HARVARD
LIBRARY

Disk Image Content Model and Metadata Analysis

ACTIVITY 2: Metadata Analysis

DRAFT - 2016-03-24

Submitted By:

253 36th Street, Suite C302
Brooklyn, NY 11232

Telephone 1.917.475.9630

bertram@avpreserve.com
amy@avpreserve.com
April 2016

Introduction

This metadata analysis takes a multi-faceted approach to identify the informational elements that will be necessary to document the provenance, the technicalities, and the structural characteristics of digital content typically serialized as disk images for long-term storage in a preservation repository. The initial analysis itself is a generic examination of disk images as *file formats that are generated by contemporary software, reside within contemporary file systems, and are made available via contemporary operating systems*. One outcome of the generic examination includes selected informational elements, as well as a conceptual model to illustrate the nature of disk images. From the generic analysis, this examination will recommend specific approaches to structure the selected information elements from the conceptual model in a consistent and repeatable metadata scheme.

Building from the generic analysis, the examination will focus attention on modeling the disk image content and metadata in the context of Harvard University Library DRS Object Descriptors.

Conceptual Landscape

In the context of library and archive practice, disk images pose an intriguing problem of encapsulation. A disk image, by nature, is a unique sequence of bytes (or sequences of bytes) that encapsulates an existing organizational system of files, directories, and accompanying metadata. Because a disk image encompasses a system of files and directories and, yet, is itself a file within a file system, extractable metadata will be available at four levels (at least):

1) Capture file system level
2) Disk image file format level
3) Encapsulated file system level
4) Target directories and files level

These levels are all sources of metadata that could be mined for information about provenance, process history, technical, and, even, descriptive metadata. The following figure illustrates the conceptual landscape of disk images. We will use this conceptual model (and the designated vocabulary) as we look specifically at HUL's DRS context.

