“What? So what?”
JHOVE2 Next-Generation Characterization

JHOVE2 Project Team
California Digital Library
Portico
Stanford University
Agenda

8:00  Continental breakfast
8:30  Welcome and introductions
8:35  Review agenda and outcomes
8:40  Characterization
8:55  JHOVE2 project
9:15  Demonstration
9:40  Tea/coffee break
10:00 Integration
10:45 Module development
11:30 Questions/discussion
12:30 Lunch
Outcomes

Provide an understanding of

- Role of characterization in preservation and curation activities
- Purpose and deliverables of the JHOVE2 project
- New JHOVE2 architecture, framework, and application
- Integration and use of JHOVE2 technology in preservation and curation systems, services, and workflows
- Development of conforming JHOVE2 modules
Characterization

- 8:00  Continental breakfast
- 8:30  Welcome and introductions
- 8:35  Agenda and outcomes
- 8:40  Characterization
- 8:55  JHOVE2 project
- 9:15  Demonstration
- 9:40  Tea/coffee break
- 10:00 Integration
- 10:45 Module development
- 11:30 Questions/discussion
- 12:30 Lunch
The preservation problem

Managing the gap between what you were given and what you need

– That gap is only manageable if it is quantifiable
– Characterization tells you what you have, as a stable starting point for iterative preservation planning and action

The preservation problem

Less than a third of respondents in a recent Planets survey felt they had control over the content they were being asked to manage.

Planets, Survey Analysis Report, IST-2006-033789, DT11-D1, 2009-05-06
www.planets-project.eu/market-survey/reports/

– How do you know what you have?
– How can you verify that you received what you expected?
– How can you classify for analysis, planning, and workflow
“Tell me about yourself…”
Characterization

Automated determination of the properties of an examined digital object, and the implications of those properties

– Identification  What is it?
– Feature extraction  What about it?
– Validation  What is it, really?
– Assessment  So what?
Characterization

Automated determination of the properties of an examined digital object, and the implications of those properties

– Identification
– Feature extraction
– Validation
– Assessment

{ What?

So what?}
Characterization

Identification

- Determination of the *presumptive* format of a digital object on the basis of suggestive extrinsic hints and intrinsic signatures, both internal and external

Feature extraction

- Reporting the *intrinsic properties* of a digital object significant for classification, analysis, and planning

Validation vs. assessment
“We report, you decide…”

© Fox News Network LLC
Validation vs. assessment

Validation is the determination of the level of conformance to the normative requirements of a format’s authoritative specification

- To the extent that there is community consensus on these requirements, validation is an objective determination

Assessment is the determination of the level of acceptability for a specific purpose on the basis of locally-defined policy rules

- Since these rules are locally configurable, assessment is a subjective determination
Characterization in ingest workflows
Characterization in migration workflows
JHOVE2 project

- 8:00 Continental breakfast
- 8:30 Welcome and introductions
- 8:35 Agenda and outcomes
- 8:40 Characterization
- 8:55 JHOVE2 project
- 9:15 Demonstration
- 9:40 Tea/coffee break
- 10:00 Integration
- 10:45 Module development
- 11:30 Questions/discussion
- 12:30 Lunch

- Goals
- Features
- Implementation
- Schedule
- Project team
- Advisory board
- Community
- Format support
- New Concepts
  - Properties
  - Reportables
  - Identifiers
  - Source units
  - Modules
  - Strategies
  - Assessment
JHOVE2 is …

… a project to develop a next-generation open source framework and application for format-aware characterization

… a collaborative initiative of CDL, Portico, and Stanford

… a two year grant from the Library of Congress as part of its NDIIPP initiative
Project goals

Address recognized deficiencies of design and implementation in JHOVE1

- API complexity and idiosyncrasy
- Internationalization
- Performance

Provide enhancements to JHOVE1 functionality

- Signature-based identification
- Recursive processing of formatted byte streams arbitrarily nested within files
- Support for aggregate objects spanning multiple files
- Support for rules-based assessment
**Features**

Multi-stage processing
- Signature-based identification (*atomistic and aggregate*)
- Feature extraction
- Validation
- Message digesting
- Rules-based assessment

Flexible configuration
- Dependency injection

Granular modularization

Generic plug-ins

Increased performance through buffered I/O

Standardized profile and error handling

Internationalized output

Recursive processing of aggregate and arbitrarily-nested objects

Results transformable to arbitrary final form
Implementation

Java 1.6 J2SE
java.sun.com/javase/6/docs/api/

- Annotations
  java.sun.com/javase/6/docs/technotes/guides/language/annotations.html

- Buffed I/O (java.nio)
  java.sun.com/javase/6/docs/api/java/nio/package-summary.html

- Reflection
  java.sun.com/docs/books/tutorial/reflect

Spring dependency injection framework
www.springframework.org

Maven build management
maven.apache.org

Hudson continuous integration testing
hudson.dev.java.net
Implementation

Core framework is a collaborative effort

Modules implemented independently by project partners

Early prototyping, extensive refactoring

– 5 working versions “thrown away” so far
Schedule

6 months  ✓ Stakeholder engagement, needs assessment, functional requirements
6 months  ✓ Prototyping, refactoring, core framework
12 months Modules, documentation
Project team

