Conch – Appendix: Discovery Interviews

Project Acronym: PREFORMA

Grant Agreement number: 619568

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

Prepared by: MediaArea.net SARL

- Erik Piil
- Ashley Blewer
- Dave Rice

Prepared for: The PREFORMA Consortium

Date: March 2, 2015

Summary: In order to design a project responsive to the needs and concerns of a community of memory institutions, we released an online survey during the design phase to cover information. This report summarizes the findings of the survey and provides the raw data from the questionairre.

Interview Introduction

PREFORMA (PREservation FORMATs for culture information/e-archives) is a Pre-Commercial Procurement (PCP) project co-funded by the European Commission under its FP7-ICT Programme. PREFORMA aims to address the challenge of implementing good quality standardised file formats for preserving data content in the long term. The main objective is to give memory institutions full control of the process of the conformity tests of files to be ingested into archives. MediaArea is one of six teams that will present their software architecture to the PREFORMA panel in March 2015. The panel will then make a further selection amongst suppliers from phase 1 (research) to proceed to phase 2 (development).

MediaArea has been tasked with researching community standards and developing standard conformance checks for FFV1 (a lossless video codec), Matroska Multimedia Container, and Linear Pulse-Code Modulated Audio (LPCM) to support efforts of long-term preservation in memory institutions.

MediaArea is reaching out to select archivists, vendors, and technologists who utilize or develop these audiovisual formats as part of the discovery work of our current design phase.

Interview Questions

During the interview we would like to cover the following quesitons in regards to one or more of the following formats: FFV1, Matroska, LPCM:

- [ ] Please describe your use of the format.
- [ ] For what circumstances or uses would you recommend or not recommend the format?
- [ ] Please describe any concerns in the use or the selection of the format for preservation use.

- [ ] Please describe any lessons learned in the use of the format.
- [ ] How do you recommend assessing quality of or validating the format?
- [ ] Please describe any workflow or use of tools for file format conformance checking.
- [ ] What functions or features are essential in these workflows or tools? How could they be improved or expanded?
- [ ] What tools would better enable the use of the format within the OAIS model?
- [ ] What specifications, practices, rules, or expectations do you apply to the use of the format?
- [ ] Matroska and FFV1 are not standardized, is it problematic for you?
- [ ] Do you recommend specific conditions when using the format?
- [ ] Is there anything else you'd like to say about the format, conformance checking tools, or the project?

Use of the Interview

MediaArea would like to record interviews for the purpose of generating a document of interview notes. These notes will be provided to the interviewee to allow for any modifications requested by the interviewee. Once approved by the interviewee the notes will then be released under a Creative Commons license, CC-BY v4.0. Please direct any questions to info@mediaarea.net.

Interview with George Blood

Date: 12/16/2014

Interviewers: Dave, Erik

E: How is LPCM currently used at your facility?

G: We encounter LPCM in a variety of different ways. The most obvious example is when we perform the migration of analog audio recordings. Ninety-nine percent of the time, the analog materials we encounter are migrated to 96kHz/24-bit digital files with LPCM encoded data. In these files we also populate the BEXT chuck so that it exists should anyone wish to manage this metadata. Born-digital materials that we encounter are migrated in their native resolution, which is usually 48 or 44.1kHz/16-bit. We also encounter PCM from ADAT and DTRS at 20 and 24 bit. We see a tiny bit of format migration where we are taking other PCM formats and "normalizing" the data. For example, taking uncompressed files or things that are going to JPEG2000/MXF, etc.

E: Have you encountered any interoperability with LPCM?

G: In order to discuss interoperability issues with LPCM, we need to divide this conversation into LPCM as a codec itself, and what we take to be as issues with file wrappers. The biggest interoperability issue we've had in the past is text encoding for metadata; inconsistency between applications about whether the encoding was unicode or ASCII. As a solution, when we import textual metadata we now do a UTF-8 to ASCII crosswalk. We are now beyond issues of endianness, because it has just been "coded away". That is, whether the bit stream is encoded as most significant bit or least significant bit first in the file. The playback applications now universally play either version.

