ISSN 1749-8767

DCC | Digital Curation Centre Case Studies and Interviews

JSTOR/Harvard Object Validation Environment (JHOVE)

http://www.dcc.ac.uk/resource/case-studies/jhove

Martin Donnelly
HATII, University of Glasgow,
Glasgow G12 8QQ
http://www.hatii.arts.gla.ac.uk

March 2006
Version 1.0

Legal Notices

Catalogue Entry

Title	DCC Case Study – JSTOR/Harvard Object Validation Environment (JHOVE)
Creator	Martin Donnelly (author)
Subject	Data curation; formats, processes and issues; interoperability; system development; standards; legal factors; justification; methodology, and problems overcome; human factors
Description	
Publisher	HATII, University of Glasgow; University of Edinburgh; UKOLN, University of Bath; Council for the Central Laboratory of the Research Councils.
Contributor	Joy Davidson and Andrew McHugh (editors)
Date	1 December 2005 (creation)
Type	Text
Format	Adobe Portable Document Format v.1.3 (Acrobat 4.x)
Resource Identifier	ISSN 1749-8767
Language	English
Rights	© HATII, University of Glasgow

Citation Guidelines

Donnelly M, (March 2006), "JSTOR/Harvard Object Validation Environment (JHOVE) ", *Digital Curation Centre Case Studies and Interviews*, Joy Davidson and Andrew McHugh (editors), Retrieved <date>, from http://www.dcc.ac.uk/resource/case-studies/jhove

About the DCC

The JISC-funded Digital Curation Centre (DCC) provides a focus on research into digital curation expertise and best practice for the storage, management and preservation of digital information to enable its use and re-use over time. The project represents a collaboration between the University of Edinburgh, the University of Glasgow through HATII, UKOLN at the University of Bath, and the Council of the Central Laboratory of the Research Councils (CCLRC). The DCC relies heavily on active participation and feedback from all stakeholder communities. For more information, please visit www.dcc.ac.uk. The DCC is not itself a data repository, nor does it attempt to impose policies and practices of one branch of scholarship upon another. Rather, based on insight from a vibrant research programme that addresses wider issues of data curation and long-term preservation, it will develop and offer programmes of outreach and practical services to assist those who face digital curation challenges. It also seeks to complement and contribute towards the efforts of related organisations, rather than duplicate services.

DCC Case Study and Interviews

Editors

Joy Davidson
Humanities Advanced Technology and Information Institute, University of Glasgow (UK)
Andrew McHugh
Humanities Advanced Technology and Information Institute, University of Glasgow (UK)

Peer Review Board

Case Study: JSTOR/Harvard Object Validation Environment (JHOVE) *(http://hul.harvard.edu/jhove/)*

This Digital Curation Centre case study is the result of an e-mail questionnaire completed by Stephen Abrams of Harvard University Library's Office for Information Systems in July/August 2005, on subsequent e-mail communication between Mr Abrams and Martin Donnelly of the DCC and HATII, University of Glasgow, and on information available on the web. References are given in footnotes in the text.

Table of Contents

JHOVE: Executive Summary

Accurate file format information is crucial for preserving access to and the rendering of digital information over time. As such, it is vital that when a digital object is deposited in a repository, the object in question is of the type it purports to be. However, the representation of file formats is easily corruptible - whether accidental or intentional. This is of particular concern to institutions with an interest in preserving digital materials in repositories. The JSTOR/Harvard Object Validation Environment (JHOVE) is an Open Source, extensible framework for the format-specific identification, validation, and characterisation of digital objects.

Introduction

Representation types (i.e. file formats) are of fundamental technical importance to digital repositories, and are central to administrative decisions regarding actions such as ingest, storage, access and migration. However, the representation of file formats is easily corruptible: taking an MP3 file, for example, and manually changing its file extension from .mp3 to .txt does not make it a valid text file. Granted, this would be an unusual thing to do, but it serves to demonstrate that file extensions are by no means impervious to corruption, whether accidental or intentional. This is of particular concern to institutions with an interest in preserving digital materials in repositories.

Within repositories, policies can vary greatly from format to format, so in order to maximise efficiency, the processes of format identification, validation and characterisation need to be automated as much as possible. To overcome this problem, Harvard University Library (HUL) and JSTOR — The Scholarly Journal Archive — have created an extensible framework for format validation: the JSTOR/Harvard Object Validation Environment (JHOVE). In the developers' own words, JHOVE "provides functions to perform the format-specific identification, validation, and characterisation of digital objects." In short, when an object is submitted to a repository, JHOVE can be used to confirm that it is what it claims to be.

Outline of current work

Since 2002, HUL has operated its own digital repository for preservation purposes: the Digital Repository Service (DRS). Before JHOVE, digital objects submitted for deposit in the DRS were subject to validation based solely on magic numbers — short internal signatures typically found near the beginning of object bitstreams.[1] This approach is clearly insufficient for rigorous validation: an object that consisted solely of the magic number would be reported as valid despite containing no information content, and therefore being incapable of sensible rendering. This problem was the key impetus for the development of a dedicated framework for format validation.