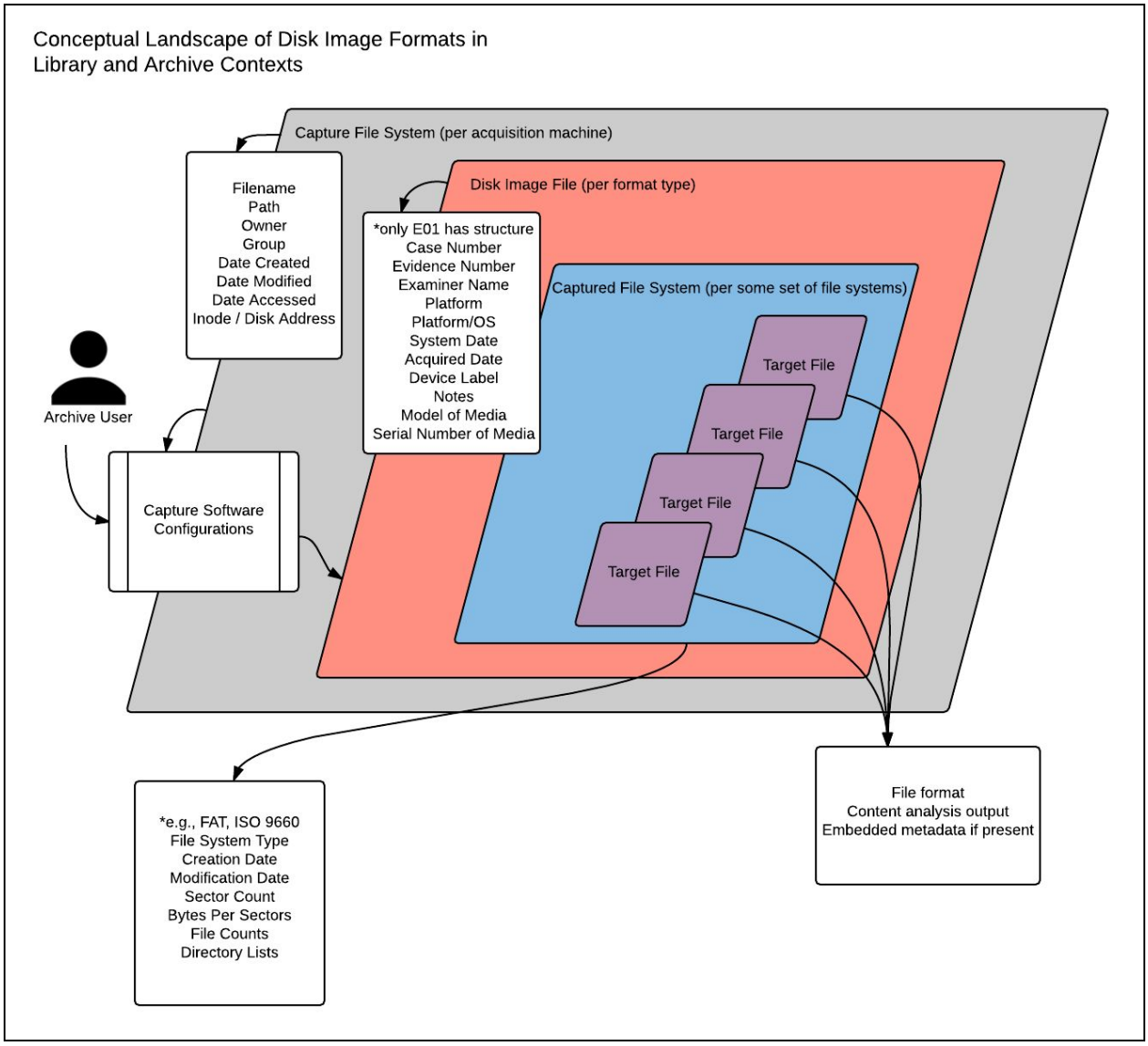

Figure 1. This illustration represents the sources of metadata that can be extracted for disk image files.

Analysis of metadata at Archive File System level (Grey in Figure 1)

All disk image files will have been generated, or captured, on a machine within a given file system. Using practices for file management, archivists can capture file system metadata before moving a disk image from an originating machine. Such an approach will ensure automated capture of the following information about the disk image file itself:

- Original file name
- Original file path
- Create date

- Modified date
- Last Accessed date
- Owner
- Group
- Permissions
- File size

Analysis of metadata at Disk Image File Format level (Red in Figure 1)

We deconstructed and evaluated each of the formats below to develop a detailed understanding of metadata available per format.

Disk Image Format metadata resources
RAW
E01
ISO[1] (which requires also a consideration of the CD-ROM specifications)[2]
BIN/CUE (which is truly just a RAW image with a side-car text file)

Only E01 is an actual binary format with structural and embedded technical and descriptive metadata.[3] The others are merely byte-by-byte captures of stored data, with no additional structure or packaging (meaning that the file format itself cannot provide any metadata of use). The following elements from E01 and the CUE text specification represent examples of the types of metadata that can be extracted at this level:

- Acquire Date
- Create Date
- Creator
- Capture Platform
- Capture Platform Operating System
- Case Number
- Description
- Device Label
- Evidence number
- Filename
- Notes

[1] http://wiki.osdev.org/ISO_9660; http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-119.pdf
[2] https://en.wikipedia.org/wiki/CD-ROM
[3] See https://docs.google.com/spreadsheets/d/1QufEjU8kF6oVqFU-Rsyq3AdsR3V_xYn3eL1joZjrmo8/edit?usp=sharing for a byte-by-byte breakdown of an example E01 file.

- Source
- Source ID
- Title

See **Metadata Comparison.diskimage-fileformat-eval** spreadsheet for more information.

Analysis of metadata at Filesystem level (Blue in Figure 1)

Each filesystem has its own specification for how bytes are organized, how files and directories are indexed, and for organizing files and directories within the geometry of the physical storage media on which the filesystem is written. The filesystem is the primary location where information is stored about files and directories.

In our analysis, we deconstructed two example file systems to understand the technical metadata available within file system structures that would be critical for reconstructing the file system and all of its underlying files and directories in the future.

