PreForma MediaInfo

Project Acronym: PREFORMA

Grant Agreement number: 619568

Project Title: PREservation FORMAts for culture information/e-archives

Prepared by:

- MediaArea SARL
- Jérôme Martinez
- Dave Rice
- Tessa Fallon
- Ashley Blewer
- Erik Piil
- Guillaume Roques

Prepared for:

Date: December 31, 2014

Summary

This reports serves as a snapshot of MediaArea's research, planning, and design work to develop a conformance checker, tentatively entitled PreForma MediaInfo.

Intended Behavior by Use Case

Overview

[[OAIS introduction]]

Conformance Checking at Creation Time

Conformance Checking at Transfer Time

Conformance Checking at Digitization Time

Verification of Lossless Digitization Until recently audiovisual digitization required a fairly inflexible set of hardware requirements and extremely limited possibilities for an open source approach to video digitization. Due to the bandwidth and processing requirements for the digitization of standard definition video required the installation of PCI cards and often the use of hardware encoders that were designed to encode video as fast as the video was being received to codecs like MPEG2 or JPEG2000. With modern connectivity options such as USB 3 and Thunderbolt it is easier to add video digitization capabilities to modern computers. Additionally modern computer processers can now transcode video losslessly in software from a video input without the need to rely on proprietary hardware-based encoders. Open source solutions such as DVA Profession, bmdcapture, and FFmpeg along with the open provision of video digitization software development kits, such as the Blackmagic SDK are facilitating new open development projects for archival video digitization.

As vendors and memory institutions are increasing considering and implementing digitization workflows that encode video directly to lossless codecs with the use of an intermediate file-based uncompressed audiovisual data, it is increasingly crucial to assess this lossless file soon after creation to detect any flaws within the digitization process.

For those digitizing video through processes that incorporate libav or FFmpeg such as bmdcapture of FFmpeg's decklink integration, a separate framemd5 may be written alongside the encoded ffv1 data. The resulting ffv1 data may then be verified against the framemd5 to verify that the correct bits were written to disk.

An inspiration for the use of framemd5 reports within a digitization workflow is inspired by the verify option with the flac utility available at http://flac.sourceforge.net/. The '-V' or –verify command is used to decode the encoded stream in parallel to the encoding process to double-check the losslessness of the transcoding. With this method any discrepancy between what data is read and transcoded versus what data is written to disk could be identified in a subsequent verification process. The use of framemd5 data within a digitization workflow enables verification in cases where an option similar to flac's –verify argument isn't available.

Assessment of Vendor Deliverables For archives that clarify specifications for audiovisual digitization projects, the conformance checker should facilitate a workflow for the archivist to express those specifications and verify received material against them. In addition to testing for the presence and order of required metadata tags the conformance checker should also be able to verify that they adher to particular patterns as expressed through regular expressions.

The conformance checker should be able to verify that files were transferred completely and that the delivered material does not contain any partial files from an incomplete or aborted transfer.

Conformance Checking at Migration Time

Fixity Verification Migration of large amounts of data introduce risk for digital corruption, sector loss. Ongoing data migration is essential for digital preservation but can require time consuming verification process. Both Matroska and FFV1 contain features for internal fixity, so that a file copied from point A to point B can be assessed at point B alone to verify the data integrity of the frames. MediaArea recommends uses Matroska's CRC features for use in digital preservation to allow for fixity verification to be more stable and achievable with the file alone without necessarily depending on external databases or records of checksums.

Obsolescence Monitoring Migration is typically an ideal time to perform obsolescence monitoring and preparing actions to limit complications in obsolescence status. Just as memory institutions must maintain the technology that their physical collections are dependent upon, this is equally true for digital collections. As this maintainance becomes more complex, costly, or unlikely archives will typically reformat material (with as little compromise to the content and characteristics of the source as possible) to a format that has more sustainable characteristics.

To counteract arising obsolescence challenges it is critical to have access to thorough sets of technical metadata in order to associate certain codecs, formats, or technologies with sustainability risks or to identify what one format should be superceded by another in a particular digital preservation. For instance an institution that utilized FFV1 version 0 as a lossless preservation codec may wish to identify such files to reformat them to FFV1 version 3 (now that it is non-experimental) in order to take advantage of version 3's additional advantages. In our research one archive found that some digitized material received from a vendor was missing technical metadata about field dominance and had to identify exactly which materials were affected to order to rectify the issue.

Introduction of Formats

Matroska

Matroska is a open-licensed audiovisual container format with extensive and flexible features and an active user community. The format is supported by a set of core utilities for manipulating and assessing Matroska files, such as mkvtoolnix and mkvalidator.

Matroska is based on EBML, Extensible Binary Meta Language. An EBML file is comprised of one of many "Elements". Each element is comprised of an identifier, a value that notes the size of the element's data payload, and the data payload itself. The data payload can either be stored data or more nested elements.

Matroska integrates a flexible and semantically comprehensive hierarchical metadata structure as well as digital preservation features such as the ability to provide CRC checksums internally per selected elements.

FFV1

FFV1 is a efficient lossless video codec which is designed in a manner responsive to the requirements of digital preservation. Version 3 of this lossless codec is highly self-descriptive and stores its own information regarding field dominence, aspect ratio, and colorspace so that it is not reliant on a container format to store this information. FFV1 version 3 mandates storage of CRCs in frame headers to allow verification of the encoded data and stores error status messages. FFV1 version 3 is also a very flexible codec allowing adjustments to the encoding process based on different priorities such as size efficiency, data resillience, or encoding speed.

Linear PCM

Development of an Open Source Conformance Checker

The conformance checker developed within the PreForma project must document and associate conformance rules with data types (such as containers or frames) and authorities (such as specifications, community practices, or the local rules of a memory institution). This design document focuses particularly on Matroska, FFV1, and LPCM.

Design Considerations

Interfaces The selected formats (MKV, FFV1, and LPCM) represent substantially distinct concepts: container, video, and audio. The optimization of a conformance checker should utilize distinct interfaces to address the conformance issues of these formats, but allow the resulting information to be summarized together.

An interface for assessing conformance of FFV1 video should enable review of the decoded FFV1 frames in association with conformance data so that inconsistencies or conformity issues may be reviewed in association of the presentation issues it may cause.

MediaArea proposes an interface to present conformity issues for audio and video streams (FFV1 and LPCM) on a timeline, so that conformance events, such as error concealment or crc validation issues may be reviewed effectively according to presentation, parent Matroska block element, or video frame.

The Matroska container requires a distinct interface that allows for its hierarchical structure to be reviewed and navigated. The presentation should allow for MKV elements to be expanded, condensed, or filtered according to element id or associated conformity issues.