The idea for JHOVE arose in 2003 during discussions between staff at the Harvard University Library (HUL) and JSTOR over topics of mutual interest. The initial conversation revolved around a tool dedicated to Adobe's Portable Document Format (PDF), which is designed to preserve formatting and provide additional security — but the potential advantages of generalising the tool to provide extensible support for a larger set of formats quickly became apparent.

HUL and JSTOR agreed jointly to fund a development project which would develop JHOVE as an Open Source, extensible framework for format-specific identification, validation, and characterisation. The original project plan had a seven-month schedule, resulting in JHOVE support for ASCII, UTF-8, PDF, and TIFF formats. This timeframe was later extended to ten months, with the extra time used to provide additional support for GIF, JPEG, and XML.

Data types

JHOVE supports a wide variety of digital objects routinely found in digital libraries and repositories. The initial selection of supported formats was developed cooperatively by HUL and JSTOR, and focused on formats used for digital surrogates of textual documents (ASCII, PDF, UTF-8, XML) and cultural heritage visual materials (GIF, JPEG, TIFF). Further system development added requirements for digital surrogates of textual documents (HTML), visual material (JPEG 2000), and audio (AIFF, WAVE). Within each of these formats, JHOVE recognises a number of different profiles / format versions.[2]

[1] To complicate matters, magic numbers vary according to the ways in which different computers store byte sequences. For example, the magic number for TIFF objects is 0x4D4D002A if the computer uses a 'big-endian' storage system, or 0x49492A00 if 'little-endian'. See http://www.webopedia.com/TERM/b/big_endian.html for a full explanation of these terms.

[2] For example, the TIFF module can distinguish between TIFF 4.0, 5.0, and 6.0 (including Classes B, G, P, R, Y), TIFF/IT (ISO 12639), TIFF/EP (ISO 12234-2), GeoTIFF, Exif 2.0, 2.1 (JEIDA-49-1998) and 2.1 (JEITA CP-3451), TIFF-FX (RFC 2301), Class F (RFC 2306), RFC 1314, and DNG. The JHOVE web site contains descriptions of all available modules, including the profiles recognised by each module and the specific criteria for that recognition. http://hul.harvard.edu/jhove/documentation.html

System operation

The project's main deliverable was an extensible framework for format-specific object identification, validation, and characterisation. To this end, JHOVE performs three types of operations:

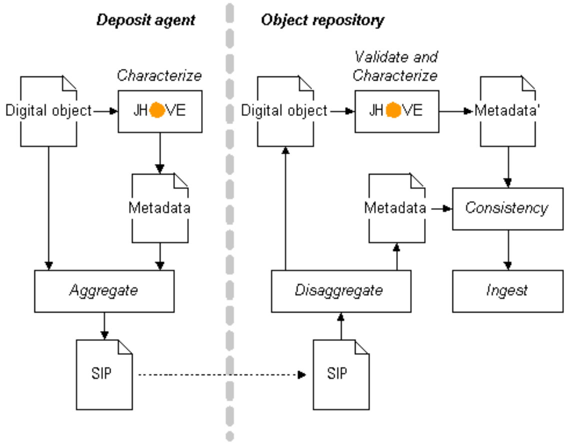
1) Identification, which determines the format of a given digital object;

2) Validation, which determines whether or not a given digital object is of the format it purports to be. (Like XML, format validation conformance is determined at two levels: *well-formedness* and *validity*);

3) Characterisation, which extracts the salient technical properties of a given digital object of known format.

Used in conjunction, these three operations support routine repository operations, as well as preservation planning and intervention.[3] JHOVE characterisation exposes the technical properties of all of the component structures that are examined during a validation operation. The intent of characterisation is to provide as complete a description of the object as possible, without actually rendering it.

The architecture and operation of HUL's preservation repository is consistent with the OAIS reference model, with JHOVE's validation and characterisation functions occurring at the point of ingest. The comparison of this characterisation with the external technical metadata found in the repository-compliant Submission Information Package (SIP) is a part of the ingest validation process.



The validation process examines a formatted digital object with respect to all of the internal data structures that are defined by the format specification. For example, a valid TIFF file will comprise a header followed by one or more Image File Directories (IFDs). A valid IFD is composed of a count of entries, followed by the entries themselves, and then the offset of the subsequent IFD.[4] In turn, each IFD entry is composed of four fields: (i) a tag indicator; (ii) a type indicator; (iii) a count of values; and (iv) the values themselves, either directly — in the entry itself — or indirectly — via a pointer to another location in the file. All of these structures must be present, and must conform to a number of restrictions for the file to be considered consistent with the TIFF specification. For example, the IFD of the file “little_endian.tif”, included as part of the standard JHOVE distribution package, appears as follows:[5]

[3] In terms of the Open Archival Information System (OAIS) reference model (ISO 14721), the need for these operations applies across the Ingest, Archival Storage, Access, and Preservation Planning functions.

