CRATE: A Simple Model for Self-Describing Web Resources

Joan A. Smith and Michael L. Nelson
Old Dominion University, Department of Computer Science
Norfolk, VA 23529 USA
{jsmit, mln}@cs.odu.edu

ABSTRACT

If not for the Internet Archive's efforts to store periodic snapshots of the web, many sites would not have any preservation prospects at all. The barrier to entry is too high for everyday web sites, which may have skilled webmasters managing them, but which lack skilled archivists to preserve them. Digital preservation is not easy. One problem is the complexity of preservation models, which have specific metadata and structural requirements. Another problem is the time and effort it takes to properly prepare digital resources for preservation in the chosen model. In this paper, we propose a simple preservation model called a CRATE, a complex object consisting of *undifferentiated metadata* and the resource byte stream. We describe the CRATE complex object and compare it with other complex-object models. Our target is the everyday, personal, departmental, or community web site where a long-term preservation strategy does not yet exist.

Categories and Subject Descriptors:H.3.5Information Storage and Retrieval Online Information Services [Web-based services]

General Terms Design, Documentation, Experimentation

Keywords Digital preservation, OAI-PMH, mod_oai

1. INTRODUCTION

Digital preservation is not easy. The OAIS Reference Model, which informs many modern digital preservation systems, specifies submission, archival and dissemination information packages (SIP, AIP, and DIP) along with the associated roles of producer, management, and consumer [7]. The management or archivist role is necessarily complex, with responsibilities that cover all aspects of digital curation. Management is responsible for ensuring that an AIP has all necessary information from the original SIP so that it is "Independently Understandable" [7]. To achieve this goal, a wide variety of object metadata is collected and organized in a repository-specific model. As the CCSDS notes in its report, defining and populating metadata elements requires expertise. Yet there are not enough experts available to prepare the billions of web pages for repository ingestion. Can we simplify this model and thereby increase preservation participation?

A very simple approach has been taken by the Internet Archive (IA), which attempts to capture periodic snapshots of the web. If not for the IA's efforts, many sites would not have any preservation prospects at all. The only metadata that IA retains for a given digital resource is that which was obtained through the web crawling request-response sequence of HTTP [1]. But this information is often insufficient. Sites vary in the type and organization of resources they serve, generally relying on the MIME specification to sort it all out at the receiver's end. MIME types can be insufficient and even incorrect, or the MIME type itself may not be adequate to describe the content [4, 25]. If this resource poses problems to today's web client, how will a future information archaeologist interpret it? It appears that the OAIS requirement for independent understanding of the resource has not been met because of insufficient resource metadata. The *simplest possible model* requires more metadata than is retrieved by conventional crawling and MIME typing alone.

After examining more than one billion web pages, Google reported in December 2005 that metadata descriptions have not been widely adopted by web site authors [2]. In the few site pages containing META tags, Google found that they were frequently misused, with markup errors such as inverted `CONTENT` and `VALUE` elements, or missing the quotation marks that delimit attributes from tags. Google noted that such errors rendered the META information useless from a crawler's perspective. Examining the HTML source, we found that Google itself did not include any labelled metadata except `DOCTYPE` and `TITLE` in the report. Even so, search engine oriented metadata alone would not be enough to maintain a resource's viability through technological migration over the decades to come. Structural, behavioral and administrative metadata is needed as well as simple descriptive information. This type of metadata is not the target of search engines, whose focus is on web resources that are *available now.*

The lack of metadata on the web is understandable. Studies of digital libraries have shown that metadata creation requires significant professional effort and review to achieve consistent, valid information [10]. For some resource types, a JPEG image for example, the requisite technical metadata can only be derived by using analytical tools. What incen-

IWAW'07, June 23, 2007, Vancouver, British Columbia, Canada.

tive is there for the average webmaster to go to such lengths? In 1998, Thomas and Griffin speculated that profit potential and general self-interest in search engine (SE) rankings would produce a wealth of local site metadata [27]. So far, this has not happened. Some sites have populated their pages with inaccurate key words to boost their ranking in search results [5], but this technique is less effective as SEs become more sophisticated and can distinguish between valid and misleading metadata. Google, for example, does not rely on a site's internal metadata to determine results rankings. Ironically, the success of Google's PageRank formula may have further reduced the incentive for sites to provide even minimal descriptive information. There is no search engine-related incentive to provide preservation metadata.

The archivist is less concerned with search engine, semantic web, or other here-and-now metadata. Google may not care about the color index of a web image, but such information *does* matter for preserving that image. For archivists, the purpose of metadata is to enable and to support long-term persistence of the resource through the decades that come, despite the inevitable technology changes that will occur in the future. Nonetheless, archival services which seek to preserve everyday websites are most likely to access such sites through web crawling, as opposed to formal submission of the site by the webmaster to the archive. At Internet Archive, for example, the only requirement is to enter a URL on the submission page. IA promises to crawl the site within 48 hours of the request. It does not ask for nor expect any special set of metadata to accompany the site's contents.

If web crawling does not produce sufficient preservation metadata, what can be done to improve the preservation prospects for everyday sites where archivist expertise is lacking? In contrast with models which have a complex ontology rich in metadata, we propose a simple model of *undifferentiated* metadata, automatically generated and serialized with the original resource byte stream. This model overcomes the difficulties inherent in metadata categorization, requires virtually no additional effort or cost by webmasters, and lowers the barrier to preservation practices for everyday websites by enabling resources to be *self-describing*.

2. RELATED WORK

A survey conducted by OCLC/RLG of digital preservation practices at professionally administered archives noted the lack of a common metadata vocabulary as well as a wide variation in repository implementation models [19]. Repositories may design their own implementations or choose from a number of existing solutions and services. To address these issues, the PREMIS Working Group produced a preservation metadata schema [20]. It clears up the relationship between an "intellectual entity" and its constituent digital object(s), and between a single digital object and its various representational instances. The PREMIS model still allows for a flexible, repository-specific implementation while standardizing the definition of entities, objects, rights, events, and agents.

Curated repositories have a number of service and software solutions available to them. Fedora [21] integrates resources, relationships, and metadata, and provides a service framework that includes preservation monitoring and support for the Open Archives Initiative. DSpace [26] is another Open Source preservation-oriented project for institutional repository management. Written in Java, it currently supports Dublin Core metadata internally but will export information in a simple XML format. Fedora, DSpace, and other managed collections can subscribe to the PANIC system [11], which facilitates awareness between preservation partners on issues relating to technical obsolescence. It utilizes semantic web services to warn participating repositories about file formats which are obsolete, and to provide options for converting the items to a newer format. Installation and maintenance of PANIC, Fedora and DSpace require a significant level of skill and commitment by the administrator.

Governments worldwide have invested considerable effort and funding in the search for digital preservation solutions. The VERS project designed a trustworthy, encapsulated object (VEO) specifically to support the long-term preservation of digital government records [30]. A VEO is a complex object which contains the resource (government record) and all relevant metadata, together with one or more digital signatures. If a record changes, the VEO is updated with the new information. In addition, a VEO may contain multiple resources and records which are closely related to one another, such as a series of land surveys. A VERS Toolkit is available [29], but it expects additional components such as entity-relationship information, which make it impractical as a solution for ordinary websites. Digital signatures also add a significant level of effort to the VERS preservation approach. Despite persuasive arguments in favor of trustworthy digital objects (TDOs) [9], the infrastructure to support them does not yet exist.