Optimization for Large File Size Design of a conformance checker should be considerate of the large file sizes associated with video. For instance, an hour-long PAL FFV1 file (which contains 90,000 frames per hour) should provide efficient access if cases where one FFV1 frame contains a CRC validation error.

A video conformance checker should be well optimized and multi-threaded to allow for multiple simultaneous processing on video files. Additionally the conformance checker should allow a file to be reviewed even as it is being processed by the conformance checker.

Compliance with Standard Specifications

Compliance with Community-Driven Implementation Standards

Compliance with Local Institutional Criteria

Performance of Fixes

Substantial care should be exercised to ensure that the Conformance Checker properly associates risk, user warnings, and assessments with each fix allowed. In order to allow a fix the software must properly understand and classify what may be fixed and be aware of how the result may be an improvement. Adjustments directly to a preservation file must be handled programmatically with great caution with diligent levels of information provided to the user.

An example of a fix that could be enabled in the RIFF format could be verifying that any odd-byte length chunk is properly followed by a null-filled byte. Odd-byte length chunks that do not adhere to this rule cause substantial interoperability issues with data in any chunk following the odd-byte length one (this is particularly found in 24 bit mono WAV files). If the odd-byte length chunk is not followed by a null-filled padding byte, then most typically the next chunk starts where the padding byte is and the padding byte may be inserted so that other following chunks increase their offset by one byte. This scenario can be verified by testing chunk id and size declaration of all following bytes so that the software may know beforehand if the fix (inserting the null-filled padding byte) will succeed in correcting the RIFF chunk structure's adherence to its specification.

Fixes for Matroska files could include fixing metadata tags that don't include a SimpleTag element or re-clustering frames if a cluster does not start on a keyframe.

Focus on Fixity

Both FFV1 and Matroska provide fixity features that serve the objectives of digitial preservation by allow data to be independently validated without the requirement of managing an external checksumming process. FFV1 version 3 mandates CRC's on each frame. Matroska documents methods to embed checksums in Matroska elements to allow for future validation of any content.

Although the Matroska specification states that "All level 1 elements should include a CRC-32" this is not the practice of most Matroska multiplexers. As part of the Fixer aspect of this project, MediaArea proposes to develop a conformance checker that allows users to add CRC-32 to selected elements.

The advantages of embedded fixity in preservation media files is significant. The use of traditional external checksums does not scale fairly for audiovisual files, because since the file sizes are larger than non-audiovisual files there are less checksums per byte, which creates challenges in addressing corruption. By utilizing many checksums to protect smaller amounts of data within a preservation file format, the impact of any corruption may be associated to a much smaller digital area than the entire file (as the case with most external checksum workflows).

Ecosystem around Open Source Reference Implementation

Feedback and Reporting

Advance Improvement of Standard Specification

FFV1 Specification Efforts to create an FFV1 specification began in April 2012, continuing through the August 2013 release of FFV1 version 3. Currently the specification remains in development at http://github.com/ffmpeg/ffv1. Ideally a specification should fully inform the development of a decoder or parser without the need to reference existing implementations (such as the ffv1 implementations within ffmpeg and libav); however MediaArea's initial research and prototyping efforts with FFV1 found the current specification insufficient to create a decoder. As a result MediaArea utilized ffmpeg's FFV1 implementation to fully interpret the specification. Several threads on the ffmpeg-devel and libav-devel listserv reference discussions about the development of the FFV1 specification and consideration of efforts to standardize the specification through a standards organization, such as IETF (Internet Engineering Task Force) [1].

In consideration of FFV1's utilization within preservation contexts, the standardization of the codec through an open standards organization would better establish FFV1 as a trustworthy, stable, and documented option. At the moment FFV1 can be seen at a tipping point in its use within preservation context. It's speed, accessibility, and digital preservation features make it an increasingly attractive option for lossless video encoding that can be found in more and more large scale projects; the standardization of FFV1 through an open standards organization would be of broad interest to digital preservation communities and facilitate greater accessibility of lossless encoding options that are both efficient and standardization.

MediaArea proposes working closely with the lead authors of the FFV1 specification in order to update the current FFV1 specification to increase its self-reliance and increase its clarity. Development of the FFV1 specification early within the PreForma project will generate substantial feedback to the authors of the specification which could then be offered through the specification's github page via pull requests or the issue tracker. MediaArea proposes at a later stage of development that the Preforma project serve as a catalyst to organize, facilitate, and sponsor the IETF standardization process for FFV1.

Considering the 2 year timeline of the PreForma project and usual pace of IETF standardization projects, we proprose at least submitting FFV1 as an Independent Submission to IETF which could provide workable timeline, encourage a detailed review process, and assign a formal RFC number to the specification.

Matroska Specification Both the Matroska specification and its underlying specification for EBML are at mature and stable stage with thorough documentation and existing validators, but several efforts of the PreForma project can serve as contributions to this specifications. The underlying EBML specification [1] has already been drafted into RFC format but is has not yet been submitted to IETF as an Independent Submission or otherwise.

Matroska has a detailed metadata specification at http://www.matroska.org/technical/specs/tagging/index.html. Each tag has an official name and description while provides rules and recommendations for use. Many of these tags could be associated with validation rules, such as expressed by regular expression to assure that the content of the tag conforms to expectations. For instance tag such as URL, EMAIL, or ISBN have specific allowable patterns for what may be contained. As part of build a conformance tool for Matroska, MediaArea will generate conformance tests for individual tags and these tests may be contributed back to the Matroska specification in a list of regex values, an XML schematron file, or other acceptable contribution method.

Other Suggested Improvements or Contributions to Standard Specifications Register an official mime type via IETF for Matroska.

Register dedicated FFV1 codecid with Matroska (current use is via fourcc).

Proposal of a tagging extension to Matroska based on the requirements of the digital preservation community.

Feedback for features and functions of FFV1 version 4, which is currently under development.

Creation of metadata translators to convert common descriptive metadata formats within memory institution. For instance convert EBUCore into the XML representation of the Matroska tagging specification so that such metadata may be easily imported and exported between EBUCore and Matroska.

