strategies and tactics for high-volume digitisation and digital preservation
<table>
<thead>
<tr>
<th>Category</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>fiction films</td>
<td>40,000</td>
</tr>
<tr>
<td>nonfiction films</td>
<td>130,000</td>
</tr>
<tr>
<td>television progs</td>
<td>780,000</td>
</tr>
<tr>
<td>scripts</td>
<td>21,000</td>
</tr>
<tr>
<td>press books</td>
<td>28,000</td>
</tr>
<tr>
<td>posters</td>
<td>19,000</td>
</tr>
<tr>
<td>photographs</td>
<td>1.6m</td>
</tr>
<tr>
<td>production + costume designs</td>
<td>5,000</td>
</tr>
<tr>
<td>library books</td>
<td>45,000</td>
</tr>
<tr>
<td>special collections</td>
<td>600</td>
</tr>
</tbody>
</table>
Unlocking Film Heritage project

- digitise + digitally preserve 10,000 films
- from all of the UK’s film archives
- including entirety of Victorian-era film
- make the films available online
- with a map interface (yay, geolocations!)
- procure, integrate + deploy digital preservation infrastructure
- consolidate + extend film digitisation capability

https://www.bfi.org.uk/britain-on-film/unlocking-film-heritage
BFI2022 video digitisation project

- digitise + digitally preserve **100,000 works** from videotape
- broadcast and non-broadcast content
- from all of the UK’s film archives
- consolidate + extend video digitisation capability
strategies and tactics for digitisation and digital preservation
strategies and tactics for digitisation and digital preservation

How on earth do you do digital preservation of 10,000 films and 100,000 tv progs in 10 years?
1. community
NO TIME TO WAIT #1, 2, 3, 4

https://mediaarea.net/NoTimeToWait3
AMIA-L Listserv

The AMIA-L list a forum for discussion among professionals interested in issues relevant to issues any and all involving all aspects of moving image materials and archives.

AMIA sponsors the AMIA-L list as a forum for discussion among professionals interested in issues relevant to issues involving all aspects of moving image materials and moving image archives, and to any related technologies or special interests of the profession. AMIA members and non-members are welcome, and subscription to AMIA-L is open to the public.

- AMIA Code of Conduct
- Public Listserv Rules and Guidelines

https://amianet.org/engage/amia-l-listserv/
IASA-TC 06 Guidelines for the Preservation of Video Recordings

Archives hold original video recordings in a range of types, from media-dependent, carrier-based analogue videotapes to computer-file-based digital recordings. The appropriate preservation treatments for this array reflect the variation in the source recordings. For analogue videotapes, for example, digitisation is called for. Meanwhile, examples of digital file-based recording may require rewrapping into a fresh file "wrapper" or a combination of digital transcoding and rewrapping.

When complete, IASA-TC 06 will cover the full range of topics in the preceding paragraph, as well as providing advice on shooting ethnographic, documentary, and oral history video footage in a manner that maximizes its "preserve-ability".

IASA-TC 06 is being disseminated in phases. At a high level, there are two dissemination phases:

1. Initial edition, publicly accessible in early 2018: focus on the preservation of video recordings on conventional carriers. For the most part, this means the digitization of analogue videotapes, but it also encompasses the transfer of content from some types of digital videotapes.

2. Expanded edition, access planned for in 2019: discussion of the preservation of digital-file-based video formats and the transfer of digitally encoded recordings in videocassette form (which may entail transcoding and/or rewrapping). Additional sections are anticipated that pertain to metadata and the production of new recordings in preservable formats.

This initial presentation on the IASA website is for comment and will support future revisions and improvements. Readers are asked to send notes to the chair of the IASA Technical Committee: Lars Gaustad (lars.gaustad@nb.no). Lars will share them with the guideline’s authors and, from time to time, the authors will summarize the comments together with their responses either via the IASA blog (www.iasa-web.org/blog) or one of the forums (e.g., www.iasa-web.org/forums/public-forums/technical-issues).

https://www.iasa-web.org/tc06/guidelines-preservation-video-recordings
Kieran O'Leary
kieranjol

iFscripts
Detailed documentation is available here:

ff1-frontend
Python ★ 24 ▼ 21

LibraryOfCongress/bagit-python
Work with BagIt packages from Python.
Python ★ 115 ▼ 65

amiacap.sourceforge/ffmpiprovisor
Repository of useful FFmpeg commands for archivists!
HTML ★ 140 ▼ 31

MediaArea/Mediainfo
Convenient unified display of the most relevant technical
and tag data for video and audio files.
Pascal ★ 203 ▼ 63

premisviewer
Transforms PREMIS XML into a human readable report
via python/Xml
Python ★ 2

Executable File | 389 lines (252 loc) | 11.8 KB
1 #!/usr/bin/env python
2 3 # Written by Kieran O'Leary, with a major review and overhaul/cleanup by Zach Kelling aka @beekay
4 # Makes a single ff1. Hof
5 6 # Import subprocess
7 # Import sys
8 # Import filecmp
9 # Import os
10 # Import shell
11 # Import cxf
12 # Import time
13 # Import lsbtools
14 # Import pep8
15 # from glob, import glob
16 try:
17 # from jifuncs import set_env, remove
18 # from jifuncs import hashLib, randomInt
19 # from jifuncs import make_mediatrace
20 # from jifuncs import make_mediafile
21 # from jifuncs import get_mediainfo
22 # from jifuncs import append_files
23 # from jifuncs import create_csv
24 # from jifuncs import generate_log
25 # from jifuncs import get_ffmpeg_info
26 except ImportError:
27 print('*** ERROR - IFIFUNCS IS MISSING ***
28 # Makeff1 requires that jifuncs.py is located in the same directory
29 # as some functions are located in that script
30 # https://github.com/kieranjol/iFscripts/blob/master/jififuncs.py
31 sys.exit()
32
33 def read_non_comment_lines(filename):
34 # Adapted from Andrew Beale - http://stackoverflow.com/a/8364873/2168972
35 for line in open(filename):
36 if line[1:] == '#':
37 yield line
38 def get_input():
39 if len(sys.argv) == 3:
40 print( "$F FFMPEG SCRIPT"
41 print( "$F PYTHON makeff1.py FILENAME"
42 print( "$F"

https://github.com/kieranjol
elif len(checksum_mismatches) > 1:
    print 'NOT LOSSLESS'
    append_csv(
        csv_report_filename,
        {  
            'output', 'NOT LOSSLESS',
            'source_video_size', ffmpeg_video_size, 'compression_ratio',
        }
    )
    generate_log(log, 'makeffmpeg.py Not Lossless."
    file_md5, shallow=False):
        print "YOUR FILES ARE LOSSLESS YOU SHOULD BE SO HAPPY!!!"
    else:
        print "The framemd5 text files are not completely identical."
        "This may be because of a lossy transcode,"
        "or a change in metadata, most likely pixel aspect ratio."
        "Please analyse the framemd5 files for source and output."

def main():
    video_files, csv_report_filename = get_input()
    make_ffmpeg(video_files, csv_report_filename)

    if __name__ == '__main__':
        main()
Ashley Blewer

audiovisual preservation training
Minimum Viable Station Documentation

Link to this document: bit.ly/mindigit
Please contribute! Contact Ashley Blewer to request write-access to this document.