If I had a wish list to deal with certain issues, however, it would be more interoperability in files larger than 2GB, a specification that was baked into the WAV file format since it's inception. When you deliver files that are larger than 2GB, there's some untested certain percentage of them that people will have problems playing, largely in part due to the newer RIFF64 spec has not yet widely adopted in all audio applications. The other thing you see are more wrapper-based issues, such as the packaging of individual mono or stereo pairs, an issue particularly in MXF. Another example being the behavior of stereo pairs in DV files. Audio applications do not uniformly interpret dual mono files as a L/R pair. Sometime you'll get sequential playback, first channel 1, then channel 2. Or ONLY channel 1. When dealing with 4 channels, such as in MXF and DV. The can be encoded as 4 mono files, or 2 stereo files. Sometime you'll get the behavior just described, or the player will only deliver the first channel of the two pairs, and sometime you'll get 4 individual channels (the correct answer). Recently we've been fighting situations

where we're decoded 4 channels from analog tape, embedding all in the SDI stream, but the application will only wrap the first 2 channels.

E: How have your clients responded to these interoperability issues?

G: We deal with them in our shop. For us, if we can solve the issue of interoperability in our shop with the files that we are creating, then we can send the file deliverables out into the wild and they won't come back with any issues. The amount of troubleshooting we used to do with clients and how they were dealing with assets in their production/repository environments took a lot of time. By resolving as many interoperability issues ourself before we ship files saves us a whole lot of time and effort, if stuff has to be investigated and re-worked in any way.

For years we've been dealing with this interoperability issue, testing a lot of files we've created. For garden-variety WAV-wrapped PCM, for example, we will open the files in JHOVE. It's not a particularly media-friendly validation environment, but it does handle these kinds of files. We find out the orders of the chunks, and whether the file is well-formed as JHOVE understands it. However, we encounter instances where we will go on and use BWFMetaEdit, either to probe the file or to fix things like padding-byte errors and such. JHOVE doesn't give us error information. It'll simply refuse to open a file or not display embedded metadata (for example of there's a padding byte error).

E: Are their other programs that you find helpful in checking for conformance?

G: I would say that we don't have tools to use for conformance, per se. Perhaps they exist, but we haven't found any. There are some really nice tools in other areas that we use, particularly as we explore MXF. What we do now is open the files in a lot of different settings and do the testing empirically, as opposed to testing for conformance.

D: For PCM there's no specific "structure"; raw PCM could be odd/even byte length, and it could start at any byte value, etc. One method of conformance checking is to look at common implementations of LPCM that do in fact have more implicit rules. For example, PCM in DV has restricted sample values that act as error codes. These things might be relevant to LPCM conformance. Beyond that we...

G: That's really more of a wrapper issue rather than a PCM codec issue...

D: Exactly. We're trying to figure out whether support of raw PCM itself is needed, or, whether we need to support conformity check on common wrappers of PCM, which can be a slippery slope. Or, should we focus on analyzing the contents of PCM for things list zero-padding in the lower bits, non-zero sampling, out-of-phase audio, etc. Taking the QCTools approach where we are analyzing the raw PCM coming out of the decoder.

G: Some of the things mentioned would be either useful or not a heavy payload. Knowing what the available headroom is, clipping or non-clipping is a light payload. Knowing whether there are padded bits, on the other hand, could be more difficult. When we were doingbit transference tests we found that bits were oftentimes dithered, as opposed to padded, which could be more difficult to detect. Knowing if files are padded with zeros would be useful information, however looking at wrapper-related issues for conformance is probably the place to draw the line, although I agree it is a slippery slope.

D: In the DV spec it adds additional rules about error concealment. Are there other standards like DV that use PCM in a modified way?

G: There are fringe cases like PrismSound 24 bit-packing and HDCD, as well as highly proprietary cases with good documentation like MLP and Dolby E. Ian Dennis at Prism sound has probes that look at bit streams and may be able to help with identifying these kinds of issues.

E: So, to confirm, would you prioritize identifying wrapper specifications over the stream itself?