Another complex-object model implemented by some digital libraries is the MPEG-21 DIDL standard, initially defined for entertainment industries to package and distribute multimedia content. An MPEG-21 DID can contain multiple representations of a single resource, for example, a video in both AVI and WMV formats, along with specific information like copyright data and soundtracks in various languages. The flexible, complex-object nature of the MPEG-21 standard has proven to be a workable format for a number of archiving projects. Old Dominion University (ODU) demonstrated the practicality of using the MPEG-21 DIDL format for preservation during the Archive Ingest and Handling Test research project [16]. MPEG-21 DIDL has also been implemented by the LANL Research Library community to represent the library's assets as complex objects consisting of the resource (text, multimedia, etc.) packaged with relevant metadata [6, 28].

METS, the Metadata Encoding and Transmission Standard, also defines an XML-formatted complex-object information package containing the resource, administrative and descriptive metadata, and other key elements [14]. The high-level METS object ("METS document") is described by a profile which lays out the semantics for that class of document. The embedded-semantics complicates METS portability, since repositories wishing to use METS must conform to a particular profile. In contrast, the high-level MPEG-21 object ("container") is an abstract data model that does not have specific semantics encoded in its declaration language, and repository contents can be expressed in a number of different ways. For example, LANL researchers created the concept of an XML tape archive [13], which is conceptually similar to the sequential storage of items on a physical tape system, and expressed the content using MPEG-21 DIDL.

Among the advantages shared by METS and MPEG-21 approaches are an XML-document format which is essentially human-readable; a complex-object approach to packaging a resource with its metadata; the ability to aggregate multiple resources together; and compatibility with HTTP. The models also share a disadvantage: a complexity of implementation that poses a challenge for widespread adoption at everyday websites, such as having specific requirements regarding the metadata that accompanies each resource. For each of these implementations, mapping information into the proper metadata category is a complex task. When it comes to preserving everyday websites, the phrase "complex object" should apply to the packaged resource rather than to the difficulty in creating it.

3. A QUALITATIVE COMPARISON OF PRESERVATION MODELS

Resource preservation requires not only the resource itself but also sufficient forensic metadata [22]. To this end, many repositories use a complex object model to package resources with related metadata. Virtually any kind of metadata can be incorporated, from executable code that affects resource behavior (example: javascript), to straightforward, descriptive "field=value" elements (e.g., Dublin Core metadata). High-level diagrams of four commonly-used models (ARC, VEO, LANL MPEG-21 and METS) can be seen in Figures 1, 5, and 6. Each of these organizes the constituent datastreams (resources) and metadata differently. For example, LANL MPEG-21 resource containers can grow in breadth (have more metadata) but not depth [6]; ARC file records are static, but may differ in breadth (amount of metadata recorded during that crawl). The minimum information set also differs for each of these models. For example, only METS requires a structural map; only VEO requires one or more digital signatures. These differences are a reflection of differences in the ontologies of the implementing repositories. The following sections look at the process of ingesting resources into each of these models. We use a hypothetical web site consisting of a single HTML page with some English-language content ("index.html") and a JPEG image ("Barfoo.jpg"), and a link to a PDF ("Foo.pdf").

Robots gather metadata during the process of crawling a site. Some information is returned from the TCP/IP connection: the IP address of the responding server, for example. Other information is part of the HTTP protocol. HTTP has over 30 header fields which may be utilized in the request, the response, or both. Most web servers will provide several header fields in the response to a request, including "Last-Modified" and "Content-Type." Fields in the response are determined in part by the requesting server and in part by the responding server; some servers may not support all header fields. Table 1 shows the information extracted from the HTTP request-response sequence for each of the example resources. Some HTTP fields are not in the table because they are not provided by the responding server (Content-MD5 for example) or because they are not applicable to this set of request-response events ("Content-Range" and "Expires," for example).

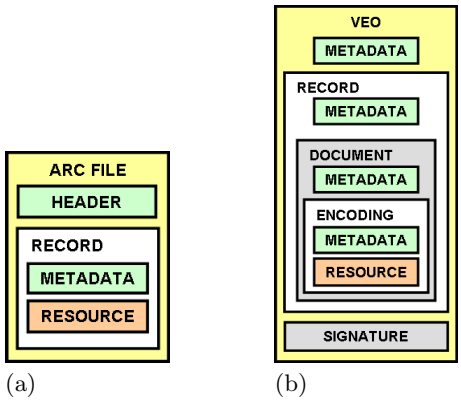


Figure 1: Complex object model representations (a)ARC and (b)VEO

3.1 ARC and WARC

The Internet Archive stores crawled sites in a file format called "ARC" [13], shown in Figure 1(a). Except for the protocol headers, web crawling using HTTP, FTP, and NNTP typically generates little or no explicit descriptive metadata. Our sample web site merely needs to be crawled by the Alexa robot for the ARC file to be created, or we could self-crawl using Heritrix [15].

This approach does not provide much in the way of future forensic information, so the Internet Archive also offers an expanded preservation-oriented crawling service, "Archive-It". The service is on a fee-based subscription, and allows the subscribing site to provide Dublin Core metadata, multiple "seed" URLs, varying schedules for each seed, and other archiving details. For our sample site, we would need to manually introduce the Dublin Core information for each resource, via the Archive-It catalog form. Even though this is an improvement forensically over plain HTTP metadata, expressing technical information in these fields is awkward, at best. Consider the Jhove analysis of our JPEG resource, shown in Figure 2. What parts of the analysis should be entered into Dublin Core fields? What kind of consequences arise from discrepancies in the output from other utilities if we do choose to include some or all of the information?

ARC files are plain ASCII text, and any characters outside that range must be "escaped". Whether we use the expanded, Dublin Core-based version or the original, the archived file conforms to the ARC format: file header with version information followed by the URL record which begins with a list of metadata fields included with this particular record, and ends with the actual content returned from the HTTP method (e.g., GET). ARC files are compressed at both the URL-record level and at the file level, for improved storage. Although not written in XML, an ARC file is mostly human-readable, once uncompressed, as shown in Figure 3.

The International Internet Preservation Consortium (IIPC) has developed an extended revision of the ARC format called "WARC" (for "Web ARChive") which lets harvesting organizations aggregate large amounts of web resources into specific collections with locally-assigned metadata such as "subject" or unique record ID. The proposed WARC format has numerous sections to clearly delineate "records" in the file. A record, in WARC terms, can be the "response," the "request", file structure ("warcinfo"), or other descriptive information. Like other managed collection models, WARC expects the repository to provide any metadata outside of

HTTP FIELD	File (or Resource)		
	PDF	JPEG	HTML
Host	www.foo-bar.com		
IP Address	128.82.7.123		
Result Code	200		
Resource	/var/www/Foo.pdf	/var/www/images/Barfoo.jpg	/var/www/index.html
Response Date	25-Mar-2007 17:28:53 GMT	25-Mar-2007 17:36:06 GMT	25-Mar-2007 17:16:00 GMT
Server	Apache/1.3.33 (Unix)		
Last-Modified	11-Mar-2007 06:28:49 GMT	03-Feb-2007 01:22:23 GMT	01-Jan-2007 09:02:09 GMT
ETag	49ec239-36eab7-45ca9e15	580014e-bec5-45ca98f1	311ff31-9100-45ca9801
Content-Length	3599031	48837	2136
Content-Type	application/pdf	image/jpeg	text/html charset=iso-8859-1
Content	(binary data)		(html content)
Content-Language	(not applicable)		en