Advance Business Cases for Managing Preservation Files

Architectural Layers

The design of the conformance checker portion of the PreForma MediaInfo application will be comprised of several layers which will communicate via a Hypervisor. The layers shall include:

- Input Layer
- Container Conformance Check
- Container Demuxer
- Stream Conformance Check
- Stream Coherancy Check
- Stream Decoding (through plugin)
- Baseband Analysis (through plugin)
- Playback and Playback Analysis (through plugin)

Note that the PreForma tender does not require decoding or subsequent baseband analysis or playback; however, from our experience in conformance checker design with DV Analyzer and QCTools and through discovery interviews, we've found that users are quick to require some form of playback in order to facilitate decision-making, response, and strategies for fixing. For instance if the conformance checker warns that the Matroska container and FFV1 codec note contradictory aspect ratios or a single FFV1 frame registers a CRC mismatch it is intuitive that the user would need to decode the video to determine which aspect ratio is correct or to assess the impact of the CRC mismatch. These layers can be supporting by designing a conformance checker and shell that is prepared to utilize FFmpeg as an optional plugin to enable additional analysis features and playback. Our overall proposal is not dependent on supporting an FFmpeg plugin but we believe that preparing a conformance checker that could support FFmpeg as an optional plugin could create a more intuitive, comprehensive, and informed user experience.

We propose incorporating several compatible utilities into PreForma MediaInfo to extend functionality and add immediate convenience for users. Each component is built as a plugin and can be replaced by a third party tool.

Transport layer

Preforma MediaInfo: File on disk or direct memory mapping

Preforma MediaInfo natively offers a file API for each operating system to enable direct file access, including files that are still in the process or being written. The inclusion of MediaInfo also offers features for direct memory mapping which will be useful for third-party development or plugins.

Plugin integration proof of concept: libcURL

libcURL is licensed under an MIT license that is compatible with both GPLv3+ and MPLv2+. curl offers extensive support for transferring data through many protocols. By incorporating curl into PreForma MediaInfo the tool will be able to assess files that may be accessible online by providing a URL (or list of URLs) in place of a filepath.

Since we will be generating a library of reference and sample files that will include large audiovisual files, users will be able to assess reference files without necessarily needing to download them.

Used as a proof of concept of plugin integration:
HTTP/HTTPS/FTP/FTPS support via MediaInfo Open-Source GPLv3+/MPL2+ and libcurl (MIT license, compatible with GPLv3+/MPL2+)

Container/Wrapper implementation checker

Preforma MediaInfo: Matroska checker

Plugin integration proof of concept: mkvalidator

mkvalidator is a basic and no more maintained Matroska checker (BSD license, compatible with GPLv3+/MPL2+) which will be used mostly for demonstration of the plugin integration.

Container/Wrapper Demultiplexing

Preforma MediaInfo

PreForma MediaInfo will utilizing MediaInfo's existing demuxing libraries which will allow for PreForma's selected video codecs, FFV1 and JPEG2000, to be assessed from within many formats founds within archives although these container formats themselves aren't the focus of the current PreForma project. Through discovery interviews with archives and vendors we have found FFV1's archival implementations to use a variety of container formats such as AVI and QuickTime as well as Matroska. In order to allow developed tools to support FFV1 even if not contained within Matroska, PreForma MediaInfo will support the following formats for demuxing (though not necessarily for conformity (yet)): - MXF (commonly found within memory institutions) - MOV/MP4 (often found containing FFV1, JPEG2000, and LPCM) - DV (video stream format which uses LPCM) - AVI (used with FFV1 by DV Profession, NOA, Austria Mediathek) - WAV (a common container for LPCM)

By supporting the demultiplexing of these formats through MediaInfo, the developed tools will be applicable to a wide variety of files that contain PreForma's selected codecs: FFV1, JPEG2000, and LPCM. This demultiplexing support can be available through MediaInfo's existing libraries in a manner that is compatible with PreForma's licensing requirements.

Plugin integration proof of concept: FFmpeg

FFmpeg is one of the most ubitiquous, comprehensive, and open tools for demultiplexing and decoding audiovisual data; however, although FFmpeg's GPLv2+ license is compatible with PreForma's selected GPLv3+ license, it is not compatible with PreForma's other selected license, MPLv2+. As the PreForma conformance project evolves to support additional formats and codecs through plugins the use of FFmpeg's features are expected to becoming more and more appealing. For instance the integration of FFmpeg can provide integration of very comprehensive decoding and demultiplexing support beyond what can be easily provided with MediaInfo's demuxing libraries. FFmpeg's libavfilter library also provides access to waveform monitoring, vectorscope, audio meters, and other essential audiovisual inspection tools.

Although PreForma MediaInfo won't incorporate FFmpeg in order to comply with the MPLv2+ licensing requirement, we would like to design plugin support for FFmpeg. In this way a memory institution using PreForma MediaInfo could separately download FFmpeg and link the two together to enable additional tools such as: - Video Waveform Monitor - Vectorscope - Ability to inspect luminance and chroma planes separately - Audio Meters

We anticipate that the implementation of FFmpeg plugin support will substantially simplify the development of other plugins for broader codec and format support so that an entire decoder or demuxer does not need to be written from scratch in order to extend support.

Stream / Essence implementation checker

Preforma MediaInfo:

- FFV1
- PCM (including D-10 Audio, AES3)

Plugin integration proof of concept: jpylyzer

For JPEG 2000 (GPLv3+ license, compatible with GPLv3+ but not with MPL2+)

Plugin integration proof of concept: DV Analyzer

For DV (BSD license, compatible with GPLv3+ and MPL2+)

Optional

- MPEG-1/2 Video (including IMX, AS-07, D-10 Video, FIMS…)
- H.264/AVC (including AS-07)
- Dirac
- AC-3 (including AS-07)
- MPEG 1/2 Audio
- AAC
- Any other essence format on sponsor request (we have skills in DV, VC-1, VC-3, MPEG-4 Visual, H.263, H.265/HEVC, FLAC, Musepack, Wavepack, , BMP, DPX, EXR, JPEG, PNG, SubRip, WebVTT, N19/STL, TTML…)

Container/Wrapper vs Stream / Essence coherency checker

Preforma MediaInfo

PreForma MediaInfo will support the coherency check between all suppoted formats (see Container/Wrapper implementation checker and Stream / Essence implementation checker parts)

Stream / Essence decoder

Preforma MediaInfo

- PCM (including D-10 Audio, AES3)

Plugin integration proof of concept: FFmpeg

FFmpeg decoder (GPLv2+ license, compatible with GPLv3+ but not with MPL2+)

Plugin integration proof of concept: OpenJPEG

OpenJPEG decoder (BSD license, compatible with GPLv3+/MPL2+)

Baseband analyzer

Preforma MediaInfo

- None (only creation of the API)

Plugin integration proof of concept: QCTools

QCTools graphs (report on and graph data documenting video signal loss, flag errors in digitization, identify which errors and artifacts are in original format and which resulted from the digital transfer based on all the data collected in the past.)