G: Yes. PCM in and of itself is extremely simple. What happens after the data is created is where it gets interesting. For example, whether it's funneled into a AES31 stream or a LightPipe or something like that, to the point where you get formatting issues.

D: In other words, there's not a whole lot of demand for "What kind of PCM is this?"

G: Exactly. To take another example, back in the day we had issues with PCM F1 issues where the emphasis flag got lost in the migrationprocess. The emphasis flag feature starts with PCM10, but is only in the F1 and 1610 family as an option. If you migrate it as a AES-31 stream or through SPDIF, there is a flag for the emphasis. Likewise CD-DA and DAT have this flag, and support for it is required in the

hardware. But the file wrappers do not have a place for this information so the feature drops out. You end up with encoded pre-emphasis without the technical metadata to know it needs to be decoded.

D: The CD format is where PCM has a lot of relevance. How do you do conformance checks of CD-R rips in particular?

G: Do you want a bit-for-bit copy or an error-corrected copy? Even if you want an ISO image, it will want to do the error correction in hardware coming off of the disc. For preservation we do DVDs to ISO images. However, it is uncommon to do ISO images for CD-R audio discs. As a result, things like CD text, pre-emphasis, and copy prohibit, buried the sub code will disappear.

D: With DVAnalyzer it will assess the bitstream coming off of the tape and throw flags indicating if there are errors in the read. This is used to determine how the read actually went. Are there similar approaches to PCM data ripped off of CD-Rs?

G: You would need access to the error correction polynomials. In our current rip station, our testers know at the block level what magnitude of error there has been, and whether it is detectable. But we don't know any details about what's going on otherwise. However, this feature is most-likely hardware-dependent and beyond the scope of your (implementation checker) project. Remember, CD playback is a system. Any test necessarily involves the data on the disc, the condition of the disc itself, and the machine that's reading the data from the disc.

Interview with Ian Henderson

Date: 12/23/2014

Interviewers: Dave, Jerome, Tessa

D: Describe your background research into FFV1 and Matroska?

I: Collection of HDCam and HDCam SR tapes for one particular job. It was open at the time as to how we were going to transfer these to digital files. Looked at lossy compressions, various other formats, but it seemed if we're going to try and capture as much information as we can, the best way is to go with lossless compression. So that limited the amount of options that were available. I looked at MXF and jpeg2000 and initially I was very keen; but that actual implementation isn't very standardized.

Perceived wisdom in archives is that you follow what the broadcasters are doing, and I don't think that's the best fit.

FFV1 seemed fairly easy and controllable option. There didn't seem to be anything negative about it in regards to preserving the original source and keeping it available for future transcriptions and migration.

The reason for using Matroska: supported FFV1, we wanted to preserve all the audio streams available on the tapes (4 for HDCam, 12 for HDCam SR) and it had good support for multi-stream, and the expanded capacity for metadata storage.

D: What tools are you using right now to work with FFV1 and/or Matroska?

I: Transcoded externally via the BBC. If I need to do any work with it, I use ffmpeg. We haven't got an awful lot of a toolset. Whatever I come across that could be useful: tools for ID, categorization, migration. At the moment, MediaInfo for examining video, and one or two other tools. QCTools was mentioned.

D: What tools do you use for compliance checking?

I: Fairly limited. We use Droid for ID, and MediaInfo, and basically we retrieve whatever metadata information is in there. FFprobe. Something we'll have to look at. Any advice would be appreciated.

No problems with FFV1; only problem we had was with previews of files. But with FFV1, no problems. We have 125-150 digital files output from the tape, no problems at all. Largely for checking, it was a visual comparison with the H264. Making sure all the footage was there, the audio streams were available. Really a manual check process. We need to get more into performance software.

We only have source material. This was quite an unusual set in that it was the first time we've ever had videotape. We needed to find the best fit for that job.

Because we retrieve material from government department, it's unusual to receive media. We're now getting more and more stitched file content that we're retrieving.

J: Do you use MediaInfo manually?