- Blog post for doc context
- Minimum Viable Station as Diagram
- Minimum Viable Station Recipes

Table of Contents

Opens examples

Hardware

Computer

Analog-to-digital converter

Time Base Corrector (TBC)

Capture Software

Disk Imaging Software (for Optical Media)

Packaging/Fixity

Quality Control Software

Metadata Software

Decks

Cables

Film

Storage

Misc.

License

Minimum Viable Workstation... Recipes!

Shopping lists for workstations at every budget!

Please contribute! Contact Ashley Blewer to request write-access to this document.

For more details for each component of the workflow, please see the original Minimum Viable Station document.

Don't forget you will need to also budget for small incidentals like cleaning supplies, tables/storage/chairs, WD-40, headphones, and power cables.

Table of Contents

Ain't Got No ($) 

Preservation-grade Bells and Whistles ($$)

Saving High School Mixtapes ($) 

Ain't Got No ($)

Description:
"If you build it, they will come." Sometimes you have to get started with what you have and upgrade later.

Setup:
- Your Computer (Any OS, laptop or desktop) | Free or around $300 (minimally, could easily be more)
- Instead of a computer, you could consider using a Raspberry Pi ($30)
- VHS Deck from your Friend's Parent's Attic (Consumer-grade) | Free or around $30
- Video8/Hi-8 Camera from your Friend's Parent's Attic (Consumer-grade) | Free or around $150
- MiniDV Camera from your Friend's Parent's Attic (Consumer-grade) | Free or around $130
- ElGato Video Capture (includes hardware, cable, and software) | $75

Total (with found/borrowed equipment): $75
Total (with dedicated equipment): around $685

Preservation-grade Bells and Whistles ($$$)

https://bits.ashleyblewer.com/blog/2017/01/16/minimum-viable-digitization-station-recipes/
Digital Tape Preservation Strategy: Preserving Data Or Video

By Kari Van Malzen | December 2, 2009 | Papers and Presentations

By David Rice and Chris Lacinak – December 2, 2009

Abstract: This paper examines preservation philosophies and strategies applied to large scale video collections that are both born-digital and tape-based. Technically and philosophically different approaches may be applied to migrating born-digital, tape-based content with decisions ranging from deck selection and choice of output to specifications of the resulting file. At the core of this is the distinction between migrating digital video as an audiovisual signal versus migrating it as data.

Introduction

In trying to conceptualize the issues around the migration of born-digital tape-based content we're challenged to separate our normal associations with videotape and the video/audio signals from the fact that the content is stored digitally on the tape. It looks and acts like a legacy analog videocassette in many ways, but some of the underlying technology is different. This is why we may discuss the "migration", and not "digitalization", of the content to the file-based domain. The content in this specific collection is already digital, born in a compressed DV codec.

Profile of DV tapes (miniDV and DVCam)

Composition: DVCam and miniDV tapes are metal-evaporated tape made with a prioritization to compact size and low cost. The lubricated tape is extremely thin, chemically complex, and susceptible to drop-outs, errors, and data loss. The tape is also finicky as damaged or deteriorated portions of the tape may play back with varying degrees of accuracy from one playback to another or from one deck to another through error correction and concealment strategies.

The Case of DV tape

With DV tape we are once again faced with a physical format that is dependent on certain hardware for playback. However, DV is a published standard that is well documented and is supported by both major hardware and software manufacturers.

Still the conflation of a digital signal and a tape-based video carrier brings about confusion on how to approach migration. The hardware for playback offers multiple types of signal outputs. The selection of output identifies a key question. Should the information on the tape be treated purely as audiovisual information similar to legacy video formats or should it be treated as stream of data? The fact is that DV tape contains a much greater amount of information beyond the audiovisual signal.

A DV stream may contain:

- Video (NTSC or PAL)
- Audio (48, 48.1, or 33 Hz 2 or 4 track)
- Metadata from the Source Tape
  - TimeCode
  - Closed Captioning (as auxiliary data, not video line 20)
  - Camera Metadata (iris, gain, white balance, etc.)
  - Original Recording Date and Time
- Metadata from Device Read (Occurences during playback)
- "Status" data (video error concealment, audio errors, etc.)

Selecting the firewire output of a DV deck treats the DV tape as data and migrates all of the above data in its native format. Selecting the Serial Digital Interface (SDI) output from a DV deck treats the DV tape like a legacy videotape discarding a great deal of metadata, decompressing the video and blocking information on whether or not the deck used error concealment or produced a drop-out.

You may be asking, why does this other metadata matter?

1. Authenticity: DV attaches metadata to every recorded frame that can tie that frame to its place in the production chain. Inconsistencies in the timestamp, timecode, and camera information that occur during filming or editing can be identified and tracked.

2. Error Identification: As addressed above, errors or concealments that occur during playback can be permanently encoded into the migrated data stream. A bit-for-bit transfer captures metadata that describes the error-related processes within the deck. This data can be analyzed for causation and possible solutions to determine if a second transfer is warranted or to identify potential issues in the workflow. Further information about different kinds of errors can be found under the DV Analyzer: Case Studies heading.