We looked at FAT filesystems (12, 16, and 32) because these are most common in today's archive repositories.[4] FAT filesystems are typical of floppy disks. We also looked at ISO 9660 and Joliet filesystems because these are typical of optical disc media, which are also current targets of disk image activity in archives and libraries.[5]

Although each filesystem contains a distinct set of internal structural and technical metadata, our aim is to identify elements of information that will be critical for a user in the future to be able to mount, navigate, and/or extract files and other information of interest from a particular filesystem. Examples of metadata that can be extracted at this level include:

- Original Media Type
- File System Type
- Bytes per Sector of Original Media
- Sector Count on Original Media
- File system create date
- File system modification date

[4] See https://docs.google.com/spreadsheets/d/1sU8emvVcdOk4Nrf3X-owYJG-1GrUt0PcPAoHdXRa_sI/edit?usp=sharing bor a byte-by-byte breakdown of an example FAT 12 filesystem.

[5] See https://docs.google.com/spreadsheets/d/1DOh9BcSAfnz3eqLu7V-dSJKvhJSm--VmbnsK196ppOk/edit?usp=sharing for a byte-by-byte breakdown of an example ISO 9660 filesystem.

- File Counts
- File Names
- File and Directory Lists

Analysis of metadata at Target Directory and File level (Purple in Figure 1)

The target of disk imaging is often the encapsulated files and directories stored within a target filesystem. The variety of files and directories that will be encountered within a given disk image is wide and unpredictable. Contents could range from one stored file to hundreds of thousands, or millions, depending on the capacity of the target source media. Files themselves only provide information that the format specification establishes. This is unpredictable and varies format to format. Sometimes formats support embedded metadata that documents file formats, encoding formats, technical information, file size, descriptive information (e.g., TIFF, JPEG, MOV, PDF, WAV); other times, there is no embedded metadata available (e.g., HTML, XML, TXT, CSV). Additionally, the content itself can be analysed to understand information about the file. This is variable depending on the type of information contained within a file (e.g., textual data versus image data or audio data). Examples of metadata that might be extracted at a target file level include:

- Filename
- File path (although, this truly is concatenated from the file system)
- File size
- File format (usually this is constructed from an analysis of the file's bytes)
- Date Created
- Date Modified
- Descriptive metadata (Creator, Genre, Subject, etc.)
- Technical metadata (Sample Rate, Bits per Sample, Colorspace, Page Count, etc.)

This information can only be extracted by software (or humans) that can read the particular file system that was used to encapsulate the files. With such software, this information can be generated at any point in the process given that the goal of disk imaging is to capture the file system and therefore to fix the state of the encapsulated files and directories.

Describing Disk Images

Based on the disk image conceptual landscape as documented above, we recommend an approach to preservation metadata that extracts metadata from each of the four levels, and through concatenation or calculation, to produce a core set of metadata that will ensure the understandability and accessibility of an acquired disk image. In order to support the understandability and accessibility of encapsulated files within the disk image file, additional approaches will be defined to capture file and directory lists, format summaries, and the output of analysis tools on the encapsulated content of the disk image.

In terms of metadata to describe technical and structural characteristics of disk image formats, there are three levels of interest for description:

1) Disk Image File Format (e.g., EWF, RAW, ISO, BIN/CUE, etc.)

   An archive will need to describe the disk image file format itself in order to be able to understand how it is packaged and what software will be needed to read the format in the future. This will include a description of the particular format and version. As noted above, only EWF and CUE provide format-specific metadata that can be leveraged from the format itself. Additionally, an archive can describe the file by its size, the date it was created/modified/accessed, who created it, what software was used to create it, and other process history metadata about its creation. This information will come from either the file system where the disk image was stored upon capture or from an analysis using an external file identification tool.

2) Encapsulated File System(s) (e.g., FAT12, FAT16, FAT32, NTFS, ISO 9660, etc.)

   An archive will need to document the format of the file system that is encapsulated within the disk image file. This file system will have a byte structure that is unique to the file system format. The file system format is the key to unlocking the files and their structure as they are organized within the file system, which also holds provenance information about the files themselves (e.g., dates of creation/modification/access, creator, etc.). To that aim, we recommend the following information elements for documenting the characteristics of a given filesystem:

   - File System Format
   - Bytes per Sector
   - Sector Count

3) Target Directories and Files

Most often, the target of disk imaging processes are the files encapsulated within the target file system on the source media object. To describe the files, an archive will use information from the encapsulated file system (e.g., file name, file path, file create/modify/access date, file size, file extension, file creator), as well as any metadata that can be extracted from known file formats. Additionally, an archive can use file identification tools to document the format of a given file.