Controler

Communication between all plugins - Shudeling - Statistics - Reporting - User management - Policies management - Human interface - Command line interface - GUI (based on Qt) - Web UI (server/client)

Ideas to put in the final doc

We are Open-Source natives: we do Open-Source since the first days. We have an Open-Source business model based on sponsored support (bug correction and feature requests). We don't do Open-Source only because Performa wants it, we believe in Open-Source. We are open to Open-source and non Open-Source (we can do non Open-source small developments on request, depending of the business model of the sponsor) Do other project think to demux (e.g. how do they plan to analyze JPEG 2000 in MOV)?

We are already involved in Open-Source with MediaInfo and recognized as it, e.g. we are officially provided by Open Source distros: - Debian: https://packages.debian.org/wheezy/mediainfo - Ubuntu: http://packages.ubuntu.com/utopic/mediainfo - RedHat / Fedora: https://apps.fedoraproject.org/packages/mediainfo - OpenSuse: http://packman.links2linux.org/package/mediainfo - Arch Linux: https://www.archlinux.org/packages/?q=mediai
- FreeBSD: http://www.freshports.org/multimedia/mediainfo/

We see 4 projects: Text, Image, A/V except JPEG 2000, JPEG 2000. Only 3 selected entities for phase 2 (3-5 written in call to tender, 3 written in kickoff meeting report)?

Test files; - PDF/A files buggy files: http://www.pdfa.org/2011/08/isartor-test-suite/ - JPEG 2000 files: https://github.com/openplanets/jpylyzer-test-files - Matroska buggy files: Homemade + request to Matroska mailing list - FFV1 buggy files: Homemade+ request to FFmpeg mailing list ## Style Guide – ## Source Code Guide

Portability

Source code MUST be built for portability between technical deployment platforms.

Modularity

Source code MUST be built in a modular fashion for improved maintainability.

Deployment

The Conformance Checker MUST allow for deployment in these five infrastructures/environments: The PREFORMA website, an evaluation framework, a stand-alone system, a network-based system, and legacy systems.

To sufficiently demonstrate the scope and functionality of the Conformance Checker, it, along with associated documentation and guidelines, must be made available at the PREFORMA project website. The PREFORMA website will be considered as the deliverable for the PREFORMA project.

In order to gather sufficient structured feedback on the conformance checking process, the Conformance Checker will require deployment within the DIRECT infrastructure for test and evaluation of the tool in the PCP procedure.

The Conformance Checker must have the capability to be packaged and run as an executible on a PC running any standard operating system (at least for: MS Windows 7, Mac OSX, common Linux distributions such as Ubuntu, Fedora, Debian, and Suse). This ensures the conformance checker can be used in small-scale institutions without centralized IT infrastructure.

The Conformance Checker must allow for deployment in network-based solutions (dedicated server, cloud solutions) for digital repositories.

The Conformance Checker must have the capability of being plugged into legacy systems via written API integration.

API's

The Conformance Checker MUST interface with other software systems via API's.

Open Source Practices

Development

Development of software in open source projects in PREFORMA MUST utilise effective open source work practices. Effective open source work practices include:

- use of nightly builds Nightly builds are automated neutral builds that reflect the current, most up-to-date status of a piece of developed software. Access made to nightly builds allow for groups of developers to work collaboratively and always assess the most current state of the software, with consideration to potential bugs or other hazards that could occur during the development process. Programmers are able to confidently determine if they “broke the build” (made the software inoperable) with their code and prevent or correct changes quickly, as needed.
- use of software configuration management systems (e.g. Git) Using a software configuration management system allow for version and revision control, an essential component to developers working collaboratively. A version control system allows multiple people to work on same or similar sections of the source code base at the same time with awareness and prevention of overlapping or conflicting work. Git will be used as the software configuration management system for this project.

- use of an open platform for open development (e.g. Github) An open platform on which to develop software facilitates the open development of that software. Public visibility and ability to contribute to the software by anyone allows for heartier, more reliable software. Feedback is more easily sought and more readily provided with the use of an open platform. Github will be used as the open platform for open development of this project.

Open Source Platforms

All development of software and all development of digital assets (related to developed open source software) in PREFORMA MUST be conducted and provided in open source projects at open development platforms.

Contribution Guide

File Naming Conventions

Files related to documentation should be named in CamelCase. Sample data should be added in snake_case with a sufficiently descriptive title.

Commit messages should concisely summarize the contribution. Commits should be cohesive and only include changes to relevant files (e.g. do not fix a typo in the Style Guide, change scope paramaters, and fix a bug all in the same commit).

Rules for Qt/C++ code:

4 spaces are used for indentation. Tabs are never used.

For more guidelines, refer to the Qt Coding Style guide: http://qt-project.org/wiki/Qt_Coding_Style For even more guidelines, Google guide on C++: http://google-styleguide.googlecode.com/svn/trunk/cppguide.html

Rules for contributing code

Contributions of code or additions to documentation must be written with Qt and must be made in the form of a branch submitted as a pull request.

1. Fork this repository (https://github.com/MediaArea/PreFormaMediaInfo/fork)
2. Create your feature branch (`git checkout -b my-new-feature`)
3. Commit your changes (`git commit -am 'Added some feature'`)
4. Push to the branch (`git push origin my-new-feature`)
5. Create a new Pull Request with a more verbose description of the proposed changes

Rules for contributing feedback

Feedback of all kind is encouraged and can either be made through opening an issue or by contacting the team directly at info@mediaarea.net

Linking

In order to facilitate self-description, intuitive discovery, and use of resulting code and documentation it is highly encouraged to utilize linking through documentation, tickets, commit messages, and within the code. For instance the registry itemizes individual conformance checks should link to code blocks and/or commits as software is developed that is associated to that conformance check. In this manner it should be feasible to easily review both human-readable descriptions of conformity checks and associated programmatic implementations.

License

The software and digital assets delivered by tenderer are made available under the following IPR conditions:

All software developed during the PREFORMA project MUST be provided under the two specific open source licenses: “GPLv3 or later” and “MPLv2 or later”.

All source code for all software developed during the PREFORMA project MUST always be identical between the two specific open source licenses (“GPLv3 or later” and “MPLv2 or later”).

All digital assets developed during the PREFORMA project MUST be provided under the open access license: Creative Commons CC-BY v4.0 and in open file formats, i.e. an open standard as defined in the European Interoperability Framework for Pan-European eGovernment Service (version 1.0 2004)

Checker Design: Conformance and Coherency

Conformance checks for both container formats (such as Matroska) and streams (such as LPCM and FFV1) shall be defined, registered, and associated with the code that performs the check. PreForma MediaInfo will perform and report on a growing list of tests per format. Many of these tests will be derived directly from the specifications or standard documents of a given file format, but other tests will derive from expected patterns and structural incoherency. Some checks focus on coherency between a stream and a container (such as if the container and stream utilize contradictory aspect ratios).