California Digital Library
- Stephen Abrams
- Patricia Cruse
- John Kunze
- Marisa Strong
- Perry Willett

Stanford University
- Richard Anderson
- Tom Cramer
- Hannah Frost

Portico
- John Meyer
- Sheila Morrissey
- Evan Owens

With help from
- Walter Henry
- Nancy Hoebelheinrich
- Keith Johnson
- Justin Littman
Advisory Board

Deutsche Nationalbibliothek (DNB)
Ex Libris
Fedora Commons / Rutgers University
Florida Center for Library Automation (FCLA)
Harvard University / GDFR project
Koninklijke Bibliotheek (KB)
Library of Congress
MIT / DSpace
NARA
National Library of Australia (NLA)
National Library of New Zealand (NLNZ)
Planets project / Universität Köln
Community

Wiki

- [http://confluence.ucop.edu/display/JHOVE2Info/Home](http://confluence.ucop.edu/display/JHOVE2Info/Home)

Mailing lists

- [JHOVE2-Announce-L@listserv.ucop.edu](mailto:JHOVE2-Announce-L@listserv.ucop.edu)
- [JHOVE2-Techtalk-L@listserve.ucop.edu](mailto:JHOVE2-Techtalk-L@listserve.ucop.edu)
“Well, there’s good news…”

AIFF
ASCII
GIF
HTML
JPEG
JPEG 2000
PDF
TIFF
UTF-8
WAVE
XML
“Well, there’s good news…”

AIFF
ASCII
dBase
GIF
HTML
ICC
JPEG
JPEG 2000
PDF
SGML
Shapefile
TIFF
UTF-8
WAVE
XML
Zip

JP2, JPX
1.0 – 1.7, ISO 32000, PDF/A, PDF/X
4.0 – 6.0, Class B, F, G, P, R, Y, TIFF-FX, TIFF/EP, TIFF/IT, GeoTiff, DNG
BWF
“… and there’s bad news”
“… but wait, there’s more good news”

Discussions are underway with a number of institutions about 3rd party development and co-development opportunities.

This should be facilitated by

- Streamlined APIs
- Common module design patterns
- Increased modularization
- More comprehensive documentation and tutorials
Properties and reportables

A property is a named, typed value
- Name
- Unique formal identifier
- Data type
  - Scalar or collection
  - Java types, JHOVE2 primitive types, or JHOVE2 reportables
- Typed value
- Description of correct semantic interpretation

A reportable is a named set of properties
- Reportables correspond to Java classes
- Properties correspond to fields
Identifiers

All formats, reportables, and properties are assigned a unique identifier in the JHOVE2 namespace

- “info” scheme URI

  info:jhove2/<type>/<name>

  info:jhove2/format/utf-8
  info:jhove2/reportable/org/jhove2/core/Product
  info:jhove2/property/org/jhove2/core/Product/Note
  info:jhove2/message/

- Property names are based on the terminology of the underlying format
Source units

A formatted object about which characterization information can be meaningfully reported

- File  e.g. TIFF
- File inside of a container  e.g. TIFF inside a Zip
- Byte stream inside a file  e.g. ICC inside a TIFF
- Directory
- Directory inside of a container
- File set
- Clump  e.g. Shapefile

For purposes of characterization, directories, file sets, and clumps are considered formats
Modules

• Application
  – Framework
    • Identification
    • Aggregation
      (“aggregate identification”)
    • Parsing / feature extraction / validation
    • Message digesting
      – Digesting algorithms
    • Assessment
  – Display

JHOVE2CommandLine

JHOVE2

IdentifierModule

AggregferModule

Format modules and profiles

DigesterModule

Adler32Digester, CRC32Digester, …

AssessmentModule

JSONDisplayer

TextDisplayer

XMLDisplayer
Modules

Framework

- Encapsulates all JHOVE2 function
- Embodies a particular characterization *strategy* as a sequence of configured modules

Displayer

- Produces human-readable results

  JSON, Text, XML

  ✓ Text format uses simple name/value pairs
  ✓ XML is an intermediate form that can be transformed via a stylesheet to a desired final form
Characterization strategy

1. Identify format
2. Dispatch to appropriate format module
   a) Extract format features and validate
      – If a nested source unit is found, process recursively…
   b) Validate format profiles (if registered)
3. Assess
4. If unitary source unit, calculate message digests (optional)
5. If an aggregate source unit, try to identify aggregate format, and if successful, process recursively…
Characterization strategy

directory/

- abc.shp
- abc.shx
- abc.dbf
- abc.tif
- xyz.pdf
Characterization strategy

- abc.shp
  - Main
- abc.shx
  - Index
- abc.dbf
  - dBASE
- abc.tif
  - GeoTIFF
- xyz.pdf
  - PDF
Characterization strategy
Characterization strategy
Profiles

A profile is a specialized module that examines prior characterization information and recognizes known format subtypes

– All registered profiles are automatically invoked as the terminal step of module processing

Profiles can also be dealt with through specific assessment rule sets
Assessment

The evaluation of prior characterization information relative to local policy

– Facilitates the analysis of object metadata in order to manage the object locally more effectively

Result of assessment can inform a decision-making process

– Determine level of risk
– Assign level of service
– Take action now or later
Practical applications

Assessment has practical applications in:

- Ingest workflows
- Migration workflows
- Digitization workflows
- Publishing workflows

It can be easily extended to build tools capable of more complex analyses:

- Weighted scoring system
- “Institutional technology profiles”
Assessment rules

Assertions whose terms are logical expressions based on prior characterization properties

- Presence/absence of a property
- Constraints on property values
- Combinations of properties/values

The evaluation of the assertion results in new characterization properties.