Archivists can use software to generate file and directory lists during the process of disk imaging, and/or a repository could centralize the generation of these file and directory lists upon ingest.

Harvard staff members are currently employing a variety of approaches to extract file and directory lists for content stored in disk images, including fiwalk (as part of BitCurator), FTK Imager (one of the outputs is a list of files), Karen's Directory Printer (generates a list of files). Only one of these approaches makes use of any file identification processes to report file format (i.e., fiwalk, which uses libmagic to perform identification).[6] File format identification and characterization (and/or validation) will be of interest to the repository, at least to report how many of each type of format are contained within the disk image.

The following illustration draws attention to the targets of description for disk image content.

[6] Archivematica sidesteps libmagic in favor of FIDO for file format identification within fiwalk.

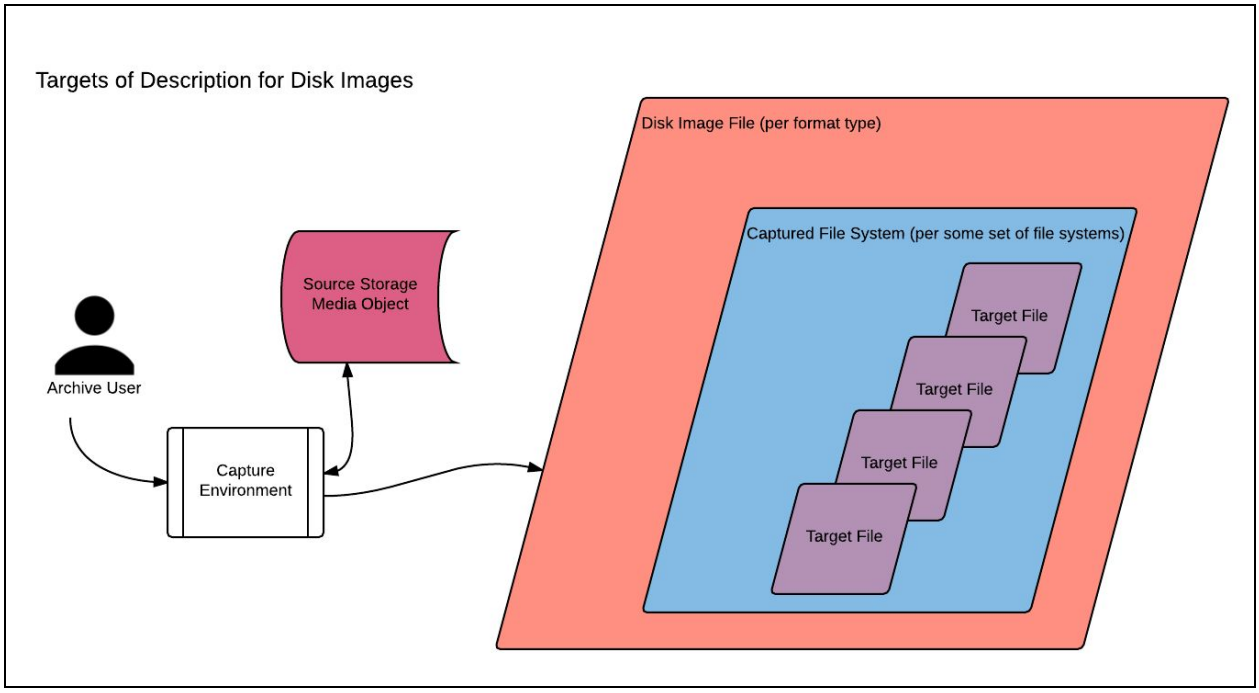

Figure 2. This illustration focuses attention on the targets of description for disk image content.