[4] The ‘offset’ indicates the starting byte address of the subsequent IFD. N.B. there is no requirement that IFDs appear consecutively in the TIFF file; IFDs and image data can be freely intermingled.

[5] Note that this file is encoded with little-endian byte ordering.

Hexadecimal form	Decimal form
0f00	15
fe00 0400 01000000 00000000	254 4 1 0
0001 0400 01000000 840b0000	256 4 1 2948
0101 0400 01000000 0c120000	257 4 1 4620
0201 0300 01000000 01000000	258 3 1 1
0301 0300 01000000 04000000	259 3 1 4
0601 0300 01000000 00000000	262 3 1 0
1101 0400 01000000 08020000	273 4 1 520
1201 0300 01000000 01000000	274 3 1 1
1501 0300 01000000 01000000	277 3 1 1
1601 0400 01000000 0c120000	278 4 1 4620
1701 0400 01000000 ac640000	279 4 1 25772
1a01 0500 01000000 c2000000	282 5 1 194
1b01 0500 01000000 ca000000	283 5 1 202
2801 0300 01000000 02000000	296 3 1 2
3101 0200 32000000 d2000000	305 2 50 210
00000000	0

Since this data is properly formed (mandatory tags are present; tags are ordered numerically; type indicators and counts are consistent with the tag definitions in the TIFF specification) it can be decoded by JHOVE as follows:

```
jhove -m tiff-hul examples/tiff/little-endian.tif
Jhove (Rel. 1.0, 2005-05-26)
 Date: 2005-10-03 11:06:27 EDT
 RepresentationInformation:
examples/tiff/little-endian.tif
  ReportingModule: TIFF-hul, Rel. 1.4
(2005-06-10)
  LastModified: 2003-08-06 16:09:48 EDT
  Size: 26292
  Format: TIFF
  Version: 5.0
  Status: Well-Formed and valid
  SignatureMatches: TIFF-hul
  MIMEtype: image/tiff
  Profile: TIFF/IT-BP/P2 (ISO 12639:1998)
  TIFFMetadata:
   ByteOrder: little-endian
   IFDs:
    Number: 1
    IFD:
     Offset: 8
     Type: TIFF
     Entries:
      NisoImageMetadata:
       MIMEType: image/tiff
       ByteOrder: little-endian
       CompressionScheme: CCITT
Group 4
       ColorSpace: white is zero
       StripOffsets: 520
       RowsPerStrip: 4620
       StripByteCounts: 25772
       PlanarConfiguration: chunky
       Orientation: normal
       ScanningSoftware: Pixel
Translations Inc., PIXTIFF
                  Version 54.2.210
       SamplingFrequencyUnit: inch
       XSamplingFrequency: 600
       YSamplingFrequency: 600
       ImageWidth: 2948
       ImageLength: 4620
       BitsPerSample: 1
       SamplesPerPixel: 1
```

JHOVE reports validation at two levels: (i) well-formed; and (ii) valid. An object is considered *well-formed* if all of the individual component structures are correct; in other words, well-formedness is a local propert*y*. An object is considered *valid* if there is overall consistency between the individual component structures / semantic-level requirements; in other words, validity is a global property.[6] In order to validate an object, JHOVE invokes validation operations according to each of the format-specific modules, until one module reports that the object is well-formed.

A GIF formatted object, for example, will be considered well-formed by JHOVE if it meets the following criteria: (i) a properly formed signature (“GIF” at byte offset 0) and version identifier (“87a” or “89a” at byte offset 3); (ii) a

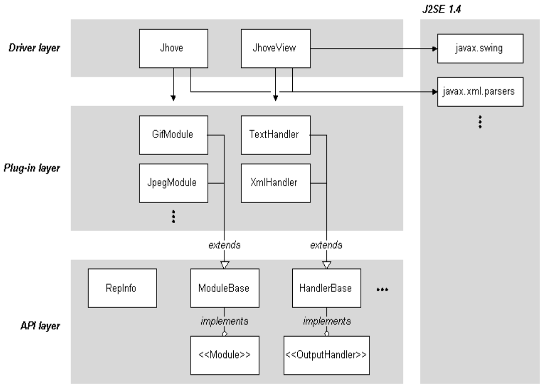
[6] The JHOVE web site gives this example: “A TIFF object is well-formed if it starts with an 8 byte header followed by a sequence of Image File Directories (IFDs), each composed of a 2 byte entry count and a series of 8 byte tagged entries. The object is valid if it meets certain additional semantic-level rules, such as that an RGB file must have at least three sample values per pixel.” http://hul.harvard.edu/jhove/

sequence of properly formed control, graphic-rendering, and special purpose blocks; and (iii) a terminator (0x00) in the last byte. Furthermore, the object will be considered valid if: (iv) it is well-formed; (v) it has at most one global colour map; and (vi) it has at most one graphic control extension preceding each image descriptor or plain text extension.