These checks shall have logical cause and effect or conditional relationships and shall be documented by the citation of external standards documentation or by the project’s own research and development. MediaArea plans to provide guidance for user communities to develop and explain their own ruleset in shareable form. An XML schema for conformance definition is provided. MediaArea’s development of conformance and policy checkers will involve several categories of tests. In addition to supporting conformity checks and logical interpretation of selected file formats, there is user desire for checks performed based on internal or institutional policy that are not necessarily embedded in the file format technical specifications. A PreForma MediaArea ‘shell’ shall be able to load multiple sets of conformity/coherency rulesets so that users may select which rulesets they choose to adhere to as well as create their own.

Conformance and coherency rulesets specifically targeting specification compliance of FFV1, Matroska, and LPCM are currently under development.

Matroska Conformance Checks (Draft)

Extension Test

Descriptor	Value
CCID	MKV-EXT
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	File name
Citation	http://www.matroska.org/node/2/revisions/153/view

Rule Clarity: Inferred

Quote:

Definition: The file extension SHOULD be one of the following (MKV, MKA, MKS, MK3D, WEBM)

Extension Test MKV

Descriptor	Value
CCID	MKV-EXT-MKV
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	File name
Citation	http://www.matroska.org/node/2/revisions/153/view

Rule Clarity: Inferred

Quote:

Definition: If the file extension is MKV, the file SHOULD contain at least one video track.

Extension Test MKA

Descriptor	Value
CCID	MKV-EXT-MKA
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	File name
Citation	http://www.matroska.org/node/7/revisions/214/view

Rule Clarity: Inferred

Quote:

Definition: If the file extension is MKA, the file SHOULD contain at least one audio track and no other type of track, i.e. “audio-only”.

Extension Test MKS

Descriptor	Value
CCID	MKV-EXT-MKS
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	File name
Citation	http://www.matroska.org/node/2/revisions/153/view

Rule Clarity: Inferred

Quote:

Definition: If the file extension is MKS, the file SHOULD contain at least one subtitle track and no other type of track, i.e. “subtitle-only”.

Extension Test MK3D

Descriptor	Value
CCID	MKV-EXT-MK3D
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	?
Target Format Part	File name, StereoMode element
Citation	http://www.matroska.org/node/2/revisions/153/view

Rule Clarity: Inferred

Quote:

Definition: If the file extension is MKV3D the file SHOULD contain at least one video track AND SHOULD contain at least one StereoMode element.

EBML Element Start

Descriptor	Value
CCID	MKV-EBML-ELEM-START
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header
Citation	specdata.xml

Rule Clarity:

Quote: “Set the EBML characteristics of the data to follow. Each EBML document has to start with this.”

Definition: An Matroska file MUST start with an EBML element id, ie. 0x1A45DFA3.

EBML vint efficiency

Descriptor	Value
CCID	EBML-VINT-EFF
Version	0
Authority	EBML Specification
Target Format	EBML
Target Format Version	all

Descriptor	Value
Target Format Part	EBML Structure
Citation	http://matroska.org/technical/specs/rfc/index.html

Rule Clarity:

Quote: Section 2.2 “IDs are always encoded in their shortest form, e.g. 1 is always encoded as 0x81 and never as 0x4001.”

Definition: The bits following the Element ID’s Length Descriptor are not more than (8 - ${bit-length-of-length-descriptor}) successive 0 bits, i.e. vint is expressed as efficiently as feasible.

Element ID Registered

Descriptor	Value
CCID	MKV-KNOWN-ELEM
Version	0
Authority	
Target Format	EBML
Target Format Version	all
Target Format Part	
Citation	

Rule Clarity: Inferred

Quote:

Definition: Ensure MKV Element ID is registered in specdata.xml (as of Dec. 13, 2014 this is 224 registered Element IDs)

Element Size 0x7F Reservation

Descriptor	Value
CCID	EBML-ELEM-SIZE-7F
Version	0
Authority	EBML Specification
Target Format	EBML
Target Format Version	all
Target Format Part	EBML Element Size
Citation	http://matroska.org/technical/specs/rfc/index.html

Rule Clarity: Warning, since it is possible (though unlikely) element size is unknown but then happens to be 127 bytes.

Quote: “Note that the shortest encoding form for 127 is 0x407f since 0x7f is reserved.”

Definition: If Element Size is set to 0x11111111 but element size is actually 127 bytes provide a warning.

Element Size Byte Length Limit

Descriptor	Value
CCID	EBML-ELEM-SIZE-CAP
Version	0
Authority	EBML Specification
Target Format	EBML
Target Format Version	all
Target Format Part	EBML Element Size
Citation	http://matroska.org/technical/specs/rfc/index.html

Rule Clarity:

Quote: Section 2.3: “The EBML element data size is encoded as a variable size integer with, by default, widths up to 8.”

Definition: The first eight bits of any Element Size may not start with 0b00000000.

Element Size Unknown

Descriptor	Value
CCID	EBML-ELEM-SIZE-UNK
Version	0
Authority	EBML Specification
Target Format	EBML
Target Format Version	all
Target Format Part	EBML Element Size
Citation	Dave

Rule Clarity: Warning

Quote: “Values with all data bits set to 1 means size unknown, which allows for dynamically generated EBML streams where the final size isn’t known beforehand.”

Definition: Warning on unknown element sizes, unoptimized MKV.

Level 0 Segment

Descriptor	Value
CCID	MKV-LEVEL-0
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header
Citation	specdata.xml

Rule Clarity:

Quote: Inferred: EBML and Segment are the only level 0 elements, both are allowed to occur multiple times.

Definition: The EBML Header MUST be immediately followed by another EBML Header Element, 0x1A45DFA3, or a Segment Element, 0x18538067. {{Can global Elements exist at level 0?!}}

Only One EBML Header recommended

Descriptor	Value
CCID	MKV-1-EBML
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Matroska structure
Citation	is there a rule to prevent this?

Rule Clarity: Warning

Quote: Assumed: Two EBML Headers in one MKV file seems contradictory.

Definition: There SHOULD only occur one EBML level 0 element within an MKV file. (EBML Headers could recur if an MKV file is an attachment of an MKV file).

File Size Consistency

Descriptor	Value
CCID	MKV-FILESIZE-MATCH
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Matroska structure
Citation	http://www.matroska.org/technical/specs/index.html#block_structure

Rule Clarity:

Quote: Inferred

Definition: The actual file size should be the sum of all level 0 Element Size declarations plus the sum of the byte sizes of level 0 Element IDs and Element Sizes.