Table 1: Metadata available from the HTTP (crawl) request/response sequence

```
<?xml version="1.0" encoding="UTF-8"?>
<jhove xmlns:xsi="http://www.w3.org/2001
  /XMLSchema-instance"
  xmlns="http://hul.harvard.edu/ois
  /xml/ns/jhove"
  xsi:schemaLocation=
  "http://hul.harvard.edu/ois/xml/ns/jhove
  http://hul.harvard.edu/ois/xml/xsd/jhove
  /1.4/jhove.xsd"
  name="Jhove" release="1.1"
  date="2006-06-05">
<date>2007-04-19T12:20:23-04:00</date>
<repInfo uri="/var/www/Barfoo.jpeg">
<reportingModule release="1.2"
  date="2005-08-22">JPEG-hul
</reportingModule>
<lastModified> 2007-02-03T18:22:23-05:00
</lastModified>
<size>25474</size>
<format>JPEG</format>
<version>1.01</version>
<status>Well-Formed and valid</status>
<sigMatch>
<module>JPEG-hul</module>
</sigMatch>
<mimeType>image/jpeg</mimeType>
<profiles>
<profile>JFIF</profile>
</profiles>
<properties>
<property>
<name>JPEGMetadata</name>
<values arity="List" type="Property">
<property>
<name>CompressionType</name>
<values arity="Scalar" type="String">
<value>Huffman coding,
Baseline DCT</value>
</values>
</property>
<property>
<name>Images</name>
<values arity="List" type="Property">
<property>
<name>Number</name>
<values arity="Scalar" type="Integer">
<value>1</value>
</values>
</property>
<property>
<name>Image</name>
<values arity="List" type="Property">
<property>
<name>NisoImageMetadata</name>
  <values arity="Scalar"
  type="NISOImageMetadata">
  <value>
  <mix:mix xmlns:mix=
    "http://www.loc.gov/mix/"
    xmlns:xsi="http://www.w3.org/2001
    /XMLSchema-instance"
  xsi:schemaLocation=
    "http://www.loc.gov/mix/
  http://www.loc.gov/mix/mix.xsd">
  <mix:BasicImageParameters>
  <mix:Format>
  <mix:MIMEType>image/jpeg</mix:MIMEType>
  <mix:ByteOrder>big-endian</mix:ByteOrder>
  <mix:Compression>
  <mix:CompressionScheme>6
  </mix:CompressionScheme>
  </mix:Compression>
  <mix:PhotometricInterpretation>
  <mix:ColorSpace>6</mix:ColorSpace>
  </mix:PhotometricInterpretation>
  </mix:Format>
  </mix:BasicImageParameters>
  <mix:ImageCreation> </mix:ImageCreation>
  <mix:ImagingPerformanceAssessment>
  <mix:SpatialMetrics>
  <mix:SamplingFrequencyUnit>3
  </mix:SamplingFrequencyUnit>
  <mix:XSamplingFrequency>0
  </mix:XSamplingFrequency>
  <mix:YSamplingFrequency>0
  </mix:YSamplingFrequency>
  <mix:ImageWidth>459</mix:ImageWidth>
  <mix:ImageLength>253</mix:ImageLength>
  </mix:SpatialMetrics>boxedminipage
  <mix:Energetics>
  <mix:BitsPerSample>
    8,8,8</mix:BitsPerSample>
  <mix:SamplesPerPixel>
    3</mix:SamplesPerPixel>
  </mix:Energetics>
  </mix:ImagingPerformanceAssessment>
  </mix:mix>
  </value>
  </values>
  </property>
  <property>
  <name>Scans</name>
  <values arity="Scalar" type="Integer">
  <value>1</value>
  </values>
  </property>
  <property>
  <name>QuantizationTables</name>
  <values arity="List" type="Property">
  <property>
  <name>QuantizationTable</name>
  <values arity="Array" type="Property">
  <property>
  <name>Precision</name>
  <values arity="Scalar" type="String">
  <value>8-bit</value>
  </values>
  </property>
  <property>
  <name>DestinationIdentifier</name>
  <values arity="Scalar" type="Integer">
  <value>0</value>
  </values>
  </property>
  </values>
  </property>
  <property>
  <name>QuantizationTable</name>
  <values arity="Array" type="Property">
  <property>
  <name>Precision</name>
  <values arity="Scalar" type="String">
  <value>8-bit</value>
  </values>
  </property>
  <property>
  <name>DestinationIdentifier</name>
  <values arity="Scalar" type="Integer">
  <value>1</value>
  </values>
  </property>
  </values>
  </property>
  </values>
  </property>
  </values>
  </property>
  </values>
  </property>
  <property>
  <name>ApplicationSegments</name>
  <values arity="List" type="String">
  <value>APP0</value>
  </values>
  </property>
  </values>
  </property>
  </properties>
  </repInfo>
  </jhove>
```

Figure 2: Metadata derived from Barfoo.jpeg using Jhove's JPEG-HUL module. Command line entry: /opt/jhove/jhove -c /opt/jhove/conf/jhove.conf -m jpeg-hul -h xml Barfoo.jpeg

```
filedesc://IA-001102.arc 0 19960923142103 text/plain 76
1 0 Alexa Internet
URL IP-address Archive-date Content-type Archive-length

http://www.dryswamp.edu:80/index.html 127.10.100.2 19961104142103 text/html 202
HTTP/1.0 200 Document follows
Date: Mon, 04 Nov 1996 14:21:06 GMT
Server: NCSA/1.4.1
Content-type: text/html Last-modified: Sat,10 Aug 1996 22:33:11 GMT
Content-length: 30
<HTML>
Hello World!!!
</HTML>
```

Figure 3: ARC file example data, from http://www.archive.org/web/researcher/ArcFileFormat.php

the HTTP request-response event information. This can be a challenge for the average web master.

3.2 VEO

The VERS system's focus on evidentiary-quality digital archives requires a great deal more metadata than is currently provided by our sample web site. The most important element, the digital signature, poses a problem in that the required PKI infrastructure is not available through our web hosting service, and we do not have a public key on record for this site. This is a common situation for everyday websites. We could perhaps substitute the MD5 message integrity check field (HTTP's Content-MD5 header) but this is clearly not the same level of assurance called for by VERS. In addition, the MD5-Digest directive in Apache defaults to "off," so it must be specifically enabled at the server.

Although it is possible, digitally-signing HTML documents is not a prevalent practice. JPEG images occasionally have embedded copyright information, but encrypted or digitally signed images are relatively rare. PDF documents, on the other hand, often have password-level protection or encryption. Adobe Distiller can digitally sign a PDF, using a certificate generated by Adobe or by the user, but this does not seem to be used very often on everyday websites. Metadata gathered using the Acroread utility with Foo.pdf (Figure 4) shows us that our Foo.pdf document is neither signed nor encrypted, although we could amend the document and provide a digital signature as part of the process of creating a VERS Encapsulated Object. In practice, the digital signature is not applied to the resource but to the VEO itself, and is designed to ensure resource fixity. The lack of encryption of our resources is fortunate in this case, since VERS specifies that a VEO may not contain encrypted data [30].

To create the VEO, we need both descriptive and technical metadata for our three resources. In addition, we need to decide how we will structure our VEO: as a single object which wraps the site's three resources, or as multiple VEOs, in which case we need to decide if the HTML-resource VEO should also contain the JPEG resource. For simplicity's sake, we will consider our 3-resource web site as a single VERS encapsulated object. A VEO requirement is that resources be included *By-Value*, not By-Reference. As Figure 1(b) shows, metadata exists at each of the VEO levels: Object, Record, Document, and Encoding. We noted in § 3.1 that little metadata is produced from the HTTP GET, so we will have to process each of our resources for the required VEO information. For example, record-level descriptive metadata for a VEO are based on Dublin Core fields, which in this example will require manual intervention for record completeness. The technical metadata usually found in a VEO, i.e., specifications about the JPEG and the PDF, can be determined by running an analysis utility like Jhove. This is one exception to the By-Value requirement of VERS, since the metadata element can point to the technical specification for that version of JPEG or PDF instead of embedding the specification in place. The final VEO is a self-documenting, human-readable XML file with one record and three "documents", each of which is represented in its original encoding by a Base64-conversion of resource content. The "only" missing element is the digital signature which, although central to the VEO model, is not easily generated.