3. Efficiency: Because so much information about the digital transfer itself is encoded in the stream, that data can be leveraged to perform targeted efficient analysis and quality control with tools such as Live Capture Plus and DV Analyzer. During a high-throughput migration of content, quality control must be selective and focused. Putting on headphones and viewing every sample may not be practical or affordable. Tools that solely analyze the decoded audiovisual playback may play a useful role but may also under- or over-report on errors through estimations rather than responding to quality control information that is explicitly documented in the stream output of a FireWire cable. If the tools managing the migration of the content can provide information about the work being done, especially the parts of the migration pertinent to errors, we can narrow our quality control to areas where the errors are known.
Aviary
An audiovisual content publishing platform. Aviary provides controlled access to intended users with search, navigation, and playback across audio and video archives, collections, and files. Pinpointing search results with playback exactly where a search term is found.

Catalyst Inventory Solution
Catalyst is a solution for creating better inventories and managing digitization workflows. It transforms unknowns into quantifiable data for better decision making, more effective resource allocation, item access and management, and planning and strategizing.

AVCC
AVCC makes collaborative, efficient item-level cataloging of audiovisual collections possible over the web. Built-in reporting on critical data enables users to uncover hidden collections, support preservation reformulation, and expand access to content.

Exactly
Exactly is an easy-to-use and in-demand application for securely and remotely sending and receiving digital data. Exactly allows recipients to create customized mailable templates for senders, incorporates data integrity measures, and notifies users when data reaches its destination.

Fixity
Fixity is a simple, powerful and popular application for automated monitoring and reporting on the data integrity of stored files. Fixity provides data for comparative analyses and sends status updates and reports automatically.

MDQC
MDQC stands for Metadata Quality Control, and it allows users to create rules on embedded metadata, scan a set of files, and report on the conformance of each file against the user-defined rules. MDQC greatly minimizes the time needed to QC large batches of files.

https://www.weareavp.com/products/
BBC Archives
Securing the past for the future

Steve Daly
Head of Technology
BBC Information and Archives
1. community
2. open source tools + resources
A complete, cross-platform solution to record, convert and stream audio and video.

Converting video and audio has never been so easy.

```bash
$ ffmpeg -i input.mp4 output.avi
```
# ffmpeg

- Extract the sound from a video and save it as MP3:

```
ffmpeg -i video.mp4 -vn sound.mp3
```

- Convert frames from a video or GIF into individual numbered images:

```
ffmpeg -i video.mp4|video.gif frame_%.png
```

- Combine numbered images (frame_1.jpg, frame_2.jpg, etc) into a video or GIF:

```
ffmpeg -i frame_%d.jpg -f image2 video.mp4|video.gif
```

- Quickly extract a single frame from a video at time mm:ss and save it as a 128x128 resolution image:

```
ffmpeg -ss mm:ss -i video.mp4 -frames 1 -s 128x128 -f image2 image.png
```

- Trim a video from a given start time mm:ss to an end time mm:ss (omit the -to flag to trim till the end):

```
ffmpeg -ss mm:ss -to mm:ss -i video.mp4 -c copy output.mp4
```

- Convert AVI video to MP4. AAC Audio @ 128kbit, h264 Video @ CRF 23:

```
```

- Remux MKV video to MP4 without re-encoding audio or video streams:

```
ffmpeg -i input_video.mkv -c copy output_video.mp4
```

- Convert MP4 video to VP9 codec. For the best quality, use a CRF value (recommended range 15-35) and -b:video MUST be 0:

```
```
NAME

ffmpeg - ffmpeg video converter

SYNOPSIS

ffmpeg [global options] [[input file options] -i input url ...] [output file options] output url ... |

DESCRIPTION

ffmpeg is a very fast video and audio converter that can also grab from a live audio/video source. It can also convert between arbitrary sample rates and resize video on the fly with a high quality polyphase filter.

ffmpeg reads from an arbitrary number of input "files" (which can be regular files, pipes, network streams, grabbing devices, etc.), specified by the "-i" option, and writes to an arbitrary number of output "files", which are specified by a plain output url. Anything found on the command line which cannot be interpreted as an option is considered to be an output url.

Each input or output url can, in principle, contain any number of streams of different types (video/audio/subtitle/attachment/data). The allowed number and/or types of streams may be limited by the container format. Selecting which streams from which inputs will go into which output is either done automatically or with the "-map" option (see the Stream selection chapter).

To refer to input files in options, you must use their indices (0-based). E.g. the first input file is 0, the second is 1, etc. Similarly, streams within a file are referred to by their indices. E.g. "2:3" refers to the fourth stream in the third input file. Also see the Stream selectors chapter.

As a general rule, options are applied to the next specified file. Therefore, order is important, and you can have the same option on the command line multiple times. Each occurrence is then applied to the next input or output file. Exceptions from this rule are the global options (e.g. verbosity level), which should be specified first.

Do not mix input and output files -- first specify all input files, then all output files. Also do not mix options which belong to different files. All options apply ONLY to the next input or output file and are reset between files.

- To set the video bitrate of the output file to 64 kbit/s:

  ffmpeg -i input.avi -b:v 64k -bufsize 64k output.avi

- To force the frame rate of the output file to 24 fps:

  ffmpeg -i input.avi -r 24 output.avi

- To force the frame rate of the input file (valid for raw formats only) to 1 fps and the frame rate of the output file to 24 fps:

  ffmpeg -i -1 input.m2v -r 24 output.avi

The format option may be needed for raw input files.

DETAILED DESCRIPTION

The transcoding process in ffmpeg for each output can be described by the following diagram:

```
  | input |
file--------> demuxer--------> encoded data |
  |       | packets--------> decoder
```
FFmpeg is a powerful tool for manipulating audiovisual files.

Unfortunately, it also has a steep learning curve, especially for users unfamiliar with a command line interface.

This app helps users through the command generation process so that more people can reap the benefits of FFmpeg.
## Change codec (transcode)

- Transcode to deinterlaced Apple ProRes LT
- Transcode to an H.264 access file
- Transcode from DCP to an H.264 access file
- Transcode your file with the FFV1 Version 3 Codec in a Matroska container
- Convert DVD to H.264
- Transcode to an H.265/HEVC MP4
- Transcode to an Ogg Theora

- Convert WAV to MP3
- Generate two access MP3s (with and without copyright)
- Convert WAV to AAC/MP4
## Preservation tasks

- Batch processing (Mac/Linux)
- Batch processing (Windows)
- Check decoder errors
- Check FFV1 fixity
- Create MD5 checksums (video frames)
- Create MD5 checksums (audio samples)
- Create MD5 checksum(s) for A/V stream data only
- Get checksum for video/audio stream
- QCTools report (with audio)
- QCTools report (no audio)
- Read/Extract EIA-608 Closed Captioning

[https://amiaopensource.github.io/ffmprovisr/](https://amiaopensource.github.io/ffmprovisr/)
Preservation tasks

- Batch processing (Mac/Linux)
- Batch processing (Windows)

Create PowerShell script to batch process with FFmpeg

As of Windows 10, it is possible to run Bash via Bash on Ubuntu on Windows, allowing you to use bash scripting. To enable Bash on Windows, see these instructions.

