, an object is found to have multiple, potential metadata sidecar files.
- A file was deleted from any version of the object.
- An empty System file is found in the AIP.
- The object’s Title field indicates “Untitled,” or “Title Unknown”
- The object’s Date field indicates “Undated,” “n.d.,” or “Date not indicated”

Warn (any single criterion met)

- A file in the object does not conform to the recognized list of sustainable MIME types or is not indicated as a Preferred or Acceptable file format according to the LOC RFS.
- The extensions of all files in the object are according to extensions typically associated with the MIME type of the file at hand.
- Through classification, the object is found to include only derivative file formats.
- Through classification, the object is found to include only metadata files recognized as secondary metadata, or contains no sidecar metadata file.
- An empty Producer file is found in the AIP.
- One or more file names in an object resemble a URL.
- The object contains two or more files with the same checksum.
- One or more object-level metadata fields is empty (Title, Creator, Date, and/or Local Identifier).

Fail (any single criterion met)

- A mismatch occurs between the known MIME type for a file and its file extension.
- The MIME type for a file cannot be found.
- The object contains no recognizable preservation files or derivatives.
- A file in the object does not have a file extension.
- One or more object-level metadata fields (Title, Creator, Date, Local Identifier) contains an empty string surrounded by quotes.

II. Sample Analysis Requests

Object analysis is performed on a per-collection basis. Output from the analysis prototype is captured in a number of OpenSearch dashboards. These may be tailored to a specific analysis request. Request output of any dashboard is made available via an exported PDF. What follows is a table of sample analysis requests.

Request	**Definition**
Missing meaningful object-level metadata	Capture a list of objects that are missing one or more fields of object-level metadata (Title, Creator, Date, Local Identifier)
Unsustainable MIME types	Capture a list of objects containing one or more files whose MIME type is considered to be unsustainable or is no longer deemed appropriate for long term preservation purposes.
MIME/extension mismatch	Capture a list of objects containing one or more files whose existing file extensions are not recognized as pertaining to the MIME type of the file(s) at hand.
Warnings related to unexpected MIME and file extension pairings	Capture a list of objects containing one or more files whose extensions may be feasible for the MIME type at hand, but not ideal. For example, a .txt extension appears on an XML file.
Missing expected derivatives	In the context of a collection where all or the majority of objects should include derivatives, capture those in which derivatives are missing.
Only derivatives are present	Capture a list of objects that do not contain preservation files and instead only contain derivatives.
MIME type search	List all objects in a collection that contain files of a specific MIME type, or those that are missing said type.
Missing sidecar metadata	Capture a list of objects that are missing a known or recognizable (based on naming convention) metadata sidecar file.
Empty and unwanted files	Capture a list of objects that contain empty files (0-byte) or files of an unwanted MIME type. e.g. .ds_store, thumbs.db, etc.
Enumerate objects based on metadata type	Capture a list of objects that contain a specific type of metadata file. e.g. bag-info.txt, mets.xml, etc.

Note that larger collections require more lead time to load into the analysis tool. If new objects are actively being ingested into a collection, there may be some lag on when these are subsequently analyzed.

Once initial analysis recommendations are returned, we will collaborate with the collection owner to refine the analysis and test routines based on what is learned from each collection.

Analysis requests will be estimated and prioritized in order to provide feedback across collection owners and campuses.

III. Customization

Object and metadata classification, as well as object tests supported by the prototype may be augmented to enhance their effectiveness based on the collection that is to be analyzed. For example, if a series of objects contain .iiq files, it may be desired to flag these with a Warn test result, even though these files are structurally equivalent to TIFF images. This would be considered a case where a MIME type may not match a recognized file extension. Assigning the Warn test status would then surface a list of objects that contain these files. In this example, a Fail test status could also be assigned. Any recognized MIME type may have multiple, potential file extensions associated with it.

Separately, it may be desirable to flag objects with an Info, Warn or Fail test status if they contain a file of a specific MIME type. Any recognized type may be associated with a specific test status. For example, if only a specific type of derivative is appropriate for objects in a collection, but some objects contain derivatives of a different file type, these objects could be enumerated and marked with a Warn or Fail status.

In another example, it may be desirable to recognize preservation files, derivatives or metadata files that conform to specific naming conventions. Additional conventions may be added to the prototype to increase its ability to surface objects in a collection that do not conform to the expected convention(s).

These are only a few examples of customization. However, the classification and test steps taken by the prototype are highly configurable. Depositors are encouraged to suggest and share potential customizations that may be beneficial across collections or across a broader range of holdings in the repository.

Appendix

Technical documentation is available which further explains the analysis prototype's system design, its data model and configuration and the actions taken during each step of the object analysis process. Further reading regarding Library of Congress format sustainability and descriptions, as well as the Merritt repository's workings and Amazon OpenSearch are also available.

Analysis System Design

Understanding the Object Health Data Model

OpenSearch Dashboard Walkthrough (video)

Object Analysis Codebase

Prototype Rules File (Yaml) – Used for configuration and customization

Library of Congress: Format Descriptions

Library of Congress: Sustainability of Digital Formats

Merritt Preservation System Documentation

Amazon OpenSearch Service Documentation