System outline

JHOVE's standard distribution package provides two interfaces: a command-line interface, useful for lower-level DOS and UNIX platforms, and a more user-friendly Java Swing-based graphical user interface (GUI) for Windows-based platforms.

In order to maximise its usability in the widest number of computing environments, JHOVE was implemented in Java. The developers deliberately avoided the use of third-party software in order to simplify distribution, and to avoid potential licensing or intellectual property rights issues. The use of JHOVE requires only a Java 1.4 runtime environment, freely downloadable from the web.



The system's plug-in architecture revolves around two key interfaces: *Module* and *OutputHandler*. The Module interface defines the Application Programming Interface (API) for each of the different format *modules*, one of which is required for each format covered. The set of modules covered by a particular JHOVE installation is controllable at invocation time by an XML-formatted configuration file.

The development process for each new module begins with the collection of authoritative specifications for the format and its various profiles. The first important decision is whether the format can be fully parsed in a linear, sequential manner, or whether it must be parsed in a random access manner. Once this is known, the isRandomAccess() method can be implemented, and then the proper form of the parse() method:

void parse(RandomAccessFile raf, RepInfo info);

int parse(InputStream stream, RepInfo info, int parseIndex);[7]

Each module is responsible for parsing the formatted bit stream, determining its compliance to the established specification for that format, and extracting relevant technical properties. These properties are encapsulated into a representation information (RepInfo) object. RepInfo permits the construction of an arbitrary hierarchically structured tree of properties of various *types* (e.g., String, Integer, Float) and *arities* (Scalar, Array, List), which have similar semantics to those of their Java equivalents.[8]

The display of the RepInfo data is the responsibility of an output handler. The OutputHandler interface defines the API for these handlers. Three handlers are included as part of the standard distribution: LINK TO RIR INFO

(i) Text, which uses simple name/value pairs;

(ii) XML, which uses a JHOVE-specific schema as a container for external schemas specific to certain media types, e.g. MIX for still image metadata;[9]

(iii) Audit, which produces a summary XML-formatted display useful for getting a quick sense of a large set of objects.

System development

The development process for the majority of the format modules was reasonably smooth. The PDF module took some time to complete due to the significant complexity of the underlying format. The most difficult module was the one for HTML. HTML is defined in terms of an SGML Document Type Definition

[7] The parseIndex argument is necessary to permit iterative parsing of HTML and XML formatted objects.

[8] See http://en.wikipedia.org/wiki/Arity for more on arities.

[9] http://www.loc.gov/standards/mix/

(DTD), so the module development entailed the construction of a special-purpose SGML parser focused on that DTD. Additional difficulty was caused by the fact that so little existing HTML is actually well-formed, let alone valid by the strict terms of the specification. While the other modules were constructed to terminate processing at the first error, the HTML module was designed to recover from errors and continue, the intent being to allow the module to report a fairly comprehensive set of error conditions.

The most significant difficulty in developing new modules is generating or collecting an inclusive set of test files that exercise all documented format features. Ideally, test files would be available for all possible combinations of these six requirements, as well as all possible combinations of errors, but this proved impossible to attain. Test files were generated using a number of Open Source image software packages, and additional test files were accumulated from the public domain — freely usable files found on the web. These files permitted the exercise of most, but not all, logical paths through the module code.

In practice, JHOVE has been able to process most mainstream formatted objects — i.e. those that utilise the well-known features of the format in an orthodox manner — properly and without difficulty. Occasionally, however, problems will arise while processing objects that utilise an obscure feature or collection of features that were not adequately exercised by the existing test suite. Over time, however, these occurrences have become quite rare.

Standards and Legal factors

To date, all of the formats supported by JHOVE are Open / non-proprietary, allowing JHOVE's source-code to be freely distributed without raising intellectual property concerns. If support is added in the future for closed formats — such as those used for Microsoft Office, presumably developed under some sort of non-disclosure agreement — it will be necessary to develop alternative distribution strategies. No firm plans have yet been made with regard to these formats, but as HUL's Stephen Abrams says, they raise significant preservation concerns for institutional repositories.

All JHOVE modules conform to a well-documented Java interface, so it is possible to distribute modules in compiled form only (i.e. as *.class files rather than *.java files). However, this would mean that the accepted advantages of Open Source tools — such as community-driven, responsive error detection and correction — could no longer be relied upon, as access to the original code becomes more difficult.

JHOVE itself is distributed under the terms of the GNU Lesser General Public License (LGPL).

Justification

JHOVE's chief drivers relate to quality assurance and efficiency. The project concept was based primarily on the needs of HUL and JSTOR in operating large-scale preservation repositories for their digital assets. Potential usefulness for other institutions was a secondary, although not negligible, consideration. An environmental scan taken at the project's inception did not reveal any tools, either Open Source or commercial, that could provide the required functions for the formats initially identified as being of mutual interest to the two institutions.