On Windows, the primary native command line program is PowerShell. PowerShell scripts are plain text files saved with a .ps1 extension. This entry explains how they work with the example of a PowerShell script named "newrap-m4p.ps1", which rewraps .m4p files in a given directory to .mov files.

```
#newrap-m4p.ps1 contains the following text:

$inputfiles = ls *.m4p
foreach ($file in $inputfiles) {
    $output = (JoinPath $file | ChangeExtension $file, '.mov')
    ffmpeg -i $file -map 0 -c copy
    $output
}

$inputfiles = ls *.mp4
foreach ($file in $inputfiles) {
    Open the code block:
    $output = (JoinPath $file | ChangeExtension $file, '.mov')
    ffmpeg -i $file
    Carry out the following FFmpeg command for each input file.
    Note: To call FFmpeg here as just "ffmpeg" (rather than entering the full path to ffmpeg.exe), you must make sure that it’s correctly configured. See this article, especially the section “Add to PATH.”
    -map @ retain all streams
    -c copy enable stream copy (no re-encode)
    $output
}

Close the code block:

Note: the PowerShell script (ps1 file) and all .mp4 files to be rewrapped must be contained within the same directory, and the script must be run from that directory.

Execute the .ps1 file by typing "newrap-m4p.ps1" in PowerShell.

Modify the script as needed to perform different transcodes, or to use with ffprobe."

Link to the command: [https://vitaequisencia.github.io/ffmpegtools/index.html#Batch_processing.ps1](https://vitaequisencia.github.io/ffmpegtools/index.html#Batch_processing.ps1)
ffprobe Documentation

Table of Contents

- 1 Synopsis
- 2 Description
- 3 Options
  - 3.1 Stream specifiers
  - 3.2 Generic options
  - 3.3 AVR options
  - 3.4 Main options
- 4 Writers
  - 4.1 default
  - 4.2 compact, csv
  - 4.3 flat
  - 4.4 ini
  - 4.5 json
  - 4.6 xml
- 5 Timecode
- 6 See Also
- 7 Authors

1 Synopsis
ffprobe [options] [input url]

2 Description
ffprobe gathers information from multimedia streams and prints it in human-readable fashion.

For example it can be used to check the format of the container used by a multimedia stream and the format and type of each media stream contained in it.

If a URL is specified in input, ffprobe will try to open and probe the URL content. If the URL cannot be opened or recognized as a multimedia file, a positive exit code is returned.

ffprobe may be employed both as a standalone application or in combination with a textual filter, which may perform more sophisticated processing, e.g. statistical processing or plotting.

Options are used to list some of the formats supported by ffprobe or for specifying which information to display, and for setting how ffprobe will show it.

ffprobe output is designed to be easily parsable by a textual filter, and consists of one or more sections of a form defined by the selected writer, which is specified by the -print_format option.

Sections may contain other nested sections, and are identified by a name (which may be shared by other sections), and an unique name. See the output of sections. Metadata tags stored in the container or in the streams are recognized and printed in the corresponding "FORMAT", "STREAM" or "PROGRAM_STREAM" section.

3 Options

All the numerical options, if not specified otherwise, accept a string representing a number as input, which may be followed by one of the SI unit prefixes, for example: "K", "M", or "G".

If "i" is appended to the SI unit prefix, the complete prefix will be interpreted as a unit prefix for binary multiples, which are based on powers of 1024 instead of powers of 1000. Appending "b" to the SI unit prefix multiplies the value by 8. This allows using, for example: "KB", "MB", "GB" and "TB" as number suffixes.

Options which do not take arguments are boolean options, and set the corresponding value to true. They can be set to false by prefixing the option name with "no".

For example using "--nofoo" will set the boolean option with name "foo" to false.

Metadata:
    major_brand : mp42
    minor_version : 0
    compatible_brands: isomiso2mp41
    creation_time : 2015-09-02T11:40:58.000000Z
    encoder      : FFmbc 0.7

Duration: 01:01:43.16, start: 0.000000, bitrate: 1021 kb/s
Stream #0:0(und): Video: mpeg4 (Simple Profile) (mp4v / 0x7634706d), yuv420p, 512x288 [SAR 1:1 DAR 16:9], 900 kb/s, 25 fps, 25 tbr, 25 tbn, 25 tbc (default)
    Metadata:
        creation_time : 2015-09-02T11:40:58.000000Z
        handler_name : VideoHandler

Stream #0:1(und): Audio: aac (LC) (mp4a / 0x6134706d), 48000 Hz, stereo, fltp, 116 kb/s (default)
    Metadata:
        creation_time : 2015-09-02T11:40:58.000000Z
        handler_name : SoundHandler

"streams": [
    {
        "index": 0,
        "codec_name": "mpeg4",
        "codec_long_name": "MPEG-4 part 2",
        "profile": "Simple Profile",
        "codec_type": "video",
        "codec_time_base": "1/25",
        "codec_tag_string": "mp4v",
        "codec_tag": "0x7634706d",
        "width": 512,
        "height": 288,
        "coded_width": 512,
        "coded_height": 288,
        "has_b_frames": 0,
        "sample_aspect_ratio": "1:1",
        "display_aspect_ratio": "16:9",
    
https://ffmpeg.org/ffprobe.html
Introduction

Within digital preservation environments, the generation and verification of checksums against digital files can aid in the confirmation or denial of digital authenticity over time. A checksum mismatch is an alert that a file under care has changed from a prior state; potentially triggering retrieval of backups, review of hardware, or migration of content.

Generally, if a given checksum algorithm is applied to a file, then as long as the same checksum can be regenerated from the file then the data is verified, else a mismatched checksum reveals a digital change. Further details such as the whereabouts, extent, or significance of the change in data are not revealed by the checksum mismatch but only that the data examined now is not the same as the data examined before.

The FFmpeg \texttt{=fammad5} format and \texttt{=framecrc} format as used to decode input audiovisual data to produce one checksum per frame. These formats facilitate testing functions such as verifying that an adjusted decoder maintains intended results or that an FFmpeg decoder decodes a stream to the same data as another decoder.

By producing checksums on a more granular level, such as per frame, it is more feasible to assess the extent or location of digital change in the event of a checksum mismatch. By decoding a file and processing the decoded data to generate a framesd5 document, each decoded audio and video frame is documented according to its timestamp, digital size, and MD5 checksum. For the first three frames of video, the framesd5 output could be:

```
#b 0: 1001/24000
 0, 0, 6, 3, 518400, 3c1a4a1a7548d8b34d7819d908a3be78a
 0, 1, 1, 518400, 3c1b4a19d6e0d3158a75d7e6601f6f6
 0, 1, 2, 518400, 3c0e4d4c4648d73b580d1e55da55a435
```

In this output the columns refer to the stream number, counting from zero, (column 1), the decoding and presentation timestamps (column 2 and 3), the samples duration (column 4), the size of the data checksummed in bytes14 (column 5), and the MD5 checksum for that data.

Storing a framesd5 file along with each audiovisual file does not replace the function of a traditional whole-file checksum. It is still possible for a file to be changed in a way that would result in a mismatch for a whole-file checksum analysis, but not create any difference between a stored framesd5 output and a newly created framesd5 output. This could occur when embedded metadata is edited but the stored audiovisual data remains the same.

For audiovisual data, storing both a whole-file checksum and a framesd5 output enables greater awareness of digital change in managed files, a more strategic and aware response to change, and the ability to verify lossless transcoding. If an audiovisual file is found to have a mismatch between a newly generated whole-file checksum and one generated previously, indicative of digital change, then comparison between a stored framesd5 document and a newly generated one could facilitate in pinpointing the digital change as it affects audiovisual presentation if at all.
Reconsidering the Checksum for Audiovisual Preservation

This article was authored by Dave Rice and initially published in the IASA Journal number 29 under a CC-BY-ND license.

Reconsidering the Checksum for Audiovisual Preservation:

Detecting digital change in audiovisual data with decoders and checksums

Firstly, what are checksums for?

A checksum is a small data value computed from a given amount of data, such as a file or bitstream, for the purpose of facilitating the future ability to detect changes in that given data. The generation and verification of checksums for digital archival holdings is a central principle of digital preservation and enable archivists to trust that data held within an archive is the same data that was received by the archive. Although checksum wrangling is typically a behind-the-scenes process within digital storage systems and repositories, these values are worth a closer look. The checksum value is generally expressed in hexadecimal representation (aka base 16) comprised of the numbers 0 through 9 and the letters A through F. Several checksums algorithms, such as CRC32, MD5, and SHA-1, have been introduced offering varying degrees of processing efficiency, security, and collision resistance. As an example, the CRC32 checksum value for a file that contains the ASCII characters “checksum” would be d6f6f9a and the MD5 value for the same file would be 22619b9d4b21d1b8c7b1ab42d85e419d. If the text file changed, whether through manipulation, bit rot, or data corruption, then further evaluations of the file would produce a different checksum value. The mismatch of a newly calculated checksum and a stored checksum produced earlier is an alert that data under care has been changed.

Verifying Lossless Transcoding

The use of lossless codecs alone does not guarantee that the resulting encoded lossless audiovisual file could be used to reconstruct the original audiovisual data. A preservation-suitable lossless audiovisual encoding should decode to the same data that the original source would decode to, meaning that each pixel, frame, and timing decoded from the lossless version should be the same as the decoded original.

An original uncompressed digital audiovisual file called “uncompressed.mov” could produce this framemd5 output (the first four video frames are listed):

0, 0, 0, 1, 518400, 6db88be7dbcc426be8669992cc74f1d0
0, 1, 1, 1, 518400, 6db88be7dbcc426be8669992cc74f1d0
0, 2, 1, 1, 518400, 2eb3ada5b42d5c6b21e3b6e7e267414
0, 3, 1, 1, 518400, e57d03e253f3997ae2db462a28d8c749

Let’s imagine that an archive decides to transcoding the uncompressed file to a lossless codec in order to reduce storage requirements. This ffmpeg command generates a lossless ffv1 encoding from the original file and copies the audio data as-is.

ffmpeg -i uncompressed.mov -map @ -c:v ffv1 -c:a copy lossless.mov

Generating a framemd5 report on the “lossless.mov” should produce the same output because both files, although utilizing different codecs, both decode to identical audiovisual presentations. If the two files do not decode to produce identical framemd5 documents then it is likely that the transcoding from the uncompressed codec to the lossless codec was not truly mathematically lossless.
MediaInfo

MediaInfo is a convenient unified display of the most relevant technical and tag data for video and audio files.

Download MediaInfo

Version 18.12, Graphical User Interface with installer, Mac App Store
Other versions (packaging, OS, interface...) are also available.

See change log

You can also use MediaInfo Online to test MediaInfo without installing any software on your computer.

The MediaInfo data display includes:

- Container: format, profile, commercial name of the format, duration, overall bit rate, writing application and library, title, author, director, album, track number, date, duration...
- Video: format, codec id, aspect, frame rate, bit rate, color space, chroma subsampling, bit depth, scan type, scan order...
- Audio: format, codec id, sample rate, channels, bit depth, language, bit rate...
- Text: format, codec id, language of subtitle...
- Chapters: count of chapters, list of chapters...

MediaInfo analytics include:

- Container: MPEG-4, QuickTime, M4A, AVI, MPEG-PS (including unprotected DVD), MPEG-TS (including unprotected Blu-ray), MXF, XCF, XAVC, WMV, FLV, Real...
- Tags: Id3v1, Id3v2, Vorbis comments, APE tags...
- Video: MPEG-1/2 Video, H.263, MPEG-4 Visual (including DivX, XviD), H.264/AVC, H.265/HEVC, FFV1...
- Audio: MPEG Audio (including MP3), AC3, DTS, AAC, Dolby E, AES3, FLAC...
- Subtitles: CEA-608, CEA-708, DTVCC, SCTE-20, SCTE-128, ATSC/53, CDP, DVB Subtitle, Teletext, SRT, SSA, ASS, SAMI...

MediaInfo features include:

- Read many video and audio file formats