Metadata Element Recommendations

Based on our assessment of the conceptual landscape of disk imaging, and in the context of describing disk images from an archive and/or library perspective, we make the following recommendations for a core set of metadata elements that combine metadata sources with target objects of description to document the technical features of the disk image object and the file system bitstream, the process history of its creation, and the important characteristics of the source media object.

Disk Image File Object

Disk Image File format : E01
Disk Image File format version : FTK Imager version[7]
Type of Disk Image : Logical; Physical
Disk Image File size: 4 MB
Uncompressed size of imaged content: 3.6 MB
Date created : 2016-03-01
Date modified : 2016-04-01
Existence of compression : uncompressed; zlib

[7] A repository must make sure to store and manage (or retain access to) the file format specification.

File System Bitstream Object

File System format : FAT12[8]
Bytes per Sector : 512
Sector Count : 2880

Capture Event (i.e., Process History Metadata)

Software Used to Create Disk Image File : ADI3.2.0.0
Operating System Used to Create Disk Image File : Windows 7
Date Disk Image File was Captured : 2016-04-01
Disk Image File Creator : Jennifer Weintraub

Source Storage Media Object (i.e., the original physical media)

Original Source Storage Media Type : Floppy disk
Capacity (in bytes) of Original Source Storage Media : 3 MB

These proposed elements are documented in the **Metadata Comparison.file eval** spreadsheet in greater detail. In what follows, we evaluate existing approaches for encoding this type of information in structured schemas in order to understand whether an existing approach can be used by Harvard or whether Harvard will need to develop a customized method to support preservation and technical metadata for disk images.

Survey of Target Schemas for Normalization

Through our discovery process we have been unable to identify existing models for capturing disk image metadata at the varying levels of granularity illustrated in the Figure 1 above, "Metadata Hierarchy in Context of Disk Image Format Acquisition and Preservation." Many tools do some of this (FTK, for example), and sometimes this information is produced manually (as you will see in some of the approaches summarized below). There is also an absence of information about how to store in a structured way information about, for example, the file system on which the image was acquired or information about the actual disk image file itself (such as the data stored in expert witness files [EWF/E01]). Most institutions in the cultural heritage sector that are performing digital forensics (that we identified through our research) appear to use BitCurator to capture disk images. The technical metadata and minimal source

[8] A file system has a format and specification that the repository must make sure to store and manage (or retain access to) over time.

metadata produced about the disk images and the capture process are stored in either DFXML or a sidecar text file.

To get a sense of the approaches taken by others in the community, we contacted Kathleen O'Neal in the manuscripts department at the Library of Congress. They create disk images primarily when they are attempting to recover data from damaged disks. She explained that they use FTK imager and save the text files generated by the application.

Evaluated Candidates

AES standard for audio metadata (AES57)[9]

This standard was adopted in 2011 by the Audio Engineering Society to provide a vocabulary to describe structural and administrative metadata for audio formats to enable preservation. As such, the standard requires elements that specifically relate to digital and analog audio objects. Some required elements have controlled value lists the further limit use beyond the realm of audio (e.g., use/useType). For these reasons, this standard is not appropriate for the description of disk image files or their source storage media.

ContainerMD[10]

Developed by the National Library of France, ContainerMD captures technical metadata about container files. It was developed in 2011, initially to describe ARC (archive) file containers and the file structures stored within them. There are elements in ContainerMD that are currently "empty," but were created to accommodate the extension of the schema to specifically include WARC, ZIP, and TAR files. In addition, an "other" element indicates that ContainerMD could be further extended to include, for example, Disk Image files. The schema could be expanded to include elements that contain the values specified for Disk Image File Objects.