- Custom metadata that has significance in a local context
Rule configuration

Must be easy for technical and non-technical alike

Rules can be atomic or chained

Basic formation of a rule:

- **<property>**
  - Is Equal To
  - Is Not Equal To
  - Is Greater Than
  - Is Less Than
  - Contains
  - Does Not Contain

- **<value>**

Plus

- **<response if true>**
- **<response if false>**
Assessment examples

PDF

Assertion  Message [Error], Contains, IllformedDate
Result     True
Response if true  Acceptable

TIFF

Assertion  Message [Information], Contains, Non-wordAlignedOffset
Result     True
Response if true  Acceptable
Assessment examples

WAVE

Assertion$_1$ isValid, isEqualTo, Valid
Assertion$_2$ BitDepth, isEqualTo, 24
Assertion$_3$ SamplingFrequency, isEqualTo, 96000
Result       False
Response if false   Unacceptable
**Demonstration**

- 8:00  *Continental breakfast*
- 8:30  Welcome and introductions
- 8:35  Agenda and outcomes
- 8:40  Characterization
- 8:55  JHOVE2 project
- 9:15  Demonstration
- 9:40  *Tea/coffee break*
- 10:00 Integration
- 10:45 Module development
- 11:30 Questions/discussion
- 12:30 *Lunch*
Tea/coffee break

- 8:00  Continental breakfast
- 8:30  Welcome and introductions
- 8:35  Agenda and outcomes
- 8:40  Characterization
- 8:55  JHOVE2 project
- 9:15  Demonstration
- 9:40  Tea/coffee break
- 10:00 Integration
- 10:45 Module development
- 11:30 Questions/discussion
- 12:30 Lunch
Agenda

✓ 8:00  Continental breakfast
✓ 8:30  Welcome and introductions
✓ 8:35  Agenda and outcomes
✓ 8:40  Characterization
✓ 8:55  JHOVE2 project
✓ 9:15  Demonstration
✓ 9:40  Tea/coffee break

10:00  Integration
10:45  Module development
11:30  Questions/discussion

12:30  Lunch

•  Installation
•  API
•  Configuration
•  Invocation
•  Results
Installation

jhove2/
  src/
    main/
      java/
        org/
          jhove2/
            annotation/
            app/
            core/
            module/
            resources/
              config/
                jhove2-config.xml
              properties/
                unicode/
                  c0control.properties
                  c1control.properties
                  codeblock.properties
                  dispatcher.properties
                  displayr.properties
API design idioms

Inversion of control (IOC) / dependency injection

– Martin Fowler
  martinfowler.com/articles/injection.html

– Spring framework
  www.springsource.org/

Separation of concerns

– Annotation and reflection
  confluence.ucop.edu/display/JHOVE2Info/Background+Papers
Dependency injection

All JHOVE2 function is embodied in pluggable modules

- Flexible customization
  - Re-sequencing of pre-existing modules

- Easy extensibility
  - Additional format modules and profiles
  - Additional aggregate identifiers
  - Additional displayers
  - New behaviors
    - RenderabilityModule
    - VirusCheckModule
Separation of concerns

• “Let POJOs be POJOs”
  – Focus on modeling the format itself

• “Let the code write itself”
  – Reportables “know” how to expose their properties for display
  – Reference documentation generated from the code

✓ JHOVE2Doc application

Reportable: Name: UTF8Module
  Identifier: [JHOVE2] info:jhove2/reportable/org/jhove2/module/utf8
  Package: org.jhove2.module.format.utf8
From: Class UTF8Module
  Property: Name: NumCharacters
  Identifier: [JHOVE2] info:jhove2/property/org/jhove2/module/format
  Type: long
  Description: Number of UTF-8 characters
Reportable properties

Each reportable property is represented by a field and accessor and mutator methods.

The accessor method must be marked with the @ReportableProperty annotation.

```java
public class MyReportable implements Reportable {
    protected String myProperty;

    @ReportableProperty(order=1, desc="description", ref="reference")
    public String getMyProperty() {
        return this.myProperty;
    }

    public void setMyProperty(String property) {
        this.myProperty = property;
    }
}
```
JHOVE2 framework

Embodiment of a characterization strategy as a configurable sequence of modules

```java
public void characterize(Source source) throws IOException, JHOVE2Exception {
    source.getTimerInfo().setStartTime();
    source.setDeleteTempFiles(
        this.getAppConfigInfo().getDeleteTempFiles());
    /* Update summary counts of source units, by type. */
    this.sourceCounter.incrementSourceCounter(source);
    try {
        for (JHOVE2Command command : this.commands) {
            command.execute(source, this);
        }
    } finally {
        source.close();
    }
    source.getTimerInfo().setEndTime();
}
```
JHOVE2 framework
Characterization
Identification
Feature extraction
Aggregate identification and recursive characterization

```
for each ClumpSource
  addChildSource(ClumpSource)
  characterize(ClumpSource)
```

