JHOVE2 Team : Requirements Document v1.1

This page last changed on Nov 05, 2008 by sabrams.

Overview

JHOVE2 will provide a highly configurable, extensible, and scalable framework for format-aware characterization.

Declarative Requirements (Definitions)

- **1. Characterization**. (1) Information *about* a digital object that describes its character, or significant nature. Characterization information is a type of OAIS representation information. It can function as an *surrogate* for the object itself for purposes of much preservation analysis and decision making. (2) The process of deriving this information.
  The characterization process has four aspects:
  - **1.1** *Identification* is the process of determining the presumptive format of a digital object on the basis of suggestive extrinsic hints (e.g. HTTP content-type) and intrinsic signatures, both internal (e.g. magic number) and external (e.g. file extension).
  - **1.2** *Feature extraction* is the process of reporting the intrinsic properties of a digital object. These may be general reportable properties (e.g. source unit name and modification date, reportable unit size and/or offset, format profile, message digest values), or genre or format-specific properties.
  - **1.3** *Validation* is the process of determining the level of conformance of a digital object to the normative syntactic and semantic requirements defined by the authoritative specification of the object's format.
  - **1.4** *Assessment* is the process of determining the level of acceptability of a digital object for a specific purpose on the basis of locally-defined policy rules.
- **2. Format.** A set of syntactic and semantic rules for encoding abstract information content into sequences of bits. Many formats can be grouped into loose categories, or *families,* sharing a general set of encoding rules that are further restricted or extended for the specific format, or *profile*. A format version is considered a profile.
- **3. Source unit**. A named file-like entity (or directory of such entities) passed to an invocation of JHOVE2 for characterization. The characterization of a directory entails the recursive traversal and characterization of each subsidiary file and sub-directory. Files must exist in a locally-attached file system, be network accessible by an *http* scheme URL, or be readable through a programmatic interface.
- **4. Reportable unit**. A logical object of characterization. Given *N* source units, the number of reportable units will be # *N*. Each file is, by definition, a reportable unit, but proper subsets of files and aggregations of files may also constitute reportable units.
- **5. Parsing**. Syntax-directed reading of a digital object's bit streams to retrieve the set of lexical tokens that encode that object's meaning.

Imperative Requirements

- **1.** JHOVE2 can process an arbitrary number of source units during a single invocation.
  - **1.1** A file source unit (whether read directly from a file system, retrieved via HTTP, or exposed