EBMLVersion Presence

Descriptor	Value
CCID	MKV-EBML-VER
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header
Citation	

Rule Clarity:

Quote:

Definition: Within any EBML Header exactly one EMBL Version element must be present.

EBMLReadVersion Presence

Descriptor	Value
CCID	MKV-EBML-RV
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header
Citation	

Rule Clarity:

Quote:

Definition:

EBMLMaxIDLength Presence

Descriptor	Value
CCID	MKV-EBML-MAXIDL
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header
Citation	

Rule Clarity:

Quote:

Definition:

EBMLMaxSizeLength Presence

Descriptor	Value
CCID	MKV-EBML-MAXSL
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header
Citation	

Rule Clarity:

Quote:

Definition:

DocType Presence

Descriptor	Value
CCID	MKV-EBML-DOCT
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header
Citation	

Rule Clarity:

Quote:

Definition:

DocTypeVersion Presence

Descriptor	Value
CCID	MKV-EBML-DOCTV
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header
Citation	

Rule Clarity:

Quote:

Definition:

DocTypeReadVersion Presence

Descriptor	Value
CCID	MKV-EBML-DOCTRV
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header
Citation	

Rule Clarity:

Quote:

Definition:

EBML Version Coherency

Descriptor	Value
CCID	MKV-VER-COH
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header
Citation	http://www.matroska.org/technical/specs/index.html#block_structure

Rule Clarity: Inferred

Quote:

Definition: The value of EBMLVersion MUST be greater than or equal to the vale of EBMLReadVersion.

EBMLMaxIDLength Limits

Descriptor	Value
CCID	MKV-MAXID-LIMIT
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header
Citation	specdata.xml

Rule Clarity: Spec says "4 or less", but since the EBML ID length itself is 4, the EBMLMaxIDLength has not other valid value.

Quote:

Definition: MUST equal 4

EBMLMaxSizeLength Limit

Descriptor	Value
CCID	MKV-MAXSL-LIMIT
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header

Descriptor	Value
Citation	specdata.xml

Rule Clarity: “The maximum length of the sizes you’ll find in this file (8 or less in Matroska).”

Quote:

Definition: Must be less than or equal to 8 and greater than or equal to 1.

EBMLMaxSizeLength Matches

Descriptor	Value
CCID	MKV-MAXSL-MATCH
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header
Citation	specdata.xml

Rule Clarity:

Quote:

Definition: No Element Size Length exceeds the length noted in EBMLMaxSizeLength

DocType

Descriptor	Value
CCID	MKV-DOCT-KNOWN
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header
Citation	

Rule Clarity:

Quote:

Definition: MUST equal either “matroska” or “webm”

DocTypeVersion Coherency

Descriptor	Value
CCID	MKV-DOCTV-COH
Version	0

Descriptor	Value
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header
Citation	

Rule Clarity:

Quote:

Definition: The value of DocTypeVersion MUST be greater than or equal to the vale of DocTypeReadVersion.

DocTypeVersion Limits

Descriptor	Value
CCID	MKV-DOCTV-LIMIT
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	EBML Header
Citation	

Rule Clarity: Warning

Quote:

Definition: Values for DocTypeVersion and DocTypeReadVersion must be either 1, 2, 3, or 4.

Top Elements Coded on 4 Octets

Descriptor	Value
CCID	MKV-TOP-ELEM-4CODE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Matroska structure
Citation	http://www.matroska.org/technical/specs/index.html#block_structure

Rule Clarity: "All top-levels elements (Segment and direct sub-elements) are coded on 4 octets, i.e. class D elements."

Quote:

Definition: Note: this seems to contradict EBML rule to use most efficient element size, but perhaps this is an intention deviation of MKV to achieve top elements starting on mutiples of 4 octets. ?

CRC Order

Descriptor	Value
CCID	MKV-CRC-ORDER
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	CRC Element
Citation	http://www.matroska.org/technical/specs/index.html#block_structure

Rule Clarity: "The CRC element should be the first in it's parent master for easier reading."

Quote:

Definition: CRC Elements SHOULD be the first sub-Element of its parent Element.

CRC-32 Size Coherency

Descriptor	Value
CCID	MKV-CRC-COH
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	CRC Element
Citation	http://www.matroska.org/technical/specs/index.html#block_structure

Rule Clarity: Inferred: "The CRC in use is the IEEE CRC32 Little Endian"

Quote:

Definition: The Element Size of the CRC-32 Element MUST be 4 bytes (aka 32 bit).

CRC Validation

Descriptor	Value
CCID	MKV-CRC-VAL
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	CRC Element
Citation	

Rule Clarity:

Quote:

Definition: The crc hash of the CRC-32 element MUST validate the subsequent data of the parent Element, from the Element that follows the CRC-32 element to the end of the parent Element.

CRC Not Pointlessly Used

Descriptor	Value
CCID	MKV-CRC-REASON
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	CRC Element
Citation	author

Rule Clarity: Recommended

Quote:

Definition: A CRC-32 element should not be the only child Element of its parent Element (ie hashing no data).

CRC-Presence

Descriptor	Value
CCID	MKV-CRC-PRES
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	CRC Element
Citation	

Rule Clarity: “All level 1 elements should include a CRC-32.” but CRC-32 Element is NOT Mandatory. ?

Quote:

Definition: Warning when Level 1 elements have no CRC-32. Very common.

Single Segment Composition

Descriptor	Value
CCID	
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	
Citation	specdata.xml

Rule Clarity:

Quote: “Typically a Matroska file is composed of 1 segment.”

Definition: File MUST contain at least one segment.

Seek-Presence

Descriptor	Value
CCID	MKV-SEEK-PRES
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Meta Seek Element
Citation	specdata.xml

Rule Clarity:

Quote:

Definition: File MUST contain at least one Seek element.

SeekID-Presence

Descriptor	Value
CCID	MKV-SEEKID-PRES
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Meta Seek Element
Citation	specdata.xml

Rule Clarity:

Quote:

Definition: File MUST contain at least one SeekID element.

SeekID-Type

Descriptor	Value
CCID	MKV-SEEKID-TYPE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Meta Seek Element
Citation	specdata.xml

Rule Clarity:

Quote:

Definition: SeekID MUST be in binary format.

SeekPosition-Presence

Descriptor	Value
CCID	MKV-SEEKPOSITION-PRES
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Meta Seek Element
Citation	specdata.xml

Rule Clarity:

Quote:

Definition: File MUST contain at least one SeekPosition element.

Segment-Info-Presence

Descriptor	Value
CCID	MKV-SEGMENTINFO-PRES
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	specdata.xml

Rule Clarity:

Quote:

Definition: Segment information MUST contain at least one Info element.