- View information in different formats (text, sheet, tree, HTML,...)
- Customise these viewing formats
- Export information as text, CSV, HTML...
- Graphical user Interface, command line interface, or library (.dll/.so/.dylib) versions available
- Integrate with the shell (drag 'n' drop, and Context menu)
- Internationalisation: display any language on any operating system
- Localisation capability (for which volunteers are needed - please contact us)
MediaConch is an extensible, open source software project consisting of an implementation checker, policy checker, reporter, and fixer that targets preservation-level audiovisual files (specifically Matroska, Linear Pulse Code Modulation (LPCM) and FF Video Codec 1 (FFV1)) for use in memory institutions, providing detailed and batch-level conformance checking via an adaptable and flexible application program interface accessible by the command line, a graphical user interface, or a web interface.

Download MediaConch

Version 18.03.2, for Ubuntu

Other versions (packaging, OS, interface...) are also available

See ChangeLog or very latest snapshots

You can also use MediaConchOnline to test MediaConch without installing any software on your computer.

Please donate to support further development
QCTools

QCTools is a software tool that helps users analyze and understand their digitized video files through use of audiovisual analytics and filtering.
Funded by the National Endowment for the Humanities and the Knight Foundation; designed and led by Dave Rice and the Bay Area Video Coalition; developed by MediaArea, Fabio Utzig, Alexander van Scharnhorst.
MediaArea was involved in the initial development and provides binaries for all platforms.

Download QCTools

Version 1.0, for macOS
Other versions (packing, OS, interface...) are also available
See ChangeLog or very latest snapshots

Please donate to support further development

QCTools graphing features:

QCTools offers a variety of graphing features including:
- YUV Values
- Temporal Outliers (TOUT)
- Vertical Line Repetitions (VREP)
- Broadcast Range (BRNG)
- Crop Width and Height (CropW and CropH)
- Peak Signal to Noise Ratio (PSNR)
- Mean Square Error (MSE) differences per frame.
- ...Learn more...

QCTools playback filters:

The QCTools preview window is intended as an analytical playback environment that allows the user to review video through multiple filters simultaneously. The playback window includes two viewing windows which may be set to different combinations of filters.
- Histogram
- Waveform
- Waveform Target
- Line Select
- Vectorscope
- Vectorscope Target
- Extract Planes Equalized
- Extract Planes UV Equalized
- Bit Plan
- Bit Plane Noise
- Value Highlight
- Saturation Highlight
- Vectorscope
- ...Learn more...
1. community
2. open source tools + resources
3. open, standards-based archival file formats
ffv1 + matroska

codec + container

an open, standards-based solution for moving image digital preservation
**FFV1 and Matroska reading list**

**DOWNLOAD HERE [PDF]**

This is a collection of web links for further reading on the combination of FFV1 (codec) and Matroska (container), as an emerging open and standards-based digital preservation solution for the digitisation of video and film materials. This combination is going through a standardisation process via the Internet Engineering Task Force (IETF) mandated by the Preforma project, and is already in use in many archives as the preservation solution for video-source digitisation. It is also under active development as a lossless preservation solution for film scans, via the RAWcooked initiative.

Collated for FIAF by Stephen McConnachie, Head of Data, BFI, March 2018.

**FFV1 and Matroska reading list**

A collection of web links for further reading on the combination of FFV1 (codec) and Matroska (container), as an emerging open and standards-based digital preservation solution for the digitisation of video and film materials. This combination is going through a standardisation process via the Internet Engineering Task Force (IETF) mandated by the Preforma project, and is already in use in many archives as the preservation solution for video-source digitisation. It is also under active development as a lossless preservation solution for film scans, via the RAWcooked initiative.

**FFV1 in Wikipedia, includes list of archives known to be using FFV1:**

- [https://en.m.wikipedia.org/wiki/FFV1](https://en.m.wikipedia.org/wiki/FFV1)
- [https://en.m.wikipedia.org/wiki/Matroska](https://en.m.wikipedia.org/wiki/Matroska)

**CASE STUDY:** University of Indiana MDP video digitisation project, file format discussion, by Mike Casey:

- [https://mdpi.icu/ncdo/MDPwhitepaper.pdf](https://mdpi.icu/ncdo/MDPwhitepaper.pdf)

**PREFERENCES:** No Time To Wait: Open Source and FFV1 MKV for archives (blogs, videos, interviews, slides):

- [https://www.beeldengeslacht.nl/knowledgen/blog/no-time-to-wait-preservation-ffv1-matroska-symposium](https://www.beeldengeslacht.nl/knowledgen/blog/no-time-to-wait-preservation-ffv1-matroska-symposium)
- [https://github.com/preforma/mkvmatroska](https://github.com/preforma/mkvmatroska)
- [https://blogin.bkg.de/2015/05/06/improving-technical-approach-for-audiodigital-collections-through-the-preforma-project](https://blogin.bkg.de/2015/05/06/improving-technical-approach-for-audiodigital-collections-through-the-preforma-project)

**FFV1 as preservation choice - implementation risk assessments, by Peter Bulsterling:**

- [https://www.av-ri.nl/knowledgen/video/ Risk_assessment.html](https://www.av-ri.nl/knowledgen/video/Risk_assessment.html)
- [https://mediaakes.net/MediaConch/interviews/interview-exwit steward/subs.html](https://mediaakes.net/MediaConch/interviews/interview-exwit steward/subs.html)
- [http://slidertextwerkstatt.com/file/plab/presentationen/2013%e2%80%939e%e2%81%93 less.pdf](http://slidertextwerkstatt.com/file/plab/presentationen/2013%e2%80%939e%e2%81%93 less.pdf)
- [http://slidertextwerkstatt.com/file/plab/movie3D%2Fa%2Ddigital%2Dvideo%2Darchiving.html](http://slidertextwerkstatt.com/file/plab/movie3D%2Fa%2Ddigital%2Dvideo%2Darchiving.html)