I: Basically manually. We look at the files themselves or run an instance over a number of files using a Windows .bat file. There might be an implementation in our archival system, that would basically be running the same check.

I'll find out more about that system.

D: Do you move MediaInfo results into a db or store in a text file?

I: We store it in an archival system. SDB. NYU uses it. The info is stored within the database, a mixture of technical and description.

D: What coding specs do you use when you're requesting FFV1/Matroska?

I: There weren't that many options we were worried about with FFV1. We wanted FFV1 version 3, a number of slices, CRC for slices turned on,

[audio cuts out]

It was basically a set of parameters from Peter[Austria].

D: From Matroska?

I: We provided them a list of embedded metadata.

D: All official metadata tags? Not custom?

I: As far as possible, standard is always best. There's a reason that they're there.

It's quite resource heavy on playback. We don't tend to have much in the way of advanced spec PCs here. But that's to be expected with HD material on playback. MKV's been fine for holding all the stream and metadata content; FFV1 does a very good job retaining information. I'm quite happy with it.

D: Any modifications to files after they get back from vendor?

I: At the moment, we're storing as received. If there is a continuous recording session, we store that as 1 file. Shorter sections, individual files. Quite sizable files, no problems there.

J: Is it problematic with FFV1 and Matroska not being formal standards?

I: For us, it was a practical, pragmatic approach, and the fact that it was open–obviously if there's any need we can always move onto a further format. Yes, I would like to see it a bit more promoted or seen as a standardized format.

FFV1: it does seem to be under-promoted; it's not on people's horizons at the moment. People have heard something vaguely about it, or not heard of it all. It just needs its profile raised. It does become a quite favorable option once people have had a look at it. It just needs a bit more of a push.

Interview with Christopher Kummer

Date: 12/23/2014

Interviewers: Dave, Jerome, Tessa

D: Why does NOA recommend FFV11 for a few large scale projects?

C: We were one of the first companies asking for 96khz and 24-bit linear PCM. And that was regarded as crazy. We were quite alone on the market, and now it's become standard. At that time, the BWF extension of the specification was in the hands of a few vendors; we could not just jump into the BWF extensions. In 2000, why don't we store that data in Access? We can use the chunks from the db to create BWF.

In 2006-2009, we started to do very large migration projects with Swedish TV, we were trying to convince them to go with mjpeg2000. Then they decided to go for DVCPro50. They did an analysis which showed

that they didn't have the technology to include it in the production process. At that time we were developing a lot towards video, but much less on the codec and container side, more on the file logistics and container side. That was the moment we thought we have a huge market in legacy archiving and legscy digitisation. In 2010 we started to develop our product called FrameVector[?]. It is a two channel SD ingest machine which monitors our RF signals from the heads which collects [10:11] all the interactive supports from the sony[?] machine and gives you an transfer tool specially for analog material where you can move the question of proactive [?] machines to a reactive scenario. This machine is recording to a lossless mezzanine format that we call "ffvhuff", a combination of HuffYUV and FFV1. Huffyuv goes up to 10 bits and has half the payload of uncompressed. So we have a mezzanine format which can be transferred to whatever broadcast archive format you need.

Why do we use FFV1? The situation that we try to give our clients is that if you go for archives, please choose something you are able on access to transcode whatever format you want to have on access without a quality loss. So that was our starting point.

Peter Bubestinger suggested FFV1. We found some deficiencies with FFV1 when using AVI. Our current implementation is based on FFV1 in an AVI container. We decided against Matroska; the decision was a pure on access speed decision. AVI codec can be immediately fed to a transcoder without an additional file copy. Example: As a user, I would like to have 20 segments of 20 different films, 20 segments of 2-3 minutes, within an acceptable time on my machine. That is time for transfer from tape storage to cache to transcoding mechanism. . . so you have several copy mechanisms until you get to the file. And this must not exceed a certain time, or should be as low as possible, and that's why we ruled out jpeg2K. jpeg2K is sometimes 4 times slower than FFV1.3. Speed in combination with accessibility of technology in combination with no license cost in combination with I can play it on every machine in combination with it is an open-format so whoever decides to use it does not have to invest in a technology so that in 10 years to be able to read the format but he can also read out the complete information without the hope of NOA.