SegmentUID-Range

Descriptor	Value
CCID	MKV-SEGMENTUID-RNG
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	http://www.matroska.org/technical/specs/index.html

Rule Clarity: Range cannot be zero.

Quote:

Definition: SegmentUID MUST be greater than zero.

SegmentUID-Size

Descriptor	Value
CCID	MKV-SEGMENTUID-SIZE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	http://www.matroska.org/technical/specs/index.html

Rule Clarity:

Quote:

Definition: If present, SegmentUID MUST be 128 bits (16 bytes) in size.

SegmentUID-Type

Descriptor	Value
CCID	MKV-SEGMENTUID-TYPE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	http://www.matroska.org/technical/specs/index.html

Rule Clarity:

Quote:

Definition: If present, SegmentUID MUST be in binary format.

SegmentFilename-Type

Descriptor	Value
CCID	MKV-SEGMENTFILENAME-TYPE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	http://www.matroska.org/technical/specs/index.html

Rule Clarity:

Quote:

Definition: If present, SegmentFilename MUST be in UTF-8 format.

PrevUID-Size

Descriptor	Value
CCID	MKV-PREVUID-SIZE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	http://www.matroska.org/technical/specs/index.html

Rule Clarity:

Quote:

Definition: If present, PrevUID MUST be in binary format.

PrevUID-Type

Descriptor	Value
CCID	MKV-PREVUID-TYPE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	http://www.matroska.org/technical/specs/index.html

Rule Clarity:

Quote:

Definition: If present, PrevUID MUST be 128 bits (16 bytes) in size.

PrevFilename-Type

Descriptor	Value
CCID	MKV-PREVFILENAME-TYPE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	http://www.matroska.org/technical/specs/index.html

Rule Clarity:

Quote:

Definition: If present, PrevFilename MUST be in UTF-8 format.

NextUID-Size

Descriptor	Value
CCID	MKV-NEXTUID-SIZE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	http://www.matroska.org/technical/specs/index.html

Rule Clarity:

Quote:

Definition: If present, NextUID MUST be in binary format.

NextUID-Type

Descriptor	Value
CCID	MKV-NEXTUID-TYPE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	http://www.matroska.org/technical/specs/index.html

Rule Clarity:

Quote:

Definition: If present, NextUID MUST be 128 bits (16 bytes) in size.

NextFilename-Type

Descriptor	Value
CCID	MKV-NEXTFILENAME-TYPE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	http://www.matroska.org/technical/specs/index.html

Rule Clarity:

Quote:

Definition: If present, NextFilename MUST be in UTF-8 format.

SegmentFamily-Size

Descriptor	Value
CCID	MKV-SEGMENTFAMILY-SIZE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	http://www.matroska.org/technical/specs/index.html

Rule Clarity:

Quote:

Definition: If present, SegmentFamily MUST be in binary format.

SegmentFamily-Type

Descriptor	Value
CCID	MKV-SEGMENTFAMILY-TYPE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	http://www.matroska.org/technical/specs/index.html

Rule Clarity:

Quote:

Definition: If present, SegmentFamily MUST be 128 bits (16 bytes) in size.

TimecodeScale-Presence

Descriptor	Value
CCID	MKV-TIMECODESCALE-PRES
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	http://www.matroska.org/technical/specs/index.html

Rule Clarity:

Quote:

Definition: File MUST contain at least one TimecodeScale element.

Duration-Range

Descriptor	Value
CCID	MKV-DURATION-RANG
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	http://www.matroska.org/technical/specs/index.html

Rule Clarity:

Quote:

Definition: If present, duration range MUST be greater than 0

Duration-Type

Descriptor	Value
CCID	MKV-DURATION-TYPE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	http://www.matroska.org/technical/specs/index.html

Rule Clarity:

Quote:

Definition: If present, duration type MUST be float integer.

DateUTC-Type

Descriptor	Value
CCID	MKV-DATEUTC-TYPE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	specdata.xml

Rule Clarity: UTC standards inferred.

Quote:

Definition: If present, DateUTC MUST be in date format and follow UTC standards.

Title-Type

Descriptor	Value
CCID	MKV-TITLE-TYPE
Version	0
Authority	Matroska Specification
Target Format	Matroska
Target Format Version	all
Target Format Part	Segment Element
Citation	specdata.xml

Rule Clarity:

Quote:

Definition: If present, Title MUST be in UTF-8 format.

FFV1 Conformance Checks (Draft)

Missing header

Descriptor	Value
CCID	OUTOFBAND-HEADER-MISSING
Version	0
Authority	FVV1 Specification
Target Format	FFV1 >=2
Target Format Version	all
Target Format Part	Header
Citation	http://www.ffmpeg.org/~michael/ffv1.html

Rule Clarity:

Quote: “Version 2 and later files use a global header”

Definition: If version is 2 or more, there should be a global header in the container private data

version

Descriptor	Value
CCID	FFV1-HEADER-version
Version	0
Authority	FVV1 Specification
Target Format	FFV1
Target Format Version	all

Descriptor	Value
Target Format Part	Header
Citation	http://www.ffmpeg.org/~michael/ffv1.html

Rule Clarity: Warning

Quote: “version 0, 1 or 3”

Definition: Maximum known version is 3, analysis stops (note: doc sometimes indicates version 4)

version 2

Descriptor	Value
CCID	FFV1-HEADER-version2
Version	0
Authority	FVV1 Specification
Target Format	FFV1
Target Format Version	all
Target Format Part	Header
Citation	http://www.ffmpeg.org/~michael/ffv1.html

Rule Clarity: Warning

Quote: “Version 2 was never enabled in the encoder thus version 2 files should not exist”

Definition: Version 2 is forbidden, analysis stops

micro_version 2

Descriptor	Value
CCID	FFV1-HEADER-micro_version
Version	0
Authority	FVV1 Specification
Target Format	FFV1
Target Format Version	all
Target Format Part	Header
Citation	http://www.ffmpeg.org/~michael/ffv1.html

Rule Clarity: Warning

Quote: “For version 3, micro_version is 4, micro versions prior to this represent pre standard”

Definition: Not supported version, high risk of decoding issue

coder_type

Descriptor	Value
CCID	FFV1-HEADER-coder_type

Descriptor	Value
Version	0
Authority	FVV1 Specification
Target Format	FFV1
Target Format Version	all
Target Format Part	Header
Citation	http://www.ffmpeg.org/~michael/ffv1.html

Rule Clarity:

Quote: “0 (Golomb Rice), 1 (Range coder), 2 (Range coder with custom state transition table)”

Definition: coder_type >2 is not supported

state_transition_delta

Descriptor	Value
CCID	FFV1-HEADER-state_transition_delta
Version	0
Authority	FVV1 Specification
Target Format	FFV1
Target Format Version	all
Target Format Part	Header
Citation	http://www.ffmpeg.org/~michael/ffv1.html

Rule Clarity:

Quote:

Definition: (To be defined)

colorspace_type

Descriptor	Value
CCID	FFV1-HEADER-colorspace_type
Version	0
Authority	FVV1 Specification
Target Format	FFV1
Target Format Version	all
Target Format Part	Header
Citation	http://www.ffmpeg.org/~michael/ffv1.html

Rule Clarity:

Quote: “0 (YCbCr), 1 (JPEG2000_RCT)”

Definition: colorspace_type >1 is not supported

bits_per_raw_sample

Descriptor	Value
CCID	FFV1-HEADER-bits_per_raw_sample
Version	0
Authority	FVV1 Specification
Target Format	FFV1
Target Format Version	all
Target Format Part	Header
Citation	http://www.ffmpeg.org/~michael/ffv1.html

Rule Clarity: Are other values valid?