Europeana Data Model - Technical Metadata[11]

This is a Europeana-produced document with suggested technical metadata additions to the Europeana Data Model. The draft/working document was created in February 2015 and the Google spreadsheet was last edited in August 2015. The proposed additions include properties from EBUCore, EXIF, as well as some that are EDM specific. The properties describe the technical attributes of text, still images, 3D images, moving images, and audio files. Technical

[9] Schema available for online review: http://www.aes.org/standards/schemas/aes57-2011-08-27.xsd
[10] Available for online review: http://bibnum.bnf.fr/containerMD-v1/
[11] Available for download here: https://wiki.duraspace.org/download/attachments/68067745/Europeana_TechnicalMetadataProperties_20150217.docx?version=1&modificationDate=1428614201396&api=v2. Available for online viewing here: (also https://docs.google.com/spreadsheets/d/17LjVpdUHsqpLhvQ0kvvDd7ja6eAPhSB9dN3NQ0kEbko/edit#gid=0).

metadata about disk images, or properties that could incorporate data about disk images, are not part of this proposed set.

Portland Common Data Model - Technical Metadata[12]

Following on the work of EDM and EBUCore, a Fedora working group undertook the task of defining a Technical Metadata model for the new Portland Common Data Model which is supporting Fedora 4 and new Hydra implementations. Our evaluation of this technical metadata application profile reveals a similar result to our evaluation of EDM. While the effort looks generically at digital objects stored in a repository, and borrows from EBUCore, PREMIS, and Pronom, the final element set lacks the many important elements identified in the previous section for supporting disk image preservation.

University of Texas Metadata Guidelines for Video (UTVideo)[13]

The UTVideo scheme facilitates the discovery and preservation of, specifically, born-digital video content. The elements are organized into different categories of metadata: descriptive (for discovery), source (physical characteristics of the source object), technical (digital characteristics of each video file), and preservation (actions taken where video object is modified). While the scheme comprehensively captures information about the media type for which it was created, it has a significant number of required fields that are not directly relevant to the set of values being captured for disk image preservation. For example, in the technical metadata category, codec and duration are required; in source, signal format, generation, and duration are required.

BitCurator - PREMIS Preservation Metadata for Disk Image Files

BitCurator generates PREMIS (version 2.1) metadata for each data forensics tool that is applied to a disk image, providing a record of provenance for each stage of processing. Of the PREMIS metadata produced by BitCurator, two process-related properties match the process history metadata fields recommended above: *Operating system used to create file* and *Date acquired*. They are highlighted in the XML snippet below.

```
<event>
  <eventIdentifier>
   <eventIdentifierType>UUID</eventIdentifierType>
   <eventIdentifierValue>cc640440-d73f-11e5-b339-08002707017d</eventIdentifierValue>
  </eventIdentifier>
  <eventType>Capture</eventType>
```

[12] Available for reveiw here: https://wiki.duraspace.org/display/hydra/Technical+Metadata+Application+Profile.

[13] Available for download here: https://www.lib.utexas.edu/schema/Video_Metadata_Guidelines_v1.pdf

```
  <eventDetail>/.../Data_081_08/Data-081-08.E01</eventDetail>
  <eventDateTime>Tue Nov 25 15:56:19 2014</eventDateTime>
  <eventOutcomeInformation>
   <eventOutcome>E01</eventOutcome>
   <eventOutcomeDetail>Version: ADI3.2.0.0, Image size: 387295</eventOutcomeDetail>
  </eventOutcomeInformation>
 </event>
```

One issue with this BitCurator output is that it is not well-formed XML. According to the PREMIS schema, <eventOutcomeDetail> is a container property. So, the value, "Version:..." should be stored in a child property called <eventOutcomeDetailNote>. Additionally (and ideally), a more detailed output would include slightly more process-related information, namely, the operating system used to create the file and the creator of the file. This information could be captured using the PREMIS <agent> property.

Archivematica - METS and PREMIS for Disk Image Files[14]

Archivematica's approach to capturing Disk Image technical and preservation metadata is very similar to the approach recommended and outlined in the previous section. Archivematica relies on the file creator to enter most of the source metadata directly into the Archivematica interface:

- Media Manufacturer
- Serial Number
- Media Format
- Media Density
- Source Filesystem
- Imaging Interface
- Imaging Software

Additionally, Archivematica encourages users to import another set of source metadata fields from the FTK Imager output files (no explanation of the process for how this is achieved). These fields include:

- Imaging Data
- Image Fixity

[14] Description of Archivematica's approach can be found here: https://wiki.archivematica.org/Digital_forensics_image_ingest.