Data curation and user types

The primary direct users of JHOVE are data curators, especially those developing regimes for preservation monitoring and intervention; data creators will be more likely to use JHOVE indirectly, as part of integrated tools based on the JHOVE technology. For example, HUL is developing a depositor tool for use with its repository, using JHOVE to extract technical metadata from digital objects intended for submission. This tool will automatically produce the required control file containing the technical and administrative metadata that accompanies the digital objects submitted to the repository.

Human factors

The development of JHOVE involved three distinct roles: a part-time (~10%) project manager; a part-time (~25%) senior analyst dealing with overall system architecture, analysis of format specifications, and profile validation and characterisation criteria; and a full-time programmer responsible for the implementation, testing, and documentation of the system.

As the programmer grew more skilled over the course of the project, he took on increasing responsibility for the format analysis tasks. Key design decisions (such as validation criteria) are peer-reviewed by relevant technical experts in HUL, Harvard, and beyond. Such external reviewing is considered to be particularly important for formats with which the development staff have had little prior experience.

JHOVE aims to improve the lot of the system user. The forthcoming JHOVE-based depositor tool will relieve digital object creators, owners and depositors from the necessity of collecting the technical metadata that must accompany objects submitted for deposit into the HUL repository. These activities are currently performed using a mixture of semi-automated and manual methods, but even the 'automated' methods rely on manually-created metadata templates. In practice, this template-based approach does not always truly capture the correct technical characterisation of the objects. A JHOVE-based approach, however, should alleviate this manual work, while simultaneously providing a much higher degree of veracity in the metadata characterisation.

Within HUL, JHOVE has proved to be useful in detecting malformed digital objects that would otherwise have been silently accepted into HUL's preservation repository. During a recent image reformatting project, JHOVE identified a number of TIFF files that were invalid due to violations of the requirement that all value offsets must be word-aligned.[10] These files were generated by a number of independent vendors using a variety of image processing tools. Since JHOVE also exposed the values of the TIFF Artist (315), Make (271), Model (272), and Software (305) tags, HUL staff were able to trace these errors back to the appropriate vendor and tool.

The vendors are all addressing these errors through a re-evaluation of their locally developed tools or by passing the error reports on to their tool vendors. Since this particular error does not, in general, affect rendering by current TIFF tools, it is unlikely that these invalid files would have been identified without the use of JHOVE. Although this is a non-critical error today, prudent preservation stewardship recommends that all reasonable efforts should be undertaken to ensure that stored digital objects are valid with respect to their formats, thereby facilitating the proper future processing by new generations of systems and services.

Evaluation

Other than the peer reviewing involved in the project's formative evaluation, HUL relies on the experience of internal users of JHOVE, in connection with its digital repository and preservation activities. Additionally, the developers receive informal comments and suggestions from peer institutions and the wider user community. Concurrent with the release of the first production version of JHOVE in May 2005, an online discussion list (JHOVE-users) was established in order to provide the digital library, archives, repository, and preservation communities a public forum for general discussion and announcements about the system.[11]

In August 2004, a JHOVE-based investigation of the 1.1 million objects then stored in HUL's digital repository revealed a small, but non-trivial number of invalid objects (7,040 objects, approximately 0.64% of the total ingest), as well as a larger number of inconsistencies between the externally-supplied technical metadata stored in the DRS and the technical metadata extracted by JHOVE from the objects themselves (2.1%). This demonstrates the need for such a tool, as well as the scale of the problem faced by digital repositories.

Public response to JHOVE has been uniformly positive, although — as with any complex system — hindsight has provided new views on what might have been designed or implemented better. Specific areas for future improvement include:

- API and internal class implementations leading to easier integration of JHOVE functionality inside of other systems;
- More sophisticated handling of representation information (with regard to mapping between numeric values and text equivalents);

[10] Adobe Systems Incorporated, *TIFF Revision 6.0*, Final, June 3rd 1992. See http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf

[11] http://hul.harvard.edu/jhove/community.html

- More standardised processing of format profiles within modules.

Conclusion and future activities

The joint HUL/JSTOR project concluded in April 2004 with the public beta release of JHOVE, but additional work continued at HUL as part of the Archive and Ingest Handling Test (AIHT) project funded by the Library of Congress under the National Digital Information Infrastructure Preservation Program (NDIIPP). This project ran for one year, during which time additional JHOVE modules were developed to support the JPEG 2000, AIFF, WAVE, and HTML formats. The JHOVE element of the project was staffed by a full-time programmer for the duration.

HUL is currently formulating its long-range plans for JHOVE. The system has been integrated into the DRS loader process and the developers are in the process of completing a JHOVE-based repository submission tool, known as DSIP, that will automatically extract technical metadata from the objects in order to satisfy the DRS requirement for accompanying external metadata. HUL will continue to undertake the routine maintenance of the code base, but the level of effort to be applied in significant enhancements is yet to be determined.[12]

Digital curation is a fundamental aspect of the stewardship of digital assets, within the University and beyond. HUL is confident that JHOVE will continue to play a significant role in the operation of digital repositories and in preservation planning and intervention.