Quote: “commonly 8, 9, 10 or 16”

Definition:

(More header tests to add)

Descriptor	Value
CCID	
Version	
Authority	
Target Format	
Target Format Version	
Target Format Part	
Citation	

Rule Clarity:

Quote:

Definition:

crc_parity

Descriptor	Value
CCID	FFV1-HEADER-crc_parity
Version	0
Authority	FVV1 Specification
Target Format	
Target Format Version	
Target Format Part	
Citation	

Rule Clarity:

Quote: “32bit that are choosen so that the global header as a whole or slice as a whole has a crc”

Definition:

LPCM Conformance Checks (Draft)

formatType

Descriptor	Value
CCID	BWF-LPCM-FMT
Version	0
Authority	EBU BWAV v2 Specification
Target Format	BWF
Target Format Version	2
Target Format Part	FormatChunk 'fmt'
Citation	EBU Tech 3285 v2, pg. 16

Rule Clarity: Inferred

Quote: "If the field of the is set to WAVE_FORMAT_PCM, then the waveform data consists of samples represented in pulse code modulation (PCM) format."

Definition: WAVE_FORMAT_PCM = 0x0001

bitsPerSample

Descriptor	Value
CCID	BWF-LPCM-BPS
Version	0
Authority	EBU BWAV v2 Specification
Target Format	BWF
Target Format Version	2
Target Format Part	FormatChunk 'fmt'
Citation	EBU Tech 3285 v2, pg. 17

Rule Clarity: Inferred

Quote: "The field specifies the number of bits of data used to represent each sample of each channel. If there are multiple channels, the sample size is the same for each channel."

Definition: valid bits per sample 16, 20 or 24

bytesPerSecond

Descriptor	Value
CCID	BWF-LPCM-BYT
Version	0
Authority	EBU BWAV v2 Specification
Target Format	BWF
Target Format Version	2
Target Format Part	FormatChunk 'fmt'
Citation	EBU Tech 3285 v2, pg. 17

Rule Clarity: Inferred

Quote: “For PCM data, the field of the ‘fmt’ chunk should be equal to the following formula rounded up to the next whole number: (nChannels x nSamplesPerSecond x nBitsPerSample) / 8”

Definition: MUST equal (nChannels x nSamplesPerSecond x nBitsPerSample) / 8 **only important for compressed formats

blockAlignment

Descriptor	Value
CCID	BWF-LPCM-BLK
Version	0
Authority	EBU BWAV v2 Specification
Target Format	BWF
Target Format Version	2
Target Format Part	FormatChunk ‘fmt’
Citation	EBU Tech 3285 v2, pg. 17

Rule Clarity: Inferred

Quote: “The field should be equal to the following formula, rounded to the next whole number: (nChannels x nBitsPerSample)/8”

Definition: C ontainer size (in bytes) of one set of samples. MUST equal (nChannels x nBitsPerSample)/8EBU **Note: The above formulae do not always give the correct answer. Strictly speaking, the number of bytes per sample (nBitsPerSample/8) should be rounded first. Then this integer should be multiplied by (which is always an integer) to give . This in turn should be multiplied by to give].

channelCount

Descriptor	Value
CCID	BWF-LPCM-CHN
Version	0
Authority	EBU BWAV v2 Specification
Target Format	BWF
Target Format Version	2
Target Format Part	FormatChunk ‘fmt’
Citation	EBU Tech 3285 v2, pg. 17

Rule Clarity: Inferred

Quote: 1 = mono, 2 = stereo, etc.

Definition: 1 = mono, 2 = stereo, etc.

nChannels

Descriptor	Value
CCID	BWF-LPCM-CHN

Descriptor	Value
Version	0
Authority	EBU BWAV v2 Specification
Target Format	BWF
Target Format Version	2
Target Format Part	FormatChunk ‘fmt’
Citation	EBU Tech 3285 v2, pg. 17

Rule Clarity: Inferred

Quote: “Number of channels in the wave, 1 for mono, 2 for stereo”

Definition: 1 = mono, 2 = stereo, etc.

sampleRate

Descriptor	Value
CCID	BWF-LPCM-SRT
Version	0
Authority	EBU BWAV v2 Specification
Target Format	BWF
Target Format Version	2
Target Format Part	FormatChunk ‘fmt’
Citation	EBU Tech 3285 v2, pg. 17

Rule Clarity: Inferred

Quote: “Frequency of the sample rate of the wave file. This should be 48000 or 44100 etc. This rate is also used by the sample size entry in the fact chunk to determine the length in time of the data.”

Definition: 32000, 44100, 48000, etc.

Container/Stream Coherency Checks (Draft)

Aspect Ratio Match

Descriptor	Value
CCID	COHERENCY-DAR
Version	0
Authority	
Target Format	FFV1/MKV
Target Format Version	
Target Format Part	
Citation	

Rule Clarity:

Quote:

Definition: Display Aspect Ratio indicated in the container (e.g. Matroska) is not the Display Aspect Ratio indicated in the FFV1 stream

Width Match

Descriptor	Value
CCID	COHERENCY-WIDTH
Version	0
Authority	FVV1 and Container Specifications
Target Format	FFV1
Target Format Version	Container
Target Format Part	all
Citation	Header

Rule Clarity:

Quote:

Definition: Width indicated in the container (e.g. Matroska) is not the width indicated in the FFV1 stream

Height Match

Descriptor	Value
CCID	COHERENCY-HEIGHT
Version	0
Authority	FVV1 and Container Specifications
Target Format	FFV1
Target Format Version	Container
Target Format Part	all
Citation	Header

Rule Clarity:

Quote:

Definition: Height indicated in the container (e.g. Matroska) is not the height indicated in the FFV1 stream