through a programmatic interface) will be treated as the root for a recursive traversal and characterization of nested byte streams.
    - **1.2** A directory source unit will be treated as the root for a recursive traversal and characterization of subsidiary files and sub-directories.
    - **1.3** Control over the scope of recursive traversal (e.g. maximum depth) will be a local configuration option.
- **2.** A reportable unit can be a file, a proper subset of a file (in other words, a PREMIS bytestream, "Contiguous or non-contiguous data within a file that has meaningful properties for preservation purposes"), or a set of files constituting a single logically self-contained aggregate object (a PREMIS representation, "set of stored digital files and structural metadata needed to provide a complete and reasonable rendition") whose encoding rules can be mapped back to a format.
- **3.** Characterization processing will be comprised of five (potentially independent) functions: identification, feature extraction, digesting, validation, and assessment. Each of these phases will produce useful information about a reportable unit.
    - **3.1** Identification can be performed against files, byte streams (proper subsets of files), and aggregates (sets of objects).
        - **3.1.1** File-level identification will be based on extrinsic hints and internal and external signatures.
        - **3.1.2** Bytestream-level identification will be based on extrinsic hints and internal signatures.
        - **3.1.3** Aggregate-level identification will be based on extrinsic hints and file-level properties (e.g. format, naming) and structural relationships (e.g. parent/child, siblings)
            - **3.1.3.1** Extrinsic hints will be defined in terms of locally-configurable options.
        - **3.1.4** Identification may be possible for formats for which no JHOVE2 feature extraction or validation functions are available.
        - **3.1.5** Identification will be available for the following formats: ASCII, ICC profile, JPEG 2000, PDF, TIFF, SGML, Shapefile, UTF-8, WAVE, and XML, and may be available for additional formats.
            - **3.1.5.1** The full set of identifiable formats should be consistent with those whose signatures are documented in the PRONOM and GDFR registries and the Unix magic database.
            - **3.1.5.2** The set of signature information used by the identification process can be easily extended as a locally-configuration option.
        - **3.1.6** Format identification will be reported in terms of a level of confidence.
            - **3.1.6.1** Identification may report more than one format for a given reportable unit.
            - **3.1.6.2** Identification results will be accompanied by confidence levels.
            - **3.1.6.3** The definitions of reportable confidence levels will be consistent with those used by DROID and the DSpace validation framework.
            - **3.1.6.4** The confidence levels associated with multiple identification results are not required to be unique. In other words, a reported unit may be identified with more than one format all with the same level of confidence.
            - **3.1.6.5** It should be possible, either in the JHOVE2 identification module or in a user extension of that module, to aggregate or refine the results of identification in order to dispatch the reportable unit to an appropriate module (or modules) for subsequent feature extraction, validation, and assessment.
        - **3.1.7** Format identification will be reported in terms of all of the common names (e.g "Tagged Image File Format"), MIME types (image/tiff), PRONOM identifiers (fmt/10), Apple UTIs (public.tiff), GDFR identifiers, ISO standards, or other applicable identifiers that the identification process can associate with the format.
            - **3.1.7.1** Format identifiers will be strongly typed in terms of their namespace.
        - **3.1.8** JHOVE2 identification determined by signature matching (and not by extrinsic hints) will in general only be able to distinguish formats with relatively coarse granularity. For example, it will report a file as XML but not a specific XML schema. Finer-grained determination of format may be possible as the result of subsequent feature extraction and validation processing.

- **3.2** By default, feature extraction will report the most inclusive set of properties derivable from the complete parsing of a reportable unit.
    - **3.2.1** Control over the granularity of parsing and reporting will be a locally-configurable option.
    - **3.2.2** The following general properties will be provided for *all* reportable units: identity (e.g. file pathname or URI), size (in octets), and last modification date.
        - **3.2.2.1** Message digesting may be performed against files or byte streams (proper subsets of files).
            - **3.2.2.1.1** Any combination of Adler-32, CRC-32, MD2, MD5, SHA-1, SHA-256, SHA-384, and SHA-512 digests can be calculated.
            - **3.3.2.1.2** By default, *no* digests will be calculated.
            - **3.3.2.1.3** The type (or types) of digests that are calculated will be a locally-configurable option.
        - **3.2.2.2** Many other properties *may* be reported for specific reportable units.
    - **3.2.3** General preservation properties will be reported as fully as possible in terms of PREMIS.
        - **3.2.3.1** PREMIS reporting will distinguish the reportable unit as either a bytestream, a file, or a representation.
    - **3.2.4** Genre- and format-specific properties will be reported in terms of well-known public data dictionaries and vocabularies.
        - **3.2.4.1** Still image properties will be reported in terms of ANSI/NISO Z39.87.
        - **3.2.4.2** Audio properties will be reported in terms of elements defined by the AES X-098B schema (but whose XML expression may not conform to that schema).
        - **3.2.4.3** Text properties will be reported in terms of the elements defined by the TextMD schema (but whose XML expression may not conform to that schema).
        - **3.2.4.4** In cases where well-defined data dictionaries or vocabularies do exist, additional properties not defined by those dictionaries or vocabularies may also be reported.
        - **3.2.4.5** In cases where well-defined data dictionaries or vocabularies do not exist, JHOVE2 documentation will provide unambiguous definitions of the JHOVE2-specific elements used.
        - **3.2.4.6** The XML expression of the reported properties will support the use of namespaced schemas in a manner that permits their full validation.
        - **3.2.4.7** For properties that are encoded in files (or byte streams) in non-textual form, the value can be reported in either the internal non-textual form, or by an appropriate textual label (whose language is drawn from the relevant specification), or both. For example, TIFF compression types are defined internally as integer values. JHOVE2 will be able to report the value as 4 (numeric), or "CCITT T.6 Group 4 Facsimile" (textual), or both.
            - **3.2.4.7.1** JHOVE2 will provide a mechanism for the language localization of text equivalents of non-textual internal values.
    - **3.2.5** An extracted property may be reported in terms of different data dictionaries or vocabularies if there is functional overlap between those dictionaries or vocabularies.
    - **3.2.6** It is possible for feature extraction to be supported for a format for which JHOVE2 does not support validation. This may occur when a module is composed of a thinly-wrapped 3rd-party tool or API.
    - **3.2.7** Feature extraction will be available for the following formats: ASCII, ICC profile, JPEG 2000, PDF, TIFF, SGML, Shapefile, UTF-8, WAVE, and XML, and may be available for additional formats.
    - **3.2.8** To the extent possible, extracted properties will be expressed in terms that are compatible with the Planets data model.
    - **3.2.9** Feature extraction will report external dependencies of the reportable unit, including other units required for comprehensive use of the file (e.g. stylesheets, external fonts, etc.), to the extent that internal links to those dependencies are found among the extractable properties of the reportable unit.