- Source Filesystem
- Accession Data About Extent

Both of these sets of source information are stored in a METS wrapper in the amdSec/sourceMD path. This sourceMD information is connected to each intellectual set of disk images that makes up the whole of an original source media item (as organized in the METS structMap).

Additionally, Archivematica generates PREMIS metadata about ingest processes and stores that data in a METS amdSec/premis path. This PREMIS information is connected to each disk image file that is included in the fileSec of the METS object.

- mets/amdSec1/sourceMD1 - manually entered data
- mets/amdSec1/sourceMD2 - tech md imported from FTK or fiwalk
- mets/amdSec2/premis - event metadata generated during processing
- fileSec - each disk image file linked to appropriate premis amdSecs
- structMap - links disk image file groups to appropriate sourceMD amdSecs (assume that a single source object was imaged and the result was multiple image files (e.g., E01, E02, E03, etc.)

In order to generate a file listing of the files encapsulated within the imaged file system, Archivematica uses fiwalk (either a version with FIDO created by Mark Matienzo, or the version used by BitCurator, which leans on libmagic for format analysis). This extracted XML data is stored in a METS amdSec/sourceMD path and associated with the intellectual set of disk image files that constitute the whole of the original source object as organized in the METS structMap.

If files are to be extracted from the disk image and stored individually, then Archivematica runs Bulk Extractor to generate additional feature metadata about the files. The Bulk Extractor text outputs are stored in bulk as a separate file package. In this case, Archivematica runs FITS to characterize all files once they have been extracted from the disk image. In this case each file is then processed as a unique individual file and related to the disk image object.

DFXML - Filesystem Technical Metadata

The digital forensics XML schema (DFXML) was developed to enable interoperability between different forensics tools, such as The Sleuth Kit, Encase, FTK, AFFLIB, and others. A community effort has emerged around its development in the form of a working group led by the National Institute of Standards and Technology (NIST). The outputs of that group are publicly accessible

on GitHub.[15] A second GitHub repository,[16] run by Simson Garfinkel (a computer scientist at NIST), contains tools and a DTD extension of the core DFXML schema (described later). The DFXML working group GitHub site was last updated in 2014; Garfinkel's site was updated in May 2015.

DFXML can be produced using a number of tools[17] including fiwalk, which was used in this study to produce sample DFXML files. Fiwalk processes disk images using The Sleuth Kit (TSK)[18] library and outputs its results in DFXML. Fiwalk is integrated into the BitCurator environment.

Investigation

Through our investigation, we have uncovered several versions of the DFXML schema, supported by different agents and integrated into different tools. While version control appears to be somewhat deficient, the most recent version, available on the DFXML working group GitHub site, seems to be relatively stable, having last been updated in December 2014.

> **Version 1.0**[19] **–** available from the NIST website. Updated January 30, 2012.
> **Version 1.1.0**[20] – linked from BitCurator wiki. Since BitCurator uses fiwalk/SleuthKit, it is probable that this is an old link that has not been updated since the move to 1.1.1.
> **Version 1.1.1**[21] – available from the DFXML working group GitHub site. This is the latest version of the schema, updated on December 4, 2014. This is the version used by fiwalk/TSK.

The differences between versions 1.1.0 and 1.1.1 are minor, which is indicative that the schema is relatively stable. The majority of the changes relate to the expansion of annotations about the elements. A few elements are deprecated in 1.1.1. These and other changes are delineated in the table below.

Element	Change in 1.1.1
dfxml/creator/build_environment/library	Annotation expanded
dfxml/creator/version	Deprecated
dfxml/fileobject/byte_run/hashdigest:base	Deprecated
dfxml/fileobject/filename	Annotation expanded