3.3 METS & PREMIS

The METS model is at the heart of many repository systems, including DSpace and Fedora. The primary object, a METS "document," contains seven major sections (Figure 5(a)), but only the File and the Structural Map sections are required:

Section	Element	Examples
File	Location	Source Path Target Path Source URL
	Content	By-Value (Base64) By-Reference (File Ptr/URI)
Structural Map	Div	Sitemap Links In/Links Out

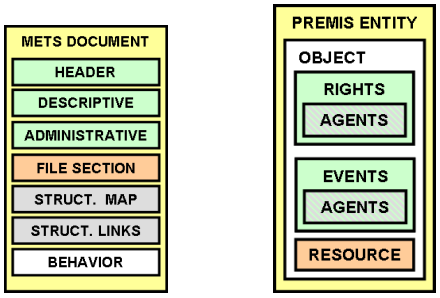
The crawling process can provide reasonable data for the File and Map sections, but for preservation purposes we should record much more information, particularly in the Descriptive and Administrative sections for which HTTP and our crawl contribute little or no data. Some metadata schemas have been endorsed by METS. For the Descriptive section, Dublin Core and MARC are recommended. For the Administrative section, recommended schemas include NYU's Schema for Technical Metadata for Text, as well as the NISO Technical Metadata for Digital Still Images. Like

```
<?adobe-xap-filters esc="CR"?>
<x:xmpmeta xmlns:x='adobe:ns:meta/'
  x:xmptk='XMP toolkit 2.9.1-13, framework 1.6'>
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:iX='http://ns.adobe.com/iX/1.0/'>
 <rdf:Description rdf:about='uuid:d46586fa-403c-4c1
  <?adobe-xap-filters esc="CR"?>
<x:xmpmeta xmlns:x='adobe:ns:meta/'
  x:xmptk='XMP toolkit 2.9.1-13, framework 1.6'>
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:iX='http://ns.adobe.com/iX/1.0/'>
 <rdf:Description rdf:about
  ='uuid:d46586fa-403c-4c1c-9713-43b5a2f3f649'
  xmlns:pdf='http://ns.adobe.com/pdf/1.3/'>
  <pdf:Producer>ESP Ghostscript 815.02</pdf:Producer>
 </rdf:Description>
 <rdf:Description rdf:about
  ='uuid:d46586fa-403c-4c1c-9713-43b5a2f3f649'
  xmlns:xap='http://ns.adobe.com/xap/1.0/'>
  <xap:ModifyDate>2007-03-14T10:00:21Z</xap:ModifyDate>
  <xap:CreateDate>2007-03-14T10:00:21Z</xap:CreateDate>
  <xap:CreatorTool>dvips(k) 5.95a Copyright 2005
    Radical Eye Software</xap:CreatorTool>
 </rdf:Description>
 <rdf:Description rdf:about
  ='uuid:d46586fa-403c-4c1c-9713-43b5a2f3f649'
  xmlns:xapMM='http://ns.adobe.com/xap/1.0/mm/'>
  <xapMM:DocumentID>uuid:ada536f0-811e-487d-b20a-23ebcfe106b7
    </xapMM:DocumentID>
 </rdf:Description>
 <rdf:Description rdf:about
  ='uuid:d46586fa-403c-4c1c-9713-43b5a2f3f649'
  xmlns:dc='http://purl.org/dc/elements/1.1/'>
  <dc:format>application/pdf</dc:format>
  <dc:title>
   <rdf:Alt>
    <rdf:li xml:lang='x-default'>jcdl07.dvi</rdf:li>
   </rdf:Alt>
  </dc:title>
 </rdf:Description>
</rdf:RDF>
</x:xmpmeta>c-9713-43b5a2f3f649'
  xmlns:pdf='http://ns.adobe.com/pdf/1.3/'>
  <pdf:Producer>ESP Ghostscript 815.02</pdf:Producer>
 </rdf:Description>
 <rdf:Description rdf:about
  ='uuid:d46586fa-403c-4c1c-9713-43b5a2f3f649'
  xmlns:xap='http://ns.adobe.com/xap/1.0/'>
  <xap:ModifyDate>2007-03-14T10:00:21Z</xap:ModifyDate>
  <xap:CreateDate>2007-03-14T10:00:21Z</xap:CreateDate>
  <xap:CreatorTool>dvips(k) 5.95a Copyright 2005
    Radical Eye Software</xap:CreatorTool>
 </rdf:Description>
 <rdf:Description rdf:about
  ='uuid:d46586fa-403c-4c1c-9713-43b5a2f3f649'
  xmlns:xapMM='http://ns.adobe.com/xap/1.0/mm/'>
  <xapMM:DocumentID>uuid:ada536f0-811e-487d-b20a-23ebcfe106b7
    </xapMM:DocumentID>
 </rdf:Description>width
 <rdf:Description rdf:about
  ='uuid:d46586fa-403c-4c1c-9713-43b5a2f3f649'
  xmlns:dc='http://purl.org/dc/elements/1.1/'>
  <dc:format>application/pdf</dc:format>
  <dc:title>
   <rdf:Alt>
    <rdf:li xml:lang='x-default'>draftFoo.dvi</rdf:li>
   </rdf:Alt>
  </dc:title>
 </rdf:Description>
</rdf:RDF>
</x:xmpmeta>
```

Figure 4: Metadata derived from Foo.pdf using Acroread.

the VERS ingestion process, the resources could be analyzed using Jhove and/or Acroread to extract data for the Technical portion of the Administrative section. Descriptive metadata is still a problem because neither Dublin Core nor MARC metadata can be derived for our sample resources.

Repositories customize METS via a *profile* which manages the types of resources it contains. An image collection can have one set of metadata specifications, while audio CD collections have another. Our site has 3 very different types of resources: HTML, PDF, and JPEG. Using the default profile (from the Library of Congress tutorial web site, for example) we would probably need to create three separate METS documents, one for each resource. Alternatively, we could adopt the PREMIS extensions to METS, which is more suited to our sample site. In PREMIS, our web site could be mapped to an "Intellectual Entity" with each of the resources comprising an "Object" contained within that entity. Figure 5(b) gives a conceptual view of the PREMIS Entity. On the other hand, our PDF can be considered a complete Intellectual Entity of its own, and we could therefore archive it as a separate object. In this case, the PDF would have a *relationship* to the HTML referral page. Like many Archival Information Packages, the METS AIP is an XML file where content may be included either By-Value or By-Reference. The structure of the AIP follows the repository's METS profile. In any case, mapping the site's resources to one or more "documents" or to one or more "entities" will depend on the particular implementation at the archiving repository. Two agencies archiving our site could adopt very different strategies and yet adhere to the METS



(a) (b)

Figure 5: (a)METS Document Object and (b)PREMIS Intellectual Entity.

model. Our experience on the AIHT Project showed us how complicated ingestion can be when two sources implement a model like METS in different ways [16]. The PREMIS data dictionary addresses this issue by providing more detailed guidelines for metadata fields and content. This is a boon to the knowledgeable archivist, but a daunting set of criteria for the typical webmaster.

3.4 MPEG-21 DID

LANL has successfully adopted the flexible MPEG-21 DIDL model for use in digital repositories. Figure 6 shows how a Technical Report is stored in the MPEG-21 format at LANL. The main MPEG-21 object, called a "container," can have multiple nested containers, items, and components.