- **3.2.10** Repetitive instances of features may be reported either by enumeration (information reported about each instance) or by tally (summary information, including cardinality, of the set of instances).
    - **3.2.10.1** The choice of enumeration or tally for a property is a locally-configurable option.
- **3.2.11** *It is not a requirement* that extracted features be expressed in a manner compatible with XCDL.

° **3.3** Validation will be based on commonly accepted objective criteria as expressed in authoritative specifications.

- **3.3.1**. Validation will be available for the following formats: ASCII, ICC profile, JPEG 2000, PDF, TIFF, SGML, Shapefile, UTF-8, WAVE, and XML, and may be available for additional formats.
    - **3.3.1.1** It is *not a requirement* for JHOVE2 to provide validation support for AIFF, GIF, HTML, and JPEG, even though these were supported by JHOVE1.
- **3.3.2** Since many format specifications suffer from ambiguous language requiring subjective interpretation, control over the applicability of certain criteria, for which there may be a reasonable difference of opinion, will be a locally configurable option.
    - **3.3.2.2** Validation conformance will be reported in terms defined by the reportable unit's format. Note that many formats do not formally distinguish "well-formedness" from "validity".
    - **3.3.2.3** The validity of XML reportable units will be reported in terms of well-formedness and validity (to a Schema or DTD) as defined by the W3C Recommendations for XML 1.0 and 1.1.
        - **3.3.3.3.1** XML conformance will be reported as one of: not well-formed, well-formed (no Schema/DTD, validity cannot be determined), well-formed but not valid, well-formed and valid.
        - **3.3.2.3.2** XML validation will support the use of XML catalogs.
        - **3.3.2.3.3** XML validation will be schema/DTD-aware. If no actionable link to a schema or DTD is found within the reportable unit and if no such actionable link is defined by an associated catalog, validation will automatically default to checking well-formedness only.
    - **3.3.2.4** Validation will, when applicable, distinguish and report separately on syntactic conformance (can the bits of the reportable unit be tokenized) and semantic conformance (does the stream of tokens cohere into a consistent representation of content).
    - **3.3.3.5** By default, validation conformance will be exhaustive, documenting all violations that can be determined (not merely the first, as was the case with JHOVE1).
        - **3.3.3.5.1** Control over the granularity and inclusiveness of conformance checking and reporting will be a locally-configurable option.
    - **3.3.3.6** Error and informational messages will be couched in terms of terminology drawn from, and will include direct citations to the relevant named/numbered criteria in, the reportable unit's format specification.
        - **3.3.3.6.1** Language localization of messages will be supported.
    - **3.3.3.7** Validation will also determine and report on the specific profiles within a general format family to the rules of which the reportable unit conforms.
        - **3.3.3.7.1** JHOVE2 will introduce standardized patterns of module design for dealing with format profiles in a common and easily extended manner.
        - **3.3.3.7.2** By default, JHOVE2 will report only the profiles whose requirements are met by a reportable unit.
            - **3.3.3.7.3.1** JHOVE2 will provide a mechanism by which a reportable unit is validated with respect to a specific named profile. In such cases, JHOVE2 will report on all violations of profile requirements.