D/J: is it a problem that FFV1 is not standardized?

C: As an archive, you're not only responsible to trust some SMPTE organization, you're responsible for buying enough replay equipment to replay the material. If you're not able to secure the software replay mechanism to be able to replay your essence in a way which gives you access to the complete technology–that's what we do with the current archive implementations. Ex: ffmpeg. At any time I'm able to reproduce the information not only with the compiled code but with the machine code. It's a huge structural advantage for clients. It's a question of can I secure the technology and all its code towards the essence? If I rely only on the SMPTE standard but I don't have the technology available, what is the SMPTE standard worth?

J: the standard is not important for you?

C: No. I'm saying that having source code available to be able to replay your essence is a significant advantage vs mjpeg2000. At the moment. [Problems with licensing terms for jpeg2000]. Open source doesn't mean no money.

D: FFV1 standardization?

C: An excellent idea.

D: What motivated your sponsorship for Niedemayer's FFV1 specification document?

C: Compared to what we would have to develop in some proprietary codec technology, we're always open to sponsoring ffmpeg development.

D: What motivated your sponsorship for original FFV1 specification?

C: The same motivation for saying in 2000 that 96 kHz is better. We believe in mathematically lossless codecs first; second, we see that we have an implementation that is easy to use, it's very fast, it's open.

D: with FFV1, are there particular aspects of the codec that are harder to control? Are there needs for tool development around FFV1?

C: You have decided to recommend Matroska?

D: No, that was a decision before we got involved in the project.

C: [SEE FILE: https://api.asm.skype.com/s/i?0-neu-d1-87c1d638991f4e8295e1c68125001de2] How we compensate all the deficiencies of the AVI container. If Matroska would have had a bigger cartography on the professional side, we would have chosen Matroska. [Discussion of AVI deficiencies: 30:00]

Interview with Hermann Lewetz and Peter Bubestinger

Date: 12/16/2014

Interviewers: Dave, Erik, Tessa

Dave: why did you decide to use FFV1 and not Matroska?

Hermann: We started with MXF and jpeg2000, and then we realized we couldn't use these without having machine to write to the format. We didn't want to use a format that was dependent on a vendor. We wanted to use something where we we had access to the technology that creates the format; we wanted to use a very common format but didn't work out, so we wanted to use the best thing we could find, which was FFV1. We thought about Matroska, but at the time it was quite new and not so widespread; and the AVI container was very widespread and in use. We were already worried about decision to use FFV1 and since AVI is an old format we wouldn't have to defend that as well. [Making this decision in 2009]

Peter: Matroska was still not widely accepted as a professional container

Dave: would you make the same decision now?

Peter: I don't think from a personal POV the position of Matroska has not changed that much yet; I like it from the technical point of view. As Hermann already said, it has a stigma of 'oh yeah this is the the thing that people download stuff from the internet with."

Dave: for FFV1, same decision?

Both: Yes

P: now even easier

Dave: how would you prioritize openness and standardization in these kind of decisions?

Peter: Openness is extremely import in comparing standardization. MXF jpeg200: implementations vary. The only importance of the standardization for us is political; it would make things easier. As a technician/developer, if I have right to use source code as I wish, I can archive my source code and this is worth more to sustainability than having a paper which may or may not match my actual implementations.

Dave: . . . any containers you worked with you struggled to understand the technology?

Peter: For video, we've received some but not too many MXF files; not just during evaluation of which format to choose but from broadcasters, and some times some parts worked, some didn't, and you couldn't figure out what was going on. Problem with born-digital files created in not really known detailed conditions. EX: conference recordings where a photo camera was used for video recording. Playing back with VLC worked fine, but transcoding it caused the audio-video to go out of sync.

Dave: Lessons learned from use of FFV1?

Hermann: What I've learned in 4 years about fflv1 is it just works; it was the right decision. If we want to change, we can migrate from this without a loss, but meanwhile I don't think I've thought about migrating from this codec because it is still the best thing I can imagine.