"Descriptors" accompany each of these elements to provide information such as origin, date, and element content-type - i.e., metadata about the metadata. Although the original

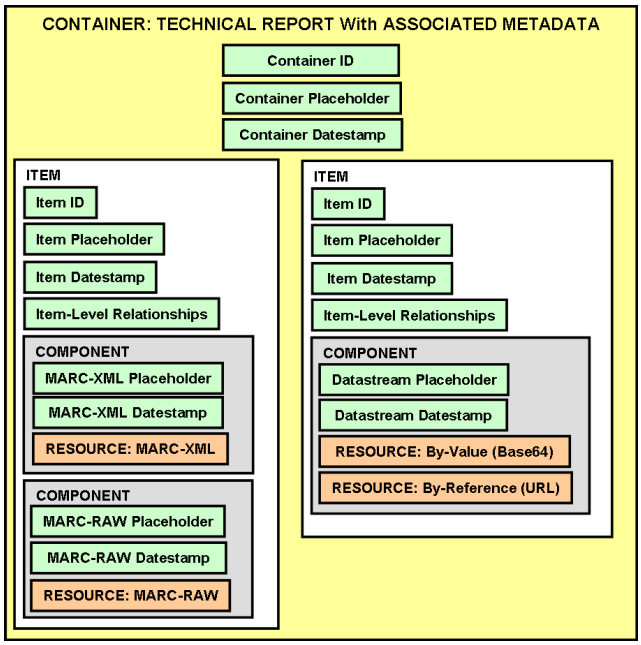


Figure 6: LANL MPEG-21 DID

industry specification permitted deep nesting of containers and objects, LANL's implementation only allows a container to grow in breadth, not depth. This approach simplifies resource access, update, and general management.

Like many other XML-based complex-object models, metadata and resources may be included either By-Reference or By-Value. If we have additional information about a resource, it can be included within the container as an additional *item*. For example, more detailed information about file Foo.pdf, including metadata about its embedded images, is produced by the Jhove PDF-HUL module (see Figure 7). The Jhove metadata would be contained within one *item-component* in the container, and the Acroread information (Figure 4) would be contained within another *item-component*. A third *item-component* could hold the complete set of response-request fields obtained from the HTTP sequence (Table 1).

LANL's use of MPEG-21 exhibits a relatively simple ontology. Harvesting our web site would produce three containers, one per resource. The number of items in each container would vary with the number of metadata source-types. If no utilities were used, only the HTTP metadata item would exist. Otherwise, one item per metadata type would be included in the container. The final item in each container is the resource itself, which may be included By-Reference or By-Value, or both.

3.5 Models, Metadata, and Interoperability

In 2005, as part of the Archive Ingest and Handling Test (AIHT), the Library of Congress tested "the feasibility of transferring digital archives in toto from one institution to another" [23]. Several issues arising during the test, and conclusions resulting from it, have influenced the development of this proposal. The first is that metadata which is characterized as *required* for resource ingestion often turns out, instead, to merely be *desired*. Some resources are valuable enough to warrant ingestion with whatever metadata is available for them, even if it does not fulfill repository "requirements." Another observation from the AIHT is that metadata markup, like ontologies, will never evolve into a universally-accepted approach [24]. Two repositories storing the same resource may record and map metadata very differently. This means that interoperability or even simple resource exchange between the repositories may involve very complex operations, even if both used, say, METS. As a result, a key conclusion of the test is that data-centric strategies are more useful than those based on implementing a particular environment or model.

4. CRATE: A SIMPLE MODEL FOR SELF-DESCRIBING WEB RESOURCES

An insightful comment by Stewart Brand is that we need data to be "born archival", not just digital [24]. As an interactive medium, the WWW is purely digital with nearly no archival-quality information. A mattress sold in the U.S. comes with more metadata than is available for the typical web page. Institutions attempting to record our digital web heritage, like the Internet Archive and the European Archive, can merely store and refresh the bits, trusting descriptors like ".pdf" to be accurate. Analyzing even a portion of the resources for confirmation of type or for more informative metadata is impractical.

On the other hand, it *is* practical and feasible for the web server to provide a variety of supporting metadata together with the resource. We have demonstrated this concept in [25], where the web server itself analyzes the resource *at time of dissemination* and includes both the resource and the analysis within the response. The routine transfer of complex objects like MPEG-21 DIDs over HTTP further supports our contention that web crawls can be used to acquire both resources and forensic metadata [16, 28]. The resource may not be *born* archival, but its adoptive parents are naturally archival.

The resulting complex object can be molded into a repository specific model, such as those discussed in § 3. Where models like METS would organize the metadata according to a profile, and LANL DIDs would organize it by container-items, we suggest a simpler model called CRATE. Instead of categorizing and ordering, CRATE contains *undifferentiated* metadata packaged together with the resource in a complex-object HTTP response.

An advantage with this simple approach is that it is *data-centric* rather than ontology-based. For example, web servers would not need to choose between METS and MPEG-21 archiving services. Similarly, the archiving repository would not need to worry about non-librarian webmasters misusing METS headers or MPEG-21 descriptors. Instead, the archiving repository could harvest the site and transform the information according to its own model, or it could adopt a store-and-wait philosophy, like the file purgatory mentioned by Clay Shirky [24]. Another advantage with the CRATE model is that it readily expands to include new types of metadata without requiring an adaptation, re-evaluation or reassignment of current metadata fields.

```
Jhove (Rel. 1.1, 2006-06-05)
Date: 2007-04-15 15:08:02 EDT
RepresentationInformation:
/var/www/Foo.pdf
ReportingModule:
PDF-hul, Rel. 1.5 (2006-03-31)
LastModified: 2007-03-14 10:00:21 EDT
Size: 3599031
Format: PDF
Version: 1.2
Status: Well-Formed and valid
SignatureMatches:
PDF-hul
MIMEtype: application/pdf
PDFMetadata:
Objects: 56
FreeObjects: 1
IncrementalUpdates: 1
DocumentCatalog:
PageLayout: SinglePage
PageMode: UseNone
Info:
Title: FooDraft.dvi
Creator: dvips(k) 5.95a Copyright 2005
  Radical Eye Software
Producer: ESP Ghostscript 815.02
CreationDate:
  Wed Mar 14 10:00:21 EDT 2007
ModDate: Wed Mar 14 10:00:21 EDT 2007
ID: 0xd001996ca55b3a9037ec31b8e1c5f5ce29,
0xd001996ca55b3a9037ec31b8e1c5f5ce29
Filters:
FilterPipeline: FlateDecode
FilterPipeline: DCTDecode
Images:
Image:
 NisoImageMetadata:
  MIMEType: application/pdf
  CompressionScheme: JPEG
  ColorSpace: RGB
  ImageWidth: 459
  ImageLength: 253
  BitsPerSample: 8
Fonts:
Type1:
 Font:
  BaseFont: SWPMHB+CMR9
  FontSubset: true
  FirstChar: 11
  LastChar: 123
  FontDescriptor:
   FontName: SWPMHB+CMR9
   Flags: Symbolic
   FontBBox: -39, -250, 1036, 750
   FontFile3: true
  EncodingDictionary:
   BaseEncoding: WinAnsiEncoding
   Differences: true
 Font:
  BaseFont: BAWORV+CMBX9
  FontSubset: true
  FirstChar: 45
  LastChar: 122
  FontDescriptor:
   FontName: BAWORV+CMBX9
   Flags: Symbolic
   FontBBox: -58, -201, 1076, 700
   FontFile3: true
  Encoding: WinAnsiEncoding
 Font:
  BaseFont: Helvetica
 Font:
  BaseFont: Times-Italic
 Font:
  BaseFont: YSMOGU+CMR7
  FontSubset: true
  FirstChar: 12
  LastChar: 123
  FontDescriptor:
   FontName: YSMOGU+CMR7
   Flags: Symbolic
   FontBBox: 0, -250, 965, 750
   FontFile3: true
  EncodingDictionary:
   BaseEncoding: WinAnsiEncoding
   Differences: true
 Font:
  BaseFont: Helvetica-Bold
 Font:
  BaseFont: ZDPLIY+CMTT9
  FontSubset: true
  FirstChar: 34
  LastChar: 122
  FontDescriptor:
   FontName: ZDPLIY+CMTT9
   Flags: FixedPitch, Symbolic
   FontBBox: -6, -228, 524, 694
   FontFile3: true
  Encoding: WinAnsiEncoding
 Font:
  BaseFont: RAJQQU+CMTI9
  FontSubset: true
  FirstChar: 11
  LastChar: 121
  FontDescriptor:
   FontName: RAJQQU+CMTI9
   Flags: Symbolic
   FontBBox: -25, -205, 866, 705
   FontFile3: true
  EncodingDictionary:
   BaseEncoding: WinAnsiEncoding
   Differences: true
 Font:
  BaseFont: DMRRSL+CMSY9
  FontSubset: true
  FirstChar: 102
  LastChar: 103
  FontDescriptor:
   FontName: DMRRSL+CMSY9
   Flags: Symbolic
   FontBBox: 0, -250, 440, 750
   FontFile3: true
  EncodingDictionary:
   BaseEncoding: WinAnsiEncoding
   Differences: true
 Font:
  BaseFont: Times-Bold
 Font:
  BaseFont: IPBEUP+CMTI7
  FontSubset: true
  FirstChar: 48
  LastChar: 121
  FontDescriptor:
   FontName: IPBEUP+CMTI7
   Flags: Symbolic
   FontBBox: -17, -204, 1268, 713
   FontFile3: true
  Encoding: WinAnsiEncoding
 Font:
  BaseFont: OBAZBH+CMMI9
  FontSubset: true
  FirstChar: 60
  LastChar: 62
  FontDescriptor:
   FontName: OBAZBH+CMMI9
   Flags: Symbolic
   FontBBox: 0, -51, 712, 552
   FontFile3: true
  Encoding: WinAnsiEncoding
 Font:
  BaseFont: Times-Roman
  EncodingDictionary:
   Differences: true
Pages:
Page:
 Sequence: 1
Page:
 Sequence: 2
```