[12] HUL anticipates scheduling periodic maintenance releases to correct errors detected during internal use or reported by external users. Since JHOVE is used internally at HUL, it is expected that the current set of modules will continue to be updated to reflect newer versions of the underlying formats. Should the in-house maintenance of JHOVE cease to be feasible, HUL will consider moving JHOVE to a distributed, collaborative development environment such as SourceForge.

Annex I – List of formats and profiles

The current JHOVE distribution (1.0) supports the following formats and profiles:

a. AIFF
   1. AIFF 1.3
   2. AIFF-C

b. ASCII

c. GIF
   1. GIF 87a
   2. GIF 89a

d. HTML
   1. HTML 3.2, 4.0, and 4.1
   2. XHTML 1.0 and 1.1

e. JPEG
   1. ISO/IEC 10918-1
   2. JFIF (JPEG File Interchange Format)
   3. Exif 2.0, 2.1 (JEIDA-49-1998) and 2.2 (JEITA CP-3451)
   4. SPIFF (Still Picture Interchange File Format, ISO/IEC 10918-3)
   5. JTIP (JPEG Tiled Image Pyramid, ISO/IEC 10918-3)
   6. JPEG-LS (ISO/IEC 14495)

f. JPEG 2000
   1. JP2 (ISO/IEC 15444-1; ITU-T Rec. T.800)
   2. JPX (ISO/IEC 15444-2) profiles

g. PDF
   1. PDF 1.0 through 1.6
   2. PDF/X
      a. PDF/X-1 (ISO 15930-1)
      b. PDF/X-1a (ISO 15930-4)
      c. PDF/X-2 (ISO 15930-5)
      d. PDF/X-3 (ISO 15930-6)
   3. PDF/A (ISO/DIS 19005-1)
   4. Tagged PDF
   5. Linearized PDF

h. TIFF
   1. TIFF 4.0, 5.0, and 6.0 (including baseline Classes B, G, P, R, and extension Class Y)
   2. TIFF/IT (ISO 12639), including the CT, LW, HC, MP, BP, BL, and FP file types and P1 and P2 conformance levels
   3. TIFF/EP (ISO 12234-2)
   4. Exif 2.0, 2.1 (JEIDA-49-1998), and 2.2 (JEITA CP-3451)
   5. GeoTIFF 1.0
   6. TIFF-FX (RFC 2301), including the C, F, J, L, M, and S profiles
   7. Class F (RFC 2306)
   8. RFC 1314
   9. DNG (Adobe Digital Negative)

i. UTF-8

j. WAVE

    1. PCMWAVEFORMAT, WAVEFORMATEX, WAVEFORMATEXTENSION
    2. BWF (Broadcast Wave Format, ECU N22-1997) version 0 and 1

k. XML 1.0

All of the above are open formats, and most are non-proprietary. To date, JHOVE has not provided support for any closed formats, i.e. those for which a complete specification is not publicly available. Since the specifications for closed formats are generally available only under some sort of non-disclosure agreement, providing the source code for modules for closed formats can be problematic, to say the least. In the future, however, as the range of formats accepted into the DRS expands to include formats for office productivity applications, such as word processing and spreadsheets, it may become necessary to develop modules for closed, proprietary formats. In that case, HUL's intention would be to find an appropriate mechanism to distribute the new modules in a manner that is consistent with potential legal restrictions, possibly by including them in the distribution package only in pre-compiled form.

Annex II – Sample Records

The following sample representation information is for a simple GIF image found on the web site of Harvard University's Arnold Arboretum.[13] This data is provided in two forms, produced by (i) the Text output handler, and (ii) the XML output handler. Text handler output is formatted as simple name/value pairs and appears as follows:

```
% jhove –m gif-hul –h text examples/gif/AA_Banner.gif
Jhove (Rel. 1.1, 2005-08-08)
 Date: 2005-10-03 16:22:14 EDT
 RepresentationInformation: examples/gif/AA_Banner.gif
  ReportingModule: GIF-hul, Rel. 1.2 (2005-01-11)
  LastModified: 2004-02-04 17:57:07 EST
  Size: 28782
  Format: GIF
  Version: 89a
  Status: Well-Formed and valid
  SignatureMatches:
   GIF-hul
  MIMEtype: image/gif
```