JHOVE2 -> Source -> AggregatorCommand -> AggregateIdentifier -> ClumpSource

execute(Source)

identify(Source) -> Set<ClumpSource>
Spring configuration: Identification

<!-- Identifier module bean -->
<bean id="Identifier" class="org.jhove2.module.identify.IdentifierModule"
    scope="prototype">
    <property name="developers">
        <list value-type="org.jhove2.core.Agent">
            <ref bean="CDLAgent"/>
            <ref bean="PorticoAgent"/>
            <ref bean="StanfordAgent"/>
        </list>
    </property>
    <property name="fileSourceIdentifier" ref="droidIdentifier"/>
</bean>

<!-- DROID identifier bean -->
<bean id="droidIdentifier" class="org.jhove2.module.identify.DroidIdentifier"
    scope="prototype">
    <property name="developers">
        <list value-type="org.jhove2.core.Agent">
            <ref bean="CDLAgent"/>
            <ref bean="PorticoAgent"/>
            <ref bean="StanfordAgent"/>
        </list>
    </property>
    <property name="configFilePath" ref="droidConfigFilePath"/>
    <property name="sigFilePath" ref="droidSigFilePath"/>
</bean>
Spring configuration: Identification

```xml
<bean id="Identifier" class="org.jhove2.module.identify.IdentifierModule"
    scope="prototype">
    <property name="developers">
        <list value-type="org.jhove2.core.Agent">
            <ref bean="CDLAgent"/>
            <ref bean="PorticoAgent"/>
            <ref bean="StanfordAgent"/>
        </list>
    </property>
    <property name="fileSourceIdentifier" ref="bsdIdentifier"/>
</bean>

<bean id="bsdIdentifier" class="org.myinstitution.identify.BsdFileIdentifier"
    scope="prototype">
    <property name="developers">
        <list value-type="org.jhove2.core.Agent">
            <ref bean="MYINSTITUTIONAGENT"/>
        </list>
    </property>
    <property name="runtimepath" ref="bsdFileRuntimePath"/>
</bean>
```
Spring configuration: Aggregation

```xml
<!-- Aggrefier module bean -->
<bean id="Aggrefier"
     class="org.jhove2.module.identify.AggrefierModule"
     scope="singleton">
  <property name="developers">
    <list value-type="org.jhove2.core.Agent">
      <ref bean="CDLAgent"/>
      <ref bean="PorticoAgent"/>
      <ref bean="StanfordAgent"/>
    </list>
  </property>
  <property name="recognizers">
    <list value-type="org.jhove2.module.identify.AggregateIdentifier">
      <ref bean="ShapeFileRecognizer"/>
    </list>
  </property>
</bean>
```
Spring configuration: Aggrefication

```xml
<!-- Aggrefier module bean -->
<bean id="Aggrefier"
    class="org.jhove2.module.identify.AggregrefierModule"
    scope="singleton">
    <property name="developers">
        <list value-type="org.jhove2.core.Agent">
            <ref bean="CDLAgent"/>
            <ref bean="PorticoAgent"/>
            <ref bean="StanfordAgent"/>
        </list>
    </property>
    <property name="recognizers">
        <list value-type="org.jhove2.module.identify.AggregateIdentifier">
            <ref bean="ShapeFileRecognizer"/>
            <ref bean="GisObjectRecognizer"/>
            <ref bean="DocBookRecognizer"/>
        </list>
    </property>
</bean>
```
Dispatch map

jhove2/src/main/resources/properties/dispatch.properties

<table>
<thead>
<tr>
<th>&lt;format-identifier&gt;</th>
<th>&lt;spring-bean-name&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>info:jhove2/format/jpeg2000</td>
<td>JPEG2000Module</td>
</tr>
<tr>
<td>info:jhove2/format/pdf</td>
<td>PDFModule</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Displayer directives

jhove2/src/main/resources/properties/displayer.properties

<property-identifier> <directive>
info\:jhove2\:property\:org\:jhove2\:core\:Agent\:Note Never
info\:jhove2\:property\:\:\:\:\:\:\:\:\:\:\:\:\:/DirectorySource\:isExtant IfFalse...

- Always (default)
- IfTrue
- IfNegative
- IfPositive
- IfZero
- Never
- IfFalse
- IfNonNegative
- IfNonPositive
- IfNonZero
**DROID-to-JHOVE2 map**

jhove2/src/main/resources/properties/droid2jhove.properties

<table>
<thead>
<tr>
<th>&lt;droid-identifier&gt;</th>
<th>&lt;jhove2-identifier&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>fmt/14</td>
<td>info:jhove2/format/pdf</td>
</tr>
<tr>
<td>fmt/392</td>
<td>info:jhove2/format/jpeg2000</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Unicode controls and code blocks

jhove2/src/main/resources/properties/unicode/c0control.properties

jhove2/src/main/resources/properties/unicode/c1control.properties

<table>
<thead>
<tr>
<th>mnemonic</th>
<th>code-point</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL</td>
<td>00</td>
</tr>
<tr>
<td>APC</td>
<td>9F</td>
</tr>
</tbody>
</table>

jhove2/src/main/resources/properties/unicode/codeblocks.properties

<table>
<thead>
<tr>
<th>code-point</th>
<th>code-point; block</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000..0x007f;</td>
<td>Basic Latin</td>
</tr>
<tr>
<td>0x0080..0x00ff;</td>
<td>Latin-1 Supplement</td>
</tr>
</tbody>
</table>