Figure 7: Metadata derived from Foo.pdf using Jhove's PDF-HUL module. The default plain-text output is used here for clarity and brevity. Command Line: /opt/jhove/jhove -c /opt/jhove/conf/jhove.conf -m pdf-hul Foo.pdf

The new information simply becomes part of the CRATE complex object, available for use or disuse by the archiving repository.

The OAIS model describes an Information Package as the "Content Information and associated Preservation Description Information which is needed to aid in the preservation of the Content Information. The Information Package has associated Packaging Information used to delimit and identify the Content Information and Preservation Description Information." [7]. Acknowledging "the reality that some submissions to an OAIS will have insufficient Representation Information or PDI to meet final OAIS preservation requirements", the CCSDS further describes three variants of the information package: submission (SIP), archival (AIP), and dissemination (DIP). Although resources accessible from a web server are unlikely to meet the criteria for SIPs or AIPS, later we introduce a method by which web servers can make a best-effort attempt to generate the content, preservation description, packaging, and descriptive information necessary to promote their web resources to a SIP or DIP at the time of an HTTP request.

4.1 Building a CRATE

An important characteristic shared by the models discussed in § 3 is that they are mostly human-readable, plain ASCII. With the exception of ARC, the model objects are also expressed in XML. Fortunately, most analysis utilities that would be likely candidates for web server installation generate their output in ASCII and/or XML. Since resources can be converted to ASCII using Base64 encoding, content can also be included in an XML document. CRATE adopts this approach, using plain ASCII and XML to express CRATE contents. A conceptual view of a CRATE is shown in Figure 8. Note that only 3 elements are defined for a CRATE: (1)an identifier; (2)metadata; and (3)the resource. There is one identifier per CRATE, which acts as a means to disambiguate among a collection of CRATE objects. A single CRATE can have up to one resource, but an unrestricted number of metadata elements. Figure 9 expresses the CRATE complex object as a UML diagram. For an object identifier to be unique and viable, it must be compatible with the system storing it. Since archiving repository characteristics can vary so widely, the CRATE Identifier is generated *by the crawling repository*, rather than by the crawled host. As the AIHT report noted, "identifiers often aren't" [24]. In any case, most repositories have their own methods for uniquely labelling each ingested resource. Expecting the small local web server to create an identifier that is simultaneously unique, disambiguates, and

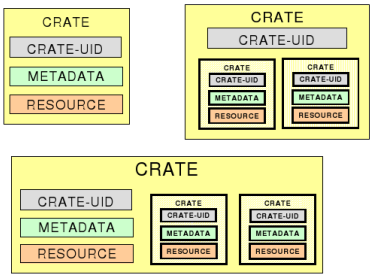


Figure 8: Examples of CRATE configurations

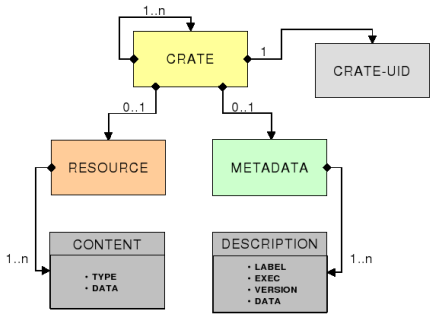


Figure 9: CRATE Complex Object

is compatible across all repositories seems unrealistic. Even using the UUID utility commonly found on UNIX/LINUX systems has disadvantages since it would have to be pregenerated and stored for each resource if it were to act as a long-term unique resource identifier across systems. Resource disambiguation or idempotence between repositories that have crawled the same sites can be done using the metadata elements of the CRATE.

Metadata is the heart of CRATE. Our goal is to automate the resource-description process; to have resources describe themselves in type-appropriate and sufficient detail; and to lower the barrier to preservation by simplifying participation requirements while maximizing resource information. Just as content types and versions vary from web site to web site, the number and type of utilities that are practical for installation on any individual web server will also vary. Archival crawls of sites will therefore produce a widely varying amount of resource information. This is partly because the kind of information useful in preserving resources varies with the type of the resource. A color index is useful in describing a JPEG; a key-word index is useful in describing an ASCII text file.

The Resource component of a CRATE can have one or many expressions. Each expression contains a `TYPE` element which describes the kind of content (text/plain, text/html, etc.), and a `CONTENT` element with the byte stream of the original resource. If the original resource is binary (a JPEG, e.g.) then a lossless encoding method such as Base64 is used. A Resource component can be expressed in more than one `TYPE`: as both text/html and text/ascii (i.e., a Base64-encoded resource), for example, but it is otherwise idempotent. The `CONTENT` field should produce a duplicate of the original resource, either directly or by reverse-encoding (e.g., Base64 back to binary). The Metadata component can have one or more `DESCRIPTION` items. A Description item has 4 elements that categorize the source and context of the metadata it contains, i.e., metadata about the metadata. A Description element does not necessarily have to hold the resource metadata. It could contain a citation to a remote utility that the harvesting crawler could use to further analyze the resource, or it could point to a location that already contains detailed information about the resource. The archiving crawler would determine when and whether to access that information.