[13] http://www.arboretum.harvard.edu/images_nav/nav_bar_aa.gif

```
  Profile: GIF 89a
  GIFMetadata:
   GraphicRenderingBlocks: 1
   Blocks:
    LogicalScreenDescriptor:
     LogicalScreenWidth: 335
     LogicalScreenHeight: 89
     ColorResolution: 8
     BackgroundColorIndex: 255
     PixelAspectRatio: 0
     GlobalColorTableFlag: Global color table follows; background color index
meaningful
     GlobalColorTableSortFlag: Not ordered
     GlobalColorTableSize: 7
    GlobalColorTable: 255, 255, 255, ...
    ImageDescriptor:
     ImageLeftPosition: 0
     ImageTopPosition: 0
     InterlaceFlag: Image is interlaced
     LocalColorTableFlag: No local color table; use global table if available
     LocalColorTableSortFlag: Not ordered
     LocalColorTableSize: 0
     NisoImageMetadata:
      MIMEType: image/gif
      ByteOrder: little-endian
      CompressionScheme: LZW
      ColorSpace: palette color
      Orientation: normal
      ImageWidth: 335
      ImageLength: 89
      BitsPerSample: 8
```

The technical properties of the image are characterised in terms of the draft NISO Z39.87 data dictionary for digital still images.[14] The output produced by the XML handler is functionally equivalent to that of the Text handler, but is defined in terms of a JHOVE-specific schema.[15] The specific Z39.87 elements are displayed using the MIX schema:

```
%jhove –m gif-hul –h xml examples/gif/AA_Banner.gif
<?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="2005-08-08">
 <date>2005-10-03T16:25:24-04:00</date>
 <repInfo uri="examples/gif/AA_Banner.gif">
```

[14] http://www.niso.org/standards/standard_detail.cfm?std_id=731

[15] http://hul.harvard.edu/ois/xml/xsd/jhove/jhove.xsd

```
<reportingModule release="1.2"
                date="2005-01-11">GIF-hul</reportingModule>
<lastModified>2004-02-04T17:57:07-05:00</lastModified>
<size>28782</size>
<format>GIF</format>
<version>89a</version>
<status>Well-Formed and valid</status>
<sigMatch>
<module>GIF-hul</module>
</sigMatch>
<mimeType>image/gif</mimeType>
<profiles>
 <profile>GIF 89a</profile>
</profiles>
<properties>
 <property>
  <name>GIFMetadata</name>
  <values arity="Array" type="Property">
  <property>
   <name>GraphicRenderingBlocks</name>
   <values arity="Scalar" type="Integer">
    <value>1</value>
   </values>
  </property>
  <property>
   <name>Blocks</name>
   <values arity="List" type="Property">
   <property>
    <name>LogicalScreenDescriptor</name>
    <values arity="Array" type="Property">
    <property>
     <name>LogicalScreenWidth</name>
     <values arity="Scalar" type="Integer">
      <value>335</value>
     </values>
    </property>
    <property>
     <name>LogicalScreenHeight</name>
     <values arity="Scalar" type="Integer">
      <value>89</value>
     </values>
    </property>
    <property>
     <name>ColorResolution</name>
     <values arity="Scalar" type="Integer">
      <value>8</value>
     </values>
    </property>
    <property>
     <name>BackgroundColorIndex</name>
     <values arity="Scalar" type="Integer">
      <value>255</value>
     </values>
    </property>
 <property>
  <name>PixelAspectRatio</name>
  <values arity="Scalar" type="Short">
   <value>0</value>
  </values>
 </property>
 <property>
  <name>GlobalColorTableFlag</name>
  <values arity="Scalar" type="String">
   <value>Global color table follows; background color
          index meaningful</value>
  </values>
 </property>
 <property>
  <name>GlobalColorTableSortFlag</name>
  <values arity="Scalar" type="String">
   <value>Not ordered</value>
  </values>
 </property>
 <property>
  <name>GlobalColorTableSize</name>
  <values arity="Scalar" type="Short">
   <value>7</value>
  </values>
 </property>
 </values>
</property>
<property>
 <name>GlobalColorTable</name>
 <values arity="Array" type="Short">
  <value>255</value>
  <value>255</value>
  <value>255</value>
  ...
 </values>
</property>
<property>
 <name>ImageDescriptor</name>
 <values arity="Array" type="Property">
 <property>
  <name>ImageLeftPosition</name>
  <values arity="Scalar" type="Integer">
   <value>0</value>
  </values>
 </property>
 <property>
  <name>ImageTopPosition</name>
  <values arity="Scalar" type="Integer">
   <value>0</value>
  </values>
 </property>
 <property>
  <name>InterlaceFlag</name>
  <values arity="Scalar" type="String">
  <value>Image is interlaced</value>
 </values>
</property>
<property>
 <name>LocalColorTableFlag</name>
 <values arity="Scalar" type="String">
  <value>No local color table; use global table if
        available</value>
 </values>
</property>
<property>
 <name>LocalColorTableSortFlag</name>
 <values arity="Scalar" type="String">
  <value>Not ordered</value>
 </values>
</property>
<property>
 <name>LocalColorTableSize</name>
 <values arity="Scalar" type="Short">
  <value>0</value>
 </values>
</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/gif</mix:MIMEType>
      <mix:ByteOrder>little-endian</mix:ByteOrder>
      <mix:Compression>
       <mix:CompressionScheme>5</mix:CompressionScheme>
      </mix:Compression>
      <mix:PhotometricInterpretation>
       <mix:ColorSpace>3</mix:ColorSpace>
      </mix:PhotometricInterpretation>
     </mix:Format>
     <mix:File>
      <mix:Orientation>1</mix:Orientation>
     </mix:File>
    </mix:BasicImageParameters>
    <mix:ImageCreation>
    </mix:ImageCreation>
     <mix:ImagingPerformanceAssessment>
      <mix:SpatialMetrics>
       <mix:ImageWidth>335</mix:ImageWidth>
       <mix:ImageLength>89</mix:ImageLength>
     </mix:SpatialMetrics>
     <mix:Energetics>
      <mix:BitsPerSample>8</mix:BitsPerSample>
```

```
           </mix:Energetics>
          </mix:ImagingPerformanceAssessment>
         </mix:mix>
        </value>
       </values>
      </property>
     </values>
    </property>
    </values>
   </property>
  </values>
  </property>
  </properties>
 </repInfo>
</jhove>
```