° **3.4** Assessment will be based on prior characterization information (identity, extracted features, validation status) evaluated in the context of locally-defined policies, heuristics, and

rules. Note that assessment refers to assessment of a instance of a format, not of the format itself.

- **3.4.1** JHOVE2 will be distributed with default assessment policies for each supported format based on evolving best practices in the preservation community.
  - **3.4.1.1** Characterization modules should be accompanied by a set of default assessment policies appropriate to the format.
- **3.4.2** Assessment rules can be modified and extended through local configuration without requiring programming changes to the JHOVE2 core framework or assessment module.
  - **3.4.2.1** To the fullest extent possible, the technology used to express and evaluate assessment rules should not constitute a barrier to use by preservation practitioners.
- **3.4.3** The expression language used to define assessment rules should be capable of documenting the reasoning behind any given policy choice.
- **3.4.4** Assessment reports should include references to the specific rules that were triggered and the characterization information that constituted the trigger event.
  - **3.4.4.1** Control over the scope and granularity of assessment reporting will be a local configuration option.

- **4.** JHOVE2 functionality will be encapsulated into reasonably-granular plug-in modules that can be installed in and invoked by the JHOVE2 framework.
  - **4.1** There will be no policy restriction as the range of function that can be encapsulated by a plug-in module.
    - **4.1.1** The plug-in interface will be designed so as to minimize any technical restriction on the range of function that can be encapsulated by a plug-in module.
    - **4.1.2** All modules will be accompanied by documentation clearly defining pre-conditions, inputs, processing, outputs, and post-conditions.
  - **4.2** In terms of abstract function (but not necessarily implementation), the canonical JHOVE2 processing stream is: identification # feature extraction # validation # assessment, applied against every reportable unit identifiable from amongst the source units.
  - **4.3** To the fullest extent possible, plug-in modules should minimize close coupling with the framework so that they can be easily deployed and invoked in other, non-JHOVE2 contexts. It should be possible, for example, for a user to invoke the identification module from the context of the user's own workflow (rather than from the JHOVE2 backplane), and then, later in that workflow, request validation. Note that do so would require that the request for validation include feature extraction information in the JHOVE2 format.
  - **4.4** A common data structure encapsulating characterization information will be available to all modules used in a given invocation.
    - **4.4.1** Modules may add new characterization information to this structure, which will be made available to all down-stream modules.
    - **4.4.2** Modules may choose to make processing decisions dependent upon the information they receive from up-stream modules.
    - **4.4.3** Intermediate results of characterization operations (identification, feature extraction, etc.) can be checkpointed to recover from requested or abnormal termination of processing.
  - **4.5** *It is not a requirement* that JHOVE2 supports job control functionality, such as pausing, resuming, and cancelling jobs.
    - **4.5.1** Conforming JHOVE2 modules should possess sufficient internal features to support job control functionality in other contexts.
  - **4.6** Individual modules may require configuration relating to external supporting libraries.
    - **4.6.1** XML validation can be configured to use an arbitrary XML parser.
    - **4.6.2** The PDF module will expose an API that will enable users with the appropriate RSA license to decrypt internal passwords.
  - **4.7** JHOVE2 will introduce a standardized pattern of error handling with error codes and more precise error messages using terminology and references drawn the appropriate specification documents.