Peter: Lessons learned: to be less afraid of using something because everyone is using something else. After 5 years, many institutions I've talked with, they were very skeptical about something new working; if it could work, someone else would have done it. This first step of trying something that is not accepted within the community or domain we're working in. And I've also seen that in other areas where IT is used and people are super cautious about which horse to bet on.

Hermann: In the beginning, I tried to find people; for example, I talked with James Snyder from Culpepper and I tried to find out why jpeg2000 was so important to him, what he thinks about FFV1 as another option, what about the problem that I can't open it with any other tool. I had the same doubts; if this is so good, why weren't other people using it or trying it?

Dave: talk more about conformance and validation; how do you assess quality or conformance? Workflow for conformance checking?

Peter: ffmpeg running for transcoding; we autocreate mpg2 DVD access copy of FFV1 files; the mpg2 is checked during the transition workflow. Ff mpeg plays well audio is there from beginning to end, everything is sync–we do not deeply inspect bitstream for standards compliance; but we do have ffmpeg tell us if something is wrong. I don't think there is a need to do bitstream analysis, but it would be really nice for the newer version to know whether the right parameters have actually been set. Currently I don't know if there is a possibility of actually checking that.

Dave: debug1; intended encoding and actual

Hermann: Necessary to have it in a standard workflow; and as problem accumulate, it would be great to have it

Peter: During development of FFV1 v. 3, I ran massive test suites in all variations; what is lacking there (and what I'm really looking forward to making) are fake tests. I'm a bit afraid of silent regressions currently happening.

Hermann: Another advantage of FFV1: there are few different implementations; few implementations, and all use the same implementation; what I fear is moment when big vendors accept FFV1 as a standard and they start to create different implementations; same thing as happened to jpeg2000. A standardization/implementation checker would be more important now.

Peter: It could be suggested once the integrity check exists that people who intend to buy any product using FFV1 should only do if the product complies to your checker. [Will send citation for AMIA-list quote]

Erik: reformatting workflow?

Hermann: When we ingest analog material we directly ingest in FFV1; no intermediate format.

Erik: FFV1 not too incorporated with analog-digital converters...

Hermann: We started having a converter converting all into SDI; and we have a DeckLink card with SDI input, and we capture FFV1 directly using ffShot trial and [REtrotap?] as an application; but because we are not very happy with the converters, we will try some other options.

Peter: But it was not necessary to have FFV1 directly in the converter hardware; we prefer having uncompressed coming and use the software to choose. It turned out in the past hardware implementations might be nice but very often you're locked into whatever they offer you and you have no ability to change it unless it's an open hardware. Other parts of workflow where we use FFV1: for born-digital we have whitelist of codecs that we allow into archive for the time being because of space considerations, so we don't transcode those to fFFV1; everything else is blown up to FFV1. For all edits, minor or opening it in a non-linear video editing system, we use FFV1 as a internal working format. We edit directly with FFV1, then export to FFV1 again.

A very important key factor is the support of FFV1 as a codec in tools one is using through tool chain. We use KDN Live (editing) and VirtualDub (in and out points). When it's necessary to transcode to FFV1 (Velocity, Avid, FileCut), that is a major turn-off for people.

H: Adobe Premier for example can import.

P: Yes, with plugins. If those few applications would have built-in support for FFV1, that would be an incredible breakthrough. You can do full HD lossless on a regular computer with FFV1 v. 3; we did this with tools like ShotCut, KDN Live to edit stuff in real time using FFV1 directly. We know that this works, it's just the fault of the applications if they can't support it.

H: Mpg2 was the proof for the archive copy because in 2010, computer couldn't handle FFV1 or SDI. We check the FFV1 directly, with the E family, 3,5,7 you don't need any special thing, and with SDI size it uses about 30% processing power. We realized that in fact we don't need the access copy for the checking.

P: We were also were surprised that people in-house were opening archive lossless copies without complaining or realizing that they're playing lossless on their office computers.