The Audit handler is generally invoked against directories of objects and formats its display as follows:

```
% jhove –h audit examples/gif
<?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="2005-08-08">
 <date>2005-10-03T16:31:32-04:00</date>
 <audit home="/users/stephen/projects/jhove">
  <file mime="image/gif"
        status="valid">examples/gif/AA_Banner.gif</file>
  <file mime="image/gif"
        status="valid">examples/gif/hul-banner.gif</file>
  <file mime="image/gif"
        status="valid">examples/gif/textonly.gif</file>
  <file mime="text/plain; charset=US-ASCII"
        status="valid">examples/gif/README</file>
 </audit>
</jhove>
<!-- Summary by MIME type:
 image/gif: 3 (3,0)
 text/plain; charset=US-ASCII: 1 (1,0)
 Total: 4(4,0)
-->
<!-- Summary by directory:
 /users/stephen/projects/jhove/examples/gif: 4 (4,0) + 0,0
 Total: 4 (4,0) + 0,0
-->
<!-- Elapsed time: 0:00:01 -->
```

The parenthetic numbers in the summary by MIME type and directory are the numbers of *valid* objects versus those that are merely *well-formed*.

Annex 3 - The DSIP loader

The DRS loader accepts Submission Information Packages (SIPs) in the form of a set of individual formatted files containing primary content and a single XML-formatted control file containing loader directives and the technical metadata for all of the content files. Prior to the JHOVE integration, all SIP files were validated using a simple magic number based approach. Now, they are truly validated using JHOVE to parse all of the significant syntactic and semantic structures within the files. Additionally, technical metadata about the content files are automatically extracted and compared for consistency with the external metadata supplied in the XML control file. Errors uncovered in this process are relayed to the objects' owners or their depositing agents for correction. The DRS currently accepts deposits only from a known set of contributors, who generate their digital objects using known workflows and technical specifications. It is thus possible for invalid objects to be rejected, since corrected versions can be resubmitted by the responsible parties. In the future, as the scope of the DRS expands to accept material of unknown provenance, object format errors uncovered by JHOVE will not block the acceptance of those objects into the repository; instead, the object metadata will indicate the nature of the format validation or metadata consistency errors.

The intent behind the DSIP tool is to minimise the potential for consistency errors between the internal properties of digital objects and the technical metadata about those objects supplied in the XML control file of the DRS SIP. DSIP will automatically generate the control file metadata using JHOVE to extract the relevant technical properties from the objects. The DRS has established minimum requirements for technical preservation metadata for raster still image and audio content. JHOVE is capable of extracting the minimally-required metadata for all of the formats in the two categories that are currently accepted into the DRS: GIF, JPEG, JPEG 2000, and TIFF for images; AIFF and WAVE for audio.

For example, the DRS requires the following technical metadata properties for all images:

BitsPerSample
CompressionScheme
ColorSpace

The DRS can accept the following optional properties:

XSamplingFrequency
YsamplingFrequency
SamplingFrequencyUnit
ImageWidth
ImageLength
Orientation
ScannerManufacturer
ScannerModelName
ScannerModelNumber
ScannerModelSerialNo
DigitalCameraManufacturer
DigtialCameraModel
ScanningSoftware
ScanningSoftwareVersionNo
ImageProducer
ProcessingSoftwareName
ProcessingSoftwareVersion

JHOVE, and therefore DSIP, can extract all of the mandatory properties, and all of the relevant optional properties, from the set of image formats currently supported by the DRS: GIF, JPEG, JPEG 2000, and TIFF. Note, however, that not all of these formats are able to provide all of these properties: the GIF format, for example, does not provide a means to record the image sampling frequency. Obviously, JHOVE cannot extract properties that do not exist within the file. Through the use of configuration files, however, DSIP can provide these missing metadata properties. DSIP is implemented as a customised JHOVE output handler that extends the Audit handler available as part of the standard JHOVE distribution package.