Example CRATE configurations are shown in Figure 8. Note that CRATE objects can be nested both broad and deep. An archiving service can use this structure to associate time-based variations of an archived resource, to package

the full content of a web site, or to keep a complex HTML resource together with its embedded multimedia content.

4.2 CRATE and Other Complex Object Models

A major difference between CRATE and other complex object models is that CRATE does not have any minimum metadata requirements other than a unique identifier. The number and type of metadata elements available can vary greatly from resource to resource, and from site to site.

Ultimately, the primary difference between a CRATE and other preservation-oriented complex object models is the metadata component. In a CRATE, all metadata is undifferentiated. Instead of populating specific metadata fields, with all of the inherent world-view such activity implies, we focus on capturing the output of analysis utilities and leave issues of parsing and provenance to the future archivist. Utilities can generate conflicting or ambiguous information which requires considerable experience and knowledge to parse and place within a repository's model. In contrast, CRATE makes no declaration about the assignment of Jhove output (Figure 7) versus Acroread output (Figure 4) in METS descriptive and technical metadata sections, for example. CRATE also makes no assertion regarding metadata validity; it merely reports the metadata produced from the utilities. This "freedom from choice" lets web sites adopt preservation practices without having to incur archival knowledge per se, and allows any number of archiving repositories to collect the information and parse it according to their own individual models.

A CRATE metadata component is characterized by the four description elements, `label, exec, version and data`. Interpretation and categorization of a metadata component and its element contents is left to the archiving repository. For example, a METS-based repository would categorize each metadata component into one of the 7 METS types, such as "Administrative" or "Structural". In the CRATE model, the *context* in which the metadata was generated, the flexibility to have a wide variety of metadata content, and the opportunity to take advantage of leading-edge utilities, are more important than defining CRATE-unique categories. This aspect facilitates mapping of CRATE information to other complex object models including METS and MPEG-21.

5. CURRENT STATUS

CRATE objects can be built in a number of ways. For example, a web crawler could be modified to process each resource upon successful GET, then pass the received content to various utilities on the crawler's server, and finally package the resulting metadata together with the resource as a CRATE object. This solution is similar to the Old Dominion University approach to the AIHT project [16] where we built "self-archiving" objects from the original tar file, but in this case instead of using "Buckets" as an archival storage facility [17], we build a CRATE [17]. VEO, METS or MPEG-21 repositories could build CRATE objects by exporting their resources and metadata in the CRATE format. For example, DSpace plugins or extensions for Fedora could be used to build CRATE objects from repository contents. Another alternative would be to run a post-crawl script on an entire web site, aggregating the collection of CRATE'd resources into a site CRATE.

An interesting approach is suggested by the new web service "Yahoo!Pipes" [8] which uses a web server in a way similar to the implementation of a Unix "pipe" command. For example, a resource from our sample web site could be fed through Yahoo!Pipes to a series of other sites which would analyze it. An additional Yahoo!Pipe would aggregate the information into CRATE format. In other words, we could use Yahoo!Pipes to build a CRATE.

As another alternative, the web site could create a CGI script which would respond to a request such as:

```
http://foo.edu/?crate=Foo.pdf
```

by processing the resource and providing the metadata + resource in the CRATE object format as the HTTP response. In this latter case, the processing burden is put upon the host server rather than on the crawling/archiving server. Similarly, if we registered a new MIME type - for example, `application/crate+xml` - the crawler could parameterize the resource request with that MIME type. A more practical solution that also uses the host web server's processing power is a *web server module* which is installed and integrated with the web server by the local webmaster.

We believe the module approach has a number of advantages in addition to sharing the processing cost of resource analysis. First, implementation is simple. Modules are a routine installation task for webmasters, requiring close to zero effort. Most web servers have a number of modules installed, including those for processing CGI requests or managing directory displays, for example. Secondly, as an Open Source product (such as mod_oai), there is no financial investment required of the web site. Another advantage is that it is simple to fine-tune the module to meet local resource profiles (only GIFs, only JPEGs, some of everything, etc.). Finally, the web server response now produces something that is much closer to the ideal of being "born archival" and not just "born digital."

Figure 10 gives an example Apache configuration section for building CRATE modules using mod_oai. The three CRATE description elements, "label", "exec" i.e., the command to invoke on the resource, and "version" to make note of the utility version information, are itemized in the configuration file. The first utility calculates the MD5 hash of the resource content, providing one possibility for disambiguation of files changed between crawls, or for locating files that appear to be identical. Next, the Unix "file" command is invoked to provide more descriptive metadata than standard MIME typing. We have added switches that direct the program to look beyond the initial assessment (-k, continue) and to peer into compressed files (-z). Jhove is implemented in two ways. First, we call Jhove via a script called "pre-Jhove" which uses the resource's MIME type to determine which Jhove HUL module should be applied to the resource. The script is only applied to image-type files, since there are a number of varying image analysis modules but relatively few images on our sample web site. For common files, like PDFs, it is faster and simpler to call the Jhove utility directly, which is what is done in the subsequent plugin. The "`%s`" within the quoted `exec` field denotes the resource name which is substituted by Apache during the GET response. A percent (%) symbol outside of quotes indicates a comment. Although this perhaps appears complicated to readers who are not also webmasters, the example configuration file shown is readily implemented and understood by system

administrators and web masters who manage Apache web servers.

```
<Location /modoai>
 SetHandler   modoai-handler
 modoai_oai_active ON
 <modoai_plugin>
  label "md5sum"
  exec "/usr/bin/md5sum %s"
  version "/usr/bin/md5sum --version"
  mime "*/*"
 </modoai_plugin>
 <modoai_plugin>
  label "file"
  exec "/usr/bin/file -kz %s"
  version "/usr/bin/file -v"
  mime "*/*"
 </modoai_plugin>
 <modoai_plugin>
  label "jhove"
  exec "/var/www/preJhove %s"
  version "/opt/jhove/jhove -v"
  mime "image/*"
 </modoai_plugin>
 <modoai_plugin>
  label "jhove"
  exec "/opt/jhove/jhove -m pdf-hul %s"
  version "/opt/jhove/jhove -v"
  mime "application/pdf"
 </modoai_plugin>
 <modoai_plugin>
  label "ots"
  exec "/usr/local/bin ots -summary %s"
  version "/usr/local/bin ots -v"
  mime "text/*"
 </modoai_plugin>
 <modoai_plugin>
  label "pronom"
  exec "java -jar DROID.jar -L%s"
  version "java -jar DROID.jar -V"
  mime "*/*"
 </modoai_plugin>
</Location /modoai>
```

Figure 10: Location section from an Apache mod_oai configuration file showing metadata plugin implementation for building CRATE objects

To test the CRATE concept, we extended the functionality of an Apache OAI-PMH module, *mod_oai*, to support a plugin architecture. The module responds to OAI-PMH-style requests according to the metadata format specified [12, 18]. The original module supported Dublin Core, HTTP-header, and MPEG-21 DIDL metadata formats. We added a new format called "oduCrate" to aggregate the results of plugin tools and other metadata with the Base64-encoded resource. Installation of this extended module enables preservation-specific requests to be made of the server without impacting its ability to respond to the usual HTTP requests [25]. A user asking for `http://foo.edu/barr.html` will still get the same document as before. To access the CRATE version of the object, the HTTP request string is in a form that specifies a CRATE-type response is expected. Using OAI-PMH syntax, for example, the request would look like this:

```
http://foo.edu/modoai/?verb=GetRecord
&identifier=http://foo.edu/barr.html
&metadataPrefix=oduCrate
```

The response returns `barr.html` together with metadata automatically generated for that specific resource, such as output from the Unix "file" command, Kea, Open Text Summarizer (OTS) [3], and other utilities. The server can therefore function simultaneously as both an agent of preservation and as a normal web server.