**Library of Congress blog: comparing formats for video digitisation:**

- [https://blogocs.ko.gov/thewave/2014/12/comparing-formats-for-video-digitisation/](https://blogocs.ko.gov/thewave/2014/12/comparing-formats-for-video-digitisation/)

**FFMPEG:** FFV1 encoding reference:

- [https://trac.ffmpeg.org/wiki/Encode/FFV1](https://trac.ffmpeg.org/wiki/Encode/FFV1)

**FFMPEG:** FFmpeg rewrite to MKV using FFV1 with Frame0D:

- [https://smartopensource.github.io/ffmpegtools/ffmpegtools_create_FPV1.mkv](https://smartopensource.github.io/ffmpegtools/ffmpegtools_create_FPV1.mkv)

**MediaConch:** conformance checker for FFV1 / Matroska:

- [https://mediaakes.net/MediaConch/documentation/FAQ.html](https://mediaakes.net/MediaConch/documentation/FAQ.html)

**MKVTools:** toolkit for manipulating Matroska files, including adding and extracting attachments, editing metadata headers, adding chapters with annotations:

- [https://mkvtools.net/download/](https://mkvtools.net/download/)

**Library of Congress file formats for video digitisation:**


**New South Wales Govt.: preservation file formats assessed:**


**FIXITY:** Frame0D article, by Dave Brosie:


**FILM SCANS:** FFV1 MKV as solution for film and video sources, SIAF JPF article by Risto Kromer:


**FILM SCANS:** FFV1 MKV solution discussed by Kieran O’Leary from SIPI:

**FFV1 Video Codec Specification**

by Michael Niedermayer <michaeln@gmx.at>

1. Introduction

2. Terminology and Conventions

3. General Description

4. Video Format

5. Color Format

6. Coding of the Sample Difference

7. Syntax

8. Configuration Record

9. Encoder Configuration Record or Parser

10. Generic FFV1 and Matroska

11. Interfaces to the Sample Container

12. Encoder Code

13. Decoder Code

14. Decoder Interface

15. Table Values, Syntax

16. Picture Structure

17. Pixel Data

18. Color Dictionary

19. Series 1: Primary Color

20. Series 2: Secondary Primary Color


22. Series 4: Sex

23. Series 5: Private Extension

FFV1 is a free, open source video codec designed for low latency streaming.
ffv1 is lossless <and> compressed

<table>
<thead>
<tr>
<th>#tb 0: 1001/30000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 0, 0, 1, 1669440, 1fb241f7b9bb14abfd8ad5b34bdc21</td>
</tr>
<tr>
<td>0, 1, 1, 1, 1669440, 38319375ae195c17016a260ad9953d0</td>
</tr>
<tr>
<td>0, 2, 2, 1, 1669440, c154e23ef725c06a9dc0e1c6bed204b0eae</td>
</tr>
<tr>
<td>0, 3, 3, 1, 1669440, 50b8df0d17ff6e54695417627156f02ed5f6a6</td>
</tr>
<tr>
<td>0, 4, 4, 1, 1669440, 36f5da2b4bceef0973555085e91f7f48da1</td>
</tr>
<tr>
<td>0, 5, 5, 1, 1669440, d36ef15ee854eef2e5e4a9286c91f797b3</td>
</tr>
<tr>
<td>0, 6, 6, 1, 1669440, 31b7232bf8e6f5d7337e2c6edc8b920c42</td>
</tr>
<tr>
<td>0, 7, 7, 1, 1669440, 7ab596a5999d8ed016ac0b8ed513a70</td>
</tr>
<tr>
<td>0, 8, 8, 1, 1669440, 47b93a1d6d801d2e02b5d141f7a6f7ff5</td>
</tr>
<tr>
<td>0, 9, 9, 1, 1669440, b8bd37a678c2302b20e331e8b34e68ee</td>
</tr>
<tr>
<td>0, 10, 10, 1, 1669440, 417b86a68591526ed3e537b8ab0474f</td>
</tr>
<tr>
<td>0, 11, 11, 1, 1669440, a88f26d342e945553009565a7f9549a</td>
</tr>
<tr>
<td>0, 12, 12, 1, 1669440, 9bf6490424ebc2b520d5d2b3398499</td>
</tr>
<tr>
<td>0, 13, 13, 1, 1669440, 7184f36a23719906a0b51e2f3e5e</td>
</tr>
<tr>
<td>0, 14, 14, 1, 1669440, 905b35a7635b5356c5d251d5edf8d0</td>
</tr>
<tr>
<td>0, 15, 15, 1, 1669440, e0f3577d67cbe642d0f7f285b7b4b733</td>
</tr>
<tr>
<td>0, 16, 16, 1, 1669440, d20a192b1a83df3a2616466ef6e5f</td>
</tr>
<tr>
<td>0, 17, 17, 1, 1669440, 84f9e143be1d3f46b0d242e5b6f2de2</td>
</tr>
<tr>
<td>0, 18, 18, 1, 1669440, c1874e6bda65b150d768357a61be6</td>
</tr>
<tr>
<td>0, 19, 19, 1, 1669440, d86e9e104ec5190b9582c527c3e69</td>
</tr>
<tr>
<td>0, 20, 20, 1, 1669440, ed37e947612d86f2a7effdbb53b6d0</td>
</tr>
<tr>
<td>0, 21, 21, 1, 1669440, 78e3a53eb3e88029c432b47c4045fe</td>
</tr>
<tr>
<td>0, 22, 22, 1, 1669440, 3671f0d799889c7343a9b6ee2c95</td>
</tr>
<tr>
<td>0, 23, 23, 1, 1669440, 3650ccf2013caba2d8a906f72f8f2</td>
</tr>
<tr>
<td>0, 24, 24, 1, 1669440, de9b7e846be1ed555dfdbfd16737337d4</td>
</tr>
<tr>
<td>0, 25, 25, 1, 1669440, 3ab9a6b1b30b79ef2396d9558ec9</td>
</tr>
<tr>
<td>0, 26, 26, 1, 1669440, e052ff48b4c48412264b1b460baf16</td>
</tr>
<tr>
<td>0, 27, 27, 1, 1669440, d463cfe682b7c443b6b2713c550</td>
</tr>
<tr>
<td>0, 28, 28, 1, 1669440, 29f62c97e17387f939a1b4535b28de</td>
</tr>
<tr>
<td>0, 29, 29, 1, 1669440, 7eb1d7180b17a90b17e3a891035088</td>
</tr>
</tbody>
</table>

50gb v210 mov

→ ffmpeg

22gb ffv1 mkv
ffv1 + matroska are standards-based

### Codec Encoding for LossLess Archiving and Realtime transmission (cellar)

<table>
<thead>
<tr>
<th>W3G</th>
<th>Name</th>
<th>Codec Encoding for LossLess Archiving and Realtime transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WG</th>
<th></th>
<th>Codecs Encoding for LossLess Archiving and Realtime transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aname</td>
<td></td>
<td>ffv1 + matroska</td>
</tr>
<tr>
<td>State</td>
<td></td>
<td>Active</td>
</tr>
<tr>
<td>Charter</td>
<td></td>
<td>charter-cellar-01-01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Document dependency graph (SVG)</td>
</tr>
<tr>
<td>URL</td>
<td></td>
<td><a href="https://datatracker.ietf.org/wg/cellar/about/">https://datatracker.ietf.org/wg/cellar/about/</a></td>
</tr>
</tbody>
</table>

### Charter for Working Group

The preservation of audiovisual materials faces challenges from technological obsolescence, analog media deterioration, and the use of propriety formats that lack formal open standards. While obsolescence and material degradation are widely addressed, the standardization of open, transparent, self-descriptive, lossless formats remains an important avenue to be undertaken by the open-source community.

FFV1 is a lossless video code and Matroska is an extensible media container based on EBML (Extensible Binary Meta Language), a binary XML format. These are open source implementations of both formats, and an increasing interest in and support for use of FFV1 and Matroska. However, there are concerns about the sustainability and credibility of existing specifications for the long-term use of these formats. These existing specifications require broader review and formalization in order to encourage widespread adoption.