(identical format to Unicode database (UCD) file [www.unicode.org/Public/UNIDATA/Blocks.txt](http://www.unicode.org/Public/UNIDATA/Blocks.txt))
Command line invocation

% jhove2 [-ik] [-b size]
[ -B Direct|NonDirect|Mapped]
[ -d JSON|Text|XML] [-f limit]
[ -t temp] [-o file] file ...

-i               Show identifiers in JSON and Text displayers
-k               Calculate message digests
-b size           I/O buffer size, in bytes                   (default: 131072)
-B type           I/O buffer type: Direct, NonDirect, Mapped  (default: Direct)
-d display       Displayer: JSON, Text, XML                  (default: Text)
-f limit          Fail fast limit                            (default: 0; no limit)
-t temp           Temporary directory
-o file           Output file                                (default: standard output)
file              File, directory, or URI source unit
package org.myinstitution.workflow;

import java.io.File;
import org.jhove2.core.JHOVE2;
import org.jhove2.core.config.Configure;
import org.jhove2.core.source.Source;
import org.jhove2.core.source.SourceFactory;
import org.jhove2.module.display.Displayer;

/** Class which invokes JHOVE2 to characterize an object *
 */
public class DigitalObjectCharacterizer {
    public enum Status {
        SUCCEED,
        FAIL
    }

    /** Performs JHOVE2 characterization on a file *
     * @param inputFile File object to be characterized *
     * @param outputFilePath Path for (XML) results of characterization *
     * @return Status indicating success or failure *
     */
    public Status characterizeFile(File inputFile, String outputFilePath) {
        JHOVE2 framework = null;
        Source source = null;
        Displayer displayer = null;
        Status status = Status.SUCCEED;
        try {
            framework = Configure.getReportable(JHOVE2.class, "JHOVE2"); // create framework object
            source = SourceFactory.getSource(inputFile); // create JHOVE2 Source object
            source.addModule(framework); // attach framework to Source
            framework.getTimerInfo().setStartTime(); // start the clock
            framework.characterize(source); // characterize the file
            framework.getTimerInfo().setEndTime(); // stop the clock
            displayer = Configure.getReportable(Displayer.class, "XML"); // create XML output handler
            displayer.setOutputFilePath(outputFilePath); // configure the XML handler
            displayer.display(source); // serialize characterization results as XML
        } catch (Exception e) {
            // my workflow exception handler behavior here
            status = Status.FAIL;
        }
        return status;
    }
}
Results

**JSON**

```json
"Path": "C:\\shapefiles"
```

**Text**

`Path: C:\shapefiles`

**XML**

```xml
<j2:feature name="Path"
    fid="info:jhove2/property/org/jhove2/core/source/DirectorySource/Path"
    fidns="JHOVE2">
  <j2:value>C:\shapefiles</j2:value>
</j2:feature>
```

- Stylesheets for transforming to JHOVE1, METS, MIX, PREMIS, ...
Results

JHOVE2 processing results in a hierarchical tree of Source units, each associated with the modules (and their results) that processed the units.

- Subsidiary source units, modules, and their individual properties can be interrogated.

```java
public interface Source
    extends Reportable
{
    public List<Source> getChildSources();
    public List<Module> getModules();
}
```
Messages

- Messages are themselves reportable properties
  - Unique identifier
    
    info:jhove2/message/org/jhove2/module/format/utf8/UTF8Module/ByteOrderMark
  - Context
    - Process  Condition arising from the process of characterization
    - Object   Condition arising in the object being characterized
  - Severity
    - Error
    - Warning
    - Info
  - Internationalizable
UTF8Module.java

```java
if (position == start && ch.isByteOrderMark()) {
    Object [] messageParms = new Object [] {position};
    this.bomMessage = new Message(Severity.INFO,
                                          Context.OBJECT,
                                          "org.jhove2.module.format.utf8.UTF8Module.bomMessage",
                                          messageParms);
}
```

jhove2-config.xml

```xml
<bean id="messageSource" class="springframework.context.support.ResourceBundleMessageSource">
    <property name="basename">
        <value>properties.messages</value>
    </property>
</bean>
```
Messages

messages.properties

# Message templates for class for
# org.jhove2.module.format.utf8.UTF8Module

org.jhove2.module.format.utf8.UTF8Module.failFastMessage=Fail fast limit exceeded; additional errors may exist but will not be reported

org.jhove2.module.format.utf8.UTF8Module.bomMessage=Byte Order Mark (BOM) at byte offset {0, number, integer}