We installed several utilities on our web server, and configured mod_oai plugins to act on resources based on MIME type. Open Text Summarizer (OTS), for example, was only applied to text-based resources and not to images. Jhove HUL modules were called based on MIME sub-type (ASCII, JPEG or TIFF) using a wrapper script. This type of customization required only a few lines of `directive` instruction in the Apache configuration file.

6. FUTURE WORK AND CONCLUSIONS

How does preservation-crawling impact the server? Metadata utility selection clearly plays a role, since some are more processing-intense than others. The type and distribution of web resources on a site also affects what level of system-effort is required to dynamically extract metadata. Since the metadata generation occurs at dissemination time, we are currently conducting experiments on the impact of various tools like Jhove on web server performance. As noted above, individual tools can be targeted to specific resource types or to sections of the site tree using the <LOCATION> directive, preventing resource+tool mismatch. Even so, dynamic analysis of resources must obviously require more of a server's CPU cycles than a simple GET request for the plain resource. Whether or not this will prove to be a reasonable trade off for sites interested in long-term preservation remains to be seen.

What incentives are there for webmasters to implement even this basic approach? CRATE does not require substantial investment by sites: no extra funding, no software costs (other than installation of an open source module), no "donation" of time or resources to the general public, and no need for a trained, in-house archivist. Sites with an obligation to adopt a preservation strategy but which lack the funds to support more formal solutions like VEO may find this simpler approach attractive. Archiving repositories may provide incentives by offering an "entry-level" preservation service to such agencies, or to college departments interested in longer-term preservation as opposed to near-term backup strategies. In summary, we believe that this generic, web-server-based approach which is simple to install and easy to maintain has a higher likelihood of adoption by everyday web sites than systems which call for a more sophisticated level of effort.

7. REFERENCES

[1] The Wayback Machine Frequently Asked Questions. `http://www.archive.org/about/faqs.php`.

[2] Google code: Web authoring statistics. `http://code.google.com/webstats/`, 2006. Accessed on 10 Feb 2007.

[3] Open text summarizer. Open source tool for summarizing texts, 2006. http://libots.sourceforge.net/.

[4] S. L. Abrams and D. Seaman. Towards a global digital format registry. *World Library and Information Congress: 69th IFLA General Conference and Council*, pages 1–9, August 2003.

[5] M. S. Bains. The search engine economy's Achilles heel? Addressing online parallel imports resulting from keyword and metatag misuse. *Stanford Technology Law Review*, 6, October 2006.

[6] J. Bekaert, X. Liu, and H. Van de Sompel. Representing digital assets for long-term preservation using MPEG-21 DID. In *Ensuring Long-term Preservation and Adding Value to Scientific and Technical Data (PV 2005)*, 2005.

[7] CCSDS. Reference model for an open archival information system (ISO 14721:2002). Technical Report CCSDS 650.0-B-1, Consultative Committee for Space Data Systems, January 2002.

[8] N. Cubrilovic. Yahoo! launches pipes. http://www.techcrunch.com/2007/02/07/yahoo-launches-pipes/, February 2007.

[9] H. M. Gladney. Trustworthy 100-year digital objects: Evidence after every witness is dead. *ACMTransactions On Information Systems*, 22(3):406–436, July 2004.

[10] B. P. Heath, D. J. McArthur, M. K. McClelland, and R. J. Vetter. Metadata lessons from the iLumina digital library. *Communications of the ACM*, 48(7):68–74, July 2005.

[11] J. Hunter and S. Choudhury. A semi-automated digital preservation system based on semantic web services. In *Joint Conference on Digital Libraries (JCDL 2004)*, pages 269–278, 2004.

[12] C. Lagoze, H. Van de Sompel, M. L. Nelson, and S. Warner. The Open Archives Initiative Protocol for Metadata Harvesting. http://www.openarchives.org/OAI/openarchivesprotocol.html.

[13] X. Liu, L. Balakireva, P. Hochstenbach, and H. Van de Sompel. File-based storage of digital objects and constituent datastreams: XML Tapes and Internet Archive ARC files. In *9th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2005)*, pages 254–265, Sept 2005.

[14] J. P. McDonough. METS: Standardized encoding for digital library objects. *International Journal on Digital Libraries*, 6(2):148–158, April 2006.

[15] G. Mohr, M. Kimpton, M. Stack, and I. Ranitovic. Introduction to heritrix, an archival quality web crawler. In *Proceedings of the 4th International Web Archiving Workshop (IWAW '04)*, Sept 2004.

[16] M. L. Nelson, J. Bollen, G. Manepalli, and R. Haq. Archive ingest and handling test, the Old Dominion University approach. *D-Lib Magazine*, 11(12), October 2005. doi:10.1045/december2005-nelson.

[17] M. L. Nelson and K. Maly. Smart objects and open archives. *D-Lib Magazine*, 7(2), February 2001. doi:10.1045/february2001-nelson.

[18] M. L. Nelson, J. A. Smith, H. Van de Sompel, X. Liu, and I. Garcia del Campo. Efficient, automatic web resource harvesting. *Proceedings of the eighth ACM international workshop on web information and data management*, November 2006.

[19] OCLC/RLG PREMIS Working Group. Implementing preservation repositories for digital materials: Current practice and emerging trends in the cultural heritage community. Report by the joint OCLC/RLG Working Group Preservation Metadata: Implementation Strategies (PREMIS), 2004.

[20] OCLC/RLG PREMIS Working Group. Data dictionary for preservation metadata. Final report of the PREMIS working group. Report by the joint OCLC/RLG Working Group Preservation Metadata: Implementation Strategies (PREMIS), May 2005.

[21] S. Payette and C. Lagoze. Flexible and extensible digital object and repository architecture (FEDORA). In *ECDL '98: Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries*, pages 41–59, London, UK, 1998. Springer-Verlag.

[22] J. Rothenberg. Ensuring the longevity of digital information. Council on Library and Information Resources, February 1999. Revision 980327. http:www.clir.org/pubs/archives/ensuring.pdf.

[23] C. Shirky. AIHT: Conceptual issues from practical tests. *D-Lib Magazine*, 11(12), December 2005. http://www.dlib.org/dlib/december05/shirky/12shirky.html.

[24] C. Shirky. Library of Congress Archive Ingest and Handling Test (AIHT) final report. Report by the National Digital Information Infrastruction & Preservation Program, June 2005.

[25] J. A. Smith and M. L. Nelson. Generating best-effort preservation metadata for web resources at time of dissemination. In *Proceedings of the Joint Conference on Digital Libraries (JCDL 2007)*, June 2007.

[26] R. Tansley, M. Bass, D. Stuve, M. Branschofsky, D. Chudnov, G. McClellan, and M. Smith. The DSpace institutional digital repository system: current functionality. In *JCDL '03: Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 87–97, Washington, DC, USA, 2003. IEEE Computer Society.

[27] C. F. Thomas and L. S. Griffin. Who will Create The Metadata For the Internet? *First Monday*, 3(12), 1998. http://www.firstmonday.dk/issues/issue3_12/thomas/.

[28] H. Van de Sompel, M. L. Nelson, C. Lagoze, and S. Warner. Resource harvesting within the OAI-PMH framework. *D-Lib Magazine*, 10(12), December 2004. doi:10.1045/december2004-vandesompel.

[29] Victoria Electronic Records System. VERS toolkit. http://www.prov.vic.gov.au/vers/toolkit/, 2005.

[30] A. Waugh. The design of the VERS encapsulated object experience with an archival information package. *International Journal on Digital Libraries*, 6(2):184–191, April 2006.