[15] https://github.com/dfxml-working-group/dfxml_schema
[16] https://github.com/simsong/dfxml
[17] Tools for generating dxml: http://www.forensicswiki.org/wiki/Category:Digital_Forensics_XML
[18] https://github.com/sleuthkit/sleuthkit
[19] http://www.nsrl.nist.gov/DFXML/fileobject.xsd
[20] https://raw.githubusercontent.com/dfxml-working-group/dfxml_schema/v1.1.0/dfxml.xsd
[21] https://github.com/dfxml-working-group/dfxml_schema/blob/master/dfxml.xsd

dfxml/fileobject/gid	Type value changed (from string to nonNegativeInteger)
dfxml/fileobject/id	Annotation expanded
dfxml/fileobject/parent_object	Annotation expanded
dfxml/fileobject/uid	Annotation expanded
dfxml/metadata	Expands options for namespaces beyond Dublin Core
dfxml/volume/error	New element in 1.1.1

Findings

We encountered a DTD file in our investigation (dfxml.dtd) that contains elements that do not appear in any of the schema documents (see below). The DTD can be found on Simson Garfinkel's GitHub[22] site and in the "Additional Tools > DFXML Scripts" tab in BitCurator. To our knowledge, the DTD is not used in the creation of DFXML files on BitCurator or via fiwalk/SleuthKit, but more investigation is necessary.

- dfxml/source/imagefile
- dfxml/source/sectorsize
- dfxml/source/device_model
- dfxml/source/device_sn
- dfxml/source/acquisition_commandline
- dfxml/source/acquisition_device
- dfxml/source/device_capabilities
- dfxml/source/devicesectors
- dfxml/source/acquisition_macaddr
- dfxml/source/acquisition_date

The elements in the DTD are interesting—in some cases they closely map to the metadata we identified as valuable in our evaluation of the EWF, RAW, ISO, and BIN files. This should be noted, in particular, because very few of the elements in DFXML 1.1.1 align to those metadata values. Our test of BitCurator and fiwalk confirmed that the source elements noted above are not part of the set of elements that output in the XML files that they produce. Correspondence with NIST representatives confirmed that DFXML does not have a scope that covers the source media object. DFXML is of great use for documenting the files and directories and geometric information structured within a disk image file, but has minimal use (as it stands now) for

[22] https://github.com/simsong/dfxml/blob/master/dtd/dfxml.dtd

documenting both the characteristics of the disk image file format itself and the original media object that was imaged in the creation of the disk image file.

Recommended Modeling in DRS Context

ContainerMD already supports expression of the majority of the recommended metadata elements, and ContainerMD was developed from the conceptual model of a single serialized byte sequence that itself contains structured information, whether that be file systems, files, or directories. Additionally, ContainerMD was designed to support containerExtensions for particular types of containers, e.g., ARC, WARC, TAR, ZIP. These extensions allow for the articulation of information elements characteristic of a given format. Because of the current absence of any disk image specific metadata schema that sufficiently describes disk image file formats, source media objects, file systems, and encapsulated files, we recommend an approach that takes advantage of ContainerMD's existing strengths and coordinates with the National Library of France (BNF) to design and approve a containerExtension for ContainerMD that is specific to disk images. Below is an example of the proposed containerExtension for disk images:

```
<containerMD:containerExtension>
      <containerMD:DiskImageContainer>
            <containerMD:diskImageType>physical</containerMD:diskImageType>
            <containerMD:sourceMediaType mediaManufacturer="3M"
            mediaSerialNumber="23542" storageCapacity="4200000">3.5-inch floppy
            disk</containerMD:sourceMediaType>
            <containerMD:fileSystems>
                  <containerMD:fileSystem bytesPerSector="2880"
                  sectorCount="512">FAT12</containerMD:fileSystem>
            </containerMD:fileSystems>
      </containerMD:DiskImageContainer>
</containerMD:containerExtension>
```

The proposed containerExtension accounts for the information elements related to disk images that are not already accounted for in the ContainerMD schema. A full example of a disk image described using ContainerMD (with the proposed containerExtension) is included as Appendix B here: https://drive.google.com/file/d/0B-FBppbZcJ2EQUV4aUFlYmZpcnc/view?usp=sharing.

One benefit of this approach is that ContainerMD natively supports verbose and/or non-verbose file summaries. In the example at the link above, we have included a non-verbose example of the currently supported file summary option in ContainerMD. This approach would allow HUL to insert an entire instance of ContainerMD within a DRS Object Descriptor using the existing PREMIS object characteristics extension mdWrap option.