XML results

<j2:feature name="ByteOrderMark" fid="info:jhove2/message/org/jhove2/module/format/utf8/UTF8Module/ByteOrderMark"
   fidns="JHOVE2">
   <j2:value>[INFO/OBJECT] Byte Order Mark (BOM) at byte offset 333,333</j2:value>
</j2:feature>
Module development

✓ 8:00  Continental breakfast
✓ 8:30  Welcome and introductions
✓ 8:35  Agenda and outcomes
✓ 8:40  Characterization
✓ 8:55  JHOVE2 project
✓ 9:15  Demonstration
✓ 9:40  Tea/coffee break
✓ 10:00 Integration
  10:45  Module development
  11:30  Questions/discussion
12:30  Lunch

- Format information
- Reportables and properties
- Interfaces
- Process
Module development

Module specification document

Implement the Java classes
  – Package namespace
  – Javadoc
  – Annotations

Modify configuration files

Review conformance with JHOVE2 interfaces

Arrange for distribution of the module
  – License
Module specification

Introduction
Identification
References
Terminology and conventions
Validity
Format profiles
Reportable properties
Configuration
Implementation Notes
Format information

Names
  – Canonical and aliases

Identifiers
  – Canonical (in the JHOVE2 namespace) and aliases

Specification documents
  – Authoritative, informative, and speculative

Normative syntax and semantics
Format transparency

A format is considered *unambiguous* if there is broad community consensus regarding the intention of *all* normative requirements of the format’s authoritative specification.

Otherwise it is considered *ambiguous*, and areas of potential ambiguity must be documented.
Module completeness

A module is considered comprehensive if all normative requirements associated with its format’s authoritative specification are validated.

Otherwise it is considered selective, and non-validated features must be documented.
Reportables and properties

Define reportables for the major conceptual structures inherent to the format

- JPEG 2000
  Box
- TIFF
  IFH, IFD
- UTF-8
  Character stream, character
- WAVE
  Chunk
Reportables and properties

A reportable implements the Reportable marker interface

```java
package org.jhove2.core

public interface Reportable {
    public I8R getReportableIdentifier();
    public String getReportableName();
    public void setReportableName(String name);
}

public abstract class AbstractReportable implements Reportable {
    protected I8R reportableIdentifier;
    protected String reportableName;
}
```
Each reportable property is represented by a field and accessor and mutator methods

The accessor method *must* be marked with the @ReportableProperty annotation

```java
public class MyReportable
    implements Reportable
{
    protected String myProperty;

    @ReportableProperty(order=1, desc="description", ref="reference")
    public String getMyProperty() {
        return this.myProperty;
    }

    public void setMyProperty(String property) {
        this.myProperty = property;
    }
}
```
import org.jhove2.module;

public interface Module
    extends Reportable
{
    public List<Agent> getDevelopers();
    public String getNote();
    public String getReleaseDate();
    public String getRightsStatement();
    public TimerInfo getTimerInfo();
    public String getVersion();
    public WrappedProductInfo getWrappedProduct();
}

public abstract class AbstractModule
    implements Module
{
    public AbstractModule(String version, String release,
        String rights
    }
}
import org.jhove2.core;

public interface JHOVE2Command extends Module {
    public void execute(JHOVE2 jhove2, Source source) throws JHOVE2Exception;
}
Identifier interface

For *atomistic* identification modules

```java
import org.jhove2.module.identify;

public interface Identifier
    extends Module
{
    public Set<FormatIdentification> identify(JHOVE2 jhove2, Source source)
        throws JHOVE2Exception;
}
```
Aggregfier interface

For aggregate identification modules

```java
import org.jhove2.module.identify;

public interface AggregateIdentifier
    extends Module
{
    public Set<ClumpSource> identify(JHOVE2 jhove2,
                                        AggregateSource source)
        throws IOException, JHOVE2Exception;
}
```
Digester interface

For digester modules

```java
import org.jhove2.module.digest;

public interface Digester
    extends Module
{
    public void digest(JHOVE2 jhove2, Source source)
        throws IOException;

    public Set<Digest> getDigests();
}
```
Digester algorithm interfaces

For digester algorithms

```java
import org.jhove2.module.digest;

public interface DigestAlgorithm
    extends Reportable
{
    public Digest getDigest();
}
public interface ArrayDigester
    extends DigestAlgorithm
{
    update void update(byte [] array);
}
public interface BufferDigester
    extends DigestAlgorithm
{
    void update(ByteBuffer buffer);
}
```
import org.jhove2.module.format;

public interface FormatModule
  extends Module
{
  public Format getFormat();
  public List<FormatProfile> getProfiles();
  public long parse(JHOVE2 jhove2, Source source)
      throws IOException, JHOVE2Exception
}

public class BaseFormatModuleCommand
  extends AbstractModule
  implements FormatModule
{
  public BaseFormatModuleCommand(String version, String release,
      String rights, Format format);
}
import org.jhove2.module.format;

public interface FormatProfile
    extends Module
{
    public Format getFormat();
}

public AbstractFormatProfile
    extends AbstractModule
    implements FormatProfile
{
    public AbstractFormatProfile(String version, String release, String rights, Format format);
}
Validator interface

```java
import org.jhove2.module.format;

public interface Validator {
    public enum Coverage {
        Exhaustive, Selective, None
    }
    public enum Validity {
        True, False, Undetermined
    }

    public Validity validate(JHOVE2 jhove2, Source source);
    public Coverage getCoverage();
    public Validity isValid();
}
```
import org.jhove2.core;

public class Agent
    extends AbstractReportable
{
    public enum Type {
        Corporate, Personal
    }
    public Agent(String name, Type type);

    public String getAddress();
    public Agent getAffiliation();
    public String getEmail();
    public String getFax();
    public String getName();
    public String getNote();
    public String getTelephone();
    public Type getType();
    public String getURI();
}
import org.jhove2.core;

public class Digest
{
    public Digest(String value, String algorithm);

    public String getAlgorithm();
    public String getValue();
    public String toString(); // [algorithm] value
}
import org.jhove2.core;

public class Document
    extends AbstractReportable
{
    public enum Intention {
        Authoritative, Informative, Speculative, Other, Unknown
    }
    public enum Type {
        Article, Codebook, ..., Other
    }
    public Document(String title, Type type, Intention intention);
    public String getAuthor();
    public String getDate();
    public String getEdition();
    public List<I8R> getIdentifiers();
    public Intention getIntention();
    public String getNote();
    public String getPublisher();
    public Type getType();
}
import org.jhove2.core;

public class Format
    extends AbstractReportable
{
    public enum Ambiguity {
        Ambiguous, Unambiguous
    }
    public enum Type {
        Family, Format
    }
    public Format(String name, I8R identifier, Type type, Ambiguity ambiguity);
    public Set<I8R> getAliasIdentifiers();
    public Set<String> getAliasNames();
    public Ambiguity getAmbiguity();
    public I8R getIdentifier();
    public String getName();
    public List<Document> getSpecifications();
    public Type getType();
}


import org.jhove2.core;

public class FormatIdentification
    extends AbstractReportable
{
    public enum Confidence {
        Negative, Tentative, Heuristic, PositiveGeneric,
        PositiveSpecific, Validated
    }
    public FormatIdentification(I8R jhove2ID, Confidence conf,
                                Ambiguity ambiguity);
    public Confidence getConfidence();
    public I8R        getIdentification();
    public List<Message> getMessages();
}
import org.jhove2.core;

public class I8R {
    public enum Namespace {
        AFNOR, AIIM, ..., JHOVE2, ..., URI, URL, URN, UTI, Other
    }

    public I8R(String value) {
        this(value, Namespace.JHOVE2);
    }
    public I8R(String value, Namespace namespace);

    public Namespace getNamespace();
    public String getValue();
    public String toString();      // [namespace] value
Module identification

Format name
- XML

Alias name
- Extensible Markup Language (XML)

JHOVE2 format identifier
- [JHOVE] info:jhove2/format/xml

Alias identifiers
- [MIME] application/xml,
  [RFC] RFC 3023
  [UTI] public.xml

Module identifier
- info:jhove2/reportable/org/jhove2/module/format/XmlModule

Module package/classname
- org.jhove2.module.format.xml.XmlModule.java
Module class

Create the Java package and class
- org.jhove2.module.format.xml.XmlModule.java

Module-level comments
- copyright statement, redistribution rights, authors, disclaimers

Library imports
- import org.jhove2.annotation.ReportableProperty;
  import org.jhove2.core.*
  import org.jhove2.module.format.*

Class inheritance
- extends BaseFormatModuleCommand
  implements Validator
Standard members

Generic module properties

```java
public static final String VERSION = "0.1.0";
public static final String RELEASE = "2009-09-23";
public static final String RIGHTS = "Copyright 2009 ..."
```

Constructor

```java
public XmlModule(Format format) {
    super(VERSION, RELEASE, RIGHTS, format);
}
```

Validator methods/stubs (if module implements Validator)

```java
public Coverage getCoverage()
public Validity validate(JHOVE2 jhove2, Source source)
public Validity isValid()
```
Reportable property fields

```java
protected String saxParser = "org.apache.xerces.parsers.SAXParser";
protected XmlDeclaration xmlDeclaration = new XmlDeclaration();
protected String xmlRootElementName;
protected List<XmlDTD> xmlDTDs;
protected HashMap<String, XmlNamespace> xmlNamespaceMap;
protected List<XmlNotation> xmlNotations;
protected List<String> xmlCharacterReferences;
protected List<XmlEntity> xmlEntitys;
protected List<XmlProcessingInstruction> xmlProcessingInstructions;
protected List<String> xmlComments;
protected XmlValidationResults xmlValidationResults = new XmlValidationResults();
protected boolean wellFormed = false;
```
Reportable property declarations

```java
@ReportableProperty(order=1, value="Java class used to parse the XML")
public String getSaxParser() {
    return saxParser;
}

@ReportableProperty(order=2, value="XML Declaration data")
public XmlDeclaration getXmlDeclaration() {
    return xmlDeclaration;
}

@ReportableProperty(order=3, value="Name of the document's root element")
public String getXmlRootElementName() {
    return xmlRootElementName;
}
```
public class XmlDeclaration
    implements Reportable
{
    protected String version;
    protected String encoding;
    protected String standalone;

    @ReportableProperty(order=1, value="XML Version")
    public String getVersion() {
        return version;
    }

    @ReportableProperty(order=2, value="Character Encoding")
    public String getEncoding() {
        return encoding;
    }

    @ReportableProperty(order=3, value="Standalone")
    public String getStandalone() {
        return standalone;
    }
}
Parse method

```java
public long parse(JHOVE2 jhove2, Source source) throws EOFException, IOException, JHOVE2Exception {
    XMLReader xmlReader;
    try {
        xmlReader = XMLReaderFactory.createXMLReader(saxParser);
        ...
    } catch (SAXException e) {
        throw new JHOVE2Exception("Could not create parser", e);
    }

    InputSource saxInputSource =
        new InputSource(source.getInputStream());
    try {
        xmlReader.parse(saxInputSource);
    } catch (SAXParseException spe) {
        wellFormed = false;
    } catch (SAXException e) {
        throw new JHOVE2Exception("Could not parse ..", e);
    }

    return 0;
}
```
Other Considerations

Validation

- The “validate” method of the Validator interface will be automatically called by the execute method of BaseFormatModuleCommand

Exception Handling

- Input data problem (e.g. mal-formed XML) should not kill the application

Test Code and Test Files

Javadoc
Configuration files

config/jhove2-config.xml
  - Add <bean> elements to Spring configuration file

properties/droid2jhove.prop
  - Mapping from DROID PUID identifiers for formats to JHOVE2 unique identifiers for formats

properties/format2bean.properties
  - Mapping from unique identifiers to Spring bean names for the format associated with the formats

properties/dispatcher.properties
  - Mapping from unique identifiers to Spring bean names for the modules associated with the formats
<bean id="XmlModule"
    class="org.jhove2.module.format.xml.XmlModule"
    scope="prototype">
    <constructor-arg ref="XmlFormat"/>
    <property name="developers">
        <list value-type="org.jhove2.core.Agent">
            <ref bean="StanfordAgent"/>
        </list>
    </property>
</bean>
<!-- XML format bean -->
<bean id="XmlFormat" class="org.jhove2.core.Format" scope="singleton">
   <constructor-arg type="java.lang.String" value="XML"/>
   <constructor-arg ref="XmlIdentifier"/>
   <constructor-arg type="org.jhove2.core.Format$Type" value="Format"/>
   <constructor-arg type="org.jhove2.core.Format$Ambiguity" value="Unambiguous"/>
   <property name="aliasIdentifiers">
      <set value-type="org.jhove2.core.I8R">
         <ref bean="XmlMIMETYPE"/>
         <ref bean="XmlRFC3023"/>
         <ref bean="XmlUTI"/>
      </set>
   </property>
   <property name="aliasNames">
      <set>
         <value>Extensible Markup Language (XML)</value>
      </set>
   </property>
   <property name="specifications">
      <list value-type="org.jhove2.core.Document">
         <ref bean="XML10Specification"/>
         <ref bean="XML11Specification"/>
      </list>
   </property>
</bean>
<!-- XML identifier bean -->
<bean id="XmlIdentifier" class="org.jhove2.core.I8R" scope="singleton">
   <constructor-arg type="java.lang.String" value="info:jhove2/format/xml"/>
</bean>

<!-- XML MIME type bean -->
<bean id="XmlMIMEType" class="org.jhove2.core.I8R" scope="singleton">
   <constructor-arg type="java.lang.String" value="application/xml"/>
   <constructor-arg type="org.jhove2.core.I8R$Namespace" value="MIME"/>
</bean>

<!-- XML RFC 3023 bean -->
<bean id="XmlRFC3023" class="org.jhove2.core.I8R" scope="singleton">
   <constructor-arg type="java.lang.String" value="RFC 3023"/>
   <constructor-arg type="org.jhove2.core.I8R$Namespace" value="RFC"/>
</bean>

<!-- XML UTI bean -->
<bean id="XmlUTI" class="org.jhove2.core.I8R" scope="singleton">
   <constructor-arg type="java.lang.String" value="public.xml"/>
   <constructor-arg type="org.jhove2.core.I8R$Namespace" value="UTI"/>
</bean>
Properties files

properties/droid2jhove.prop

fmt/101 info:jhove2/format/xml

properties/format2bean.properties

info:jhove2/format/xml XmlFormat

properties/dispatcher.properties

info:jhove2/format/xml XmlModule
Discussion

- 8:00  Continental breakfast
- 8:30  Welcome and introductions
- 8:35  Agenda and outcomes
- 8:40  Characterization
- 8:55  JHOVE2 project
- 9:15  Demonstration
- 9:40  Tea/coffee break
- 10:00 Integration
- 10:45 Module development
- 11:30 Questions/discussion
- 12:30 Lunch

- Distribution platform?
- Identifier scheme: info or http?
- Publish our properties as an ontology?
- Exhaustive type reporting?
- What have we gotten wrong (or right)?
- …
- We have some questions for you
  - Early testers/adoptors
  - Are you interested in module development?
  - Do you have assessment use cases?
  - Do you have test files you can share?
Questions?

http://confluence.ucop.edu/display/JHOVE2Info/Home
JHOVE2-Announce-L@listserv.ucop.edu
JHOVE2-Techtalk-L@listserv.ucop.edu