There is also a need for a lossless audio format to complement the lossless video codec and container format. FLAC is a lossless audio codec that has seen widespread adoption in a number of different applications including archival applications. While there are open source implementations of the codec, no formal standard exists for the codec itself or its use in container formats currently exist. Review and formalization of the FLAC code standard and its use in Matroska container formats is needed for wider adoption.

Using existing work done by the development communities of Matroska, FFV1, and FLAC, the Working Group will formalize specifications for these open and lossless formats. In order to provide authoritive, standardized specifications for users and developers, the Working Group will seek consensus throughout the process of refining and formalizing these standards. Initial specifications can be accessed here:

- FFV1: https://datatracker.ietf.org/wg/cellar/about/
- FLAC: https://datatracker.ietf.org/ietf.specs.html

---

https://datatracker.ietf.org/wg/cellar/about/
ffv1 + matroska are by archivists for archivists

Further standardisation of Matroska and FFv1
IETF 83 Prague, VideoLAN One Day in Paris and much more

Positron on 30 September 2015

Tessa Falb and Jarno Marinheiro, members of Media4all’s team, facilitated much of the interaction between the project and the IETF, which was described as the most appropriate standards body for collaboration with. This interaction led to a decision of whether the IETF should proceed with efforts to further standardise Matroska and FFV1. Attendance at IETF allowed Media4all and PREFORMA to establish relationships with IETF members/introducers who could coordinate in standardisation efforts within IETF process and procedures. Representatives from both formats participate in the meeting and subsequent online discussion were carried out with the IETF-OFFICIAL working group.

The representative-office of cultural heritage organisations that use Matroska and FFV1, such as the Austrian Bundesarchiv and the UK National Archivist also participated in discussions on these formats directly with IETF representatives. Tessa Falb has worked with Matroska, FFV1, and PREFORMA on the request of IETF community to draft and propose a working group charter which is currently under consideration by the IETF's IESG. Prior to the approval and establishment of the working group, much of the work is focused on building relationships and offline coordination and planning.

David Niles gave a presentation in Berlin, Switzerland discussing the PREFORM project and discovered it at GPPAD, a “conference” held in Paris, France. The discussion at VGC included initially many organisers such as Media4all’s work on Matroska, and other open-source groups such as Open Media, the Open Source Foundation, and others. The meeting was a productive experience for sharing to the members of the community, and there will be more opportunities to work on Matroska equivalency with the FFV1-MATROSKA facilitated IETF work on the Matroska specification. This has led to informal discussions of members to assemble the container format making it more self-descriptive and future-friendly. 

Why these formats?

After introductions, the first talk of the symposium was from Erwin Bruggen (Netherlands Sound & Vision), who gave a summary of the PREFORMA project. And with it, the insight and history into the decision-making behind the selection of these open formats and how they compared against other potential options to use in the development of a conformance checker for preservation-grade audiovisual formats. As obvious in this symposium, Matroska and FFV1 (and LPCM) were chosen by PREFORMA. The Internet Engineering Task Force working group, formed last year, adopted Matroska and FFV1 but chose to focus on FLAC.

Steve Uhlemann was able to attend the symposium and the IETF meeting, which is amazing because he is one of the founding developers of the Matroska format. His continual input on the CELLAR listserver and during the conference was absolutely invaluable. By the end of the symposium, Steve also had a thorough understanding of the unique needs of archivists and he was happy to assist in the required mapping work to ensure his format is suitable for this use case.

Run fact: Steve original came up with the Matroska format because he was trying to catch Jacques Chirac, at the time President of France, lying on television. The origins of this format seem very archivally-minded, even if that context was not known or considered at the time.

Introduction

No Time to Wait! Standardizing FFV1 and Matroska for Preservation was a symposium intentionally overlapping with Internet Engineering Task Force’s 96th meeting, held in Berlin. No Time To Wait was held on 18-20 July, 2016 and hosted by Deutsche Kinemathek, Zuse Institute Berlin, and MediaArea. The symposium was designed to bring together audiovisual archivists and audiovisual format designers with a focus on the standardization of a preservation-grade audiovisual file format combination package. The structure of this symposium was contingent heavily on the CELLAR working group and the initial meeting of this working group at IETF and first round of RFCs submitted to the organization.

Photo credit: CC BY-SA Erwin Bruggen

https://www.digitalmeetsculture.net/article/further-standardisation-of-matroska-and-ffv1/

https://mediaarea.net/blog/2016/07/26/No-Time-To-Wait-Preservation-FFV1-Matroska-Symposium
ffv1 + matroska are by archivists for archivists
ffv1 + matroska are by archivists for archivists

MKVToolNix – Matroska tools for Linux/Unix and Windows

About MKVToolNix
MKVToolNix is a set of tools to create, alter and inspect Matroska files under Linux, other Unixes and Windows.

Since May 1st 2003, the Matroska libraries themselves and my Matroska tools are officially available. I urge you to download the software, use it and report any errors you encounter to me. I'm always grateful for good bug reports because I make mistakes and you guys normally find them. Your feedback is important to me.

mkvtoolnix-gui -- a GUI for mkvmerge(1) including a chapter and a header editor

1. Synopsis
mkvtoolnix-gui [configuration-file names|configuration-file names]
1. Description
mkvtoolnix-gui [configuration-file names|configuration