        - **4.7.1** The textual representation of errors will support language localization.
    - **4.8** To the fullest extent possible, it should be easy to use JHOVE2 design patterns as the basis for module development outside the context of the JHOVE2 project.
    - **4.9** JHOVE2 modules should expose the set of their functionality in terms of a standard, but extensible, vocabulary, e.g. parse, validate, (symbolically) dump, edit, etc.
    - **4.10** JHOVE2 format modules will be structured in such a way as to facilitate wrapping 3rd-party tools or APIs in order to perform characterization for a format.
- **5.** The output from all characterization processing will be expressed in an intermediate XML form that can be recast into any desirable form through an XSL transformation.
    - **5.1** JHOVE2 will be distributed with XSL stylesheets for transforming JHOVE2 intermediate output into JHOVE1 form, in terms of both the Text and XML handlers.
    - **5.2** JHOVE2 output can be produced (possibly through an XSL transform) in terms of a METS 1.7 container.
        - **5.2.1** The METS output may be produced by JHOVE2 as its intermediate form or it may result from an XSL transformation of that intermediate form.
        - **5.2.2** *It is not a requirement* that JHOVE2 directly output PREMIS, but PREMIS output may be produced via XSL or some other transformation.
    - **5.3** The full set of JHOVE2 output will include a report of all configuration properties of the local installation and contextual information about the specific invocation.
        - **5.3.1** Reportable configuration and invocation properties will include: installation directory, plug-in modules, third-party dependencies (e.g. specific XML parser), working directory, Java configuration (JRE version/vendor/specification, installation directory, JVM version/vendor, class path, library path), number of available processors, user, per-module and aggregate timing and memory utilization.
        - **5.3.2** Control over the granularity of configuration and invocation reporting is a local configuration option.
- **6.** JHOVE2 will optionally produce human-readable displays in symbolic form of the contents of binary formatted objects.
    - **6.1** *It is not a requirement* that this display is compatible with with the canonical form of format display being developed by Planets (XCDL).
- **7.** JHOVE2 will include API-level support for editing and serializing formatted objects.
    - **7.1** It is *not a requirement* that JHOVE2 be distributed in a form that provides an editing/serializing interface. The distributed codebase (APIs and core framework) can, however, be used to develop such a function.
- **8.** JHOVE2 will operate as a stand-alone application with a command line interface (CLI) invokable from a command shell of common operating systems and computing platforms.
- **9.** To the fullest extent possible, JHOVE2 should be easily integrated into existing workflows.
    - **9.1** It is *not a requirement* that JHOVE2 be distributed with a graphical user interface (GUI).
    - **9.2** It is *not a requirement* that JHOVE2 be distributed with a web service interface.
- **10.** JHOVE2 performance will be no worse, and should be significantly better, than that of JHOVE1.

Non-Functional Requirements

- **1**. Static testing of source code will be performed as part of the test and release process.
- **2**. Documentation will include:
    - **2.1** JavaDocs for source code.
    - **2.2** Installation and user guide.
    - **2.3** Developer guide, including new format module development guide.
    - **2.4** Data dictionary for JHOVE2-specific schemas.
- **3**. Java version will be 1.6.

Open Questions

- **1.** Many format specifications suffer from ambiguity with regard to the distinction between requirements and recommendations. For example, the TIFF specification can be read to *require* that all internal offsets be word-aligned, rather than byte-aligned. In general, however, byte-alignment will have no effect on the correct rendering of the image by modern tools. The question is, is it preferable to support the local configuration of validation criteria (to allow local decisions regarding ambiguous specification language), or should the ambiguity be dealt with as a matter of assessment? In other words, should it be possible to downgrade a validation error condition to an informative condition, or should validation always report an error and let the assessment process determine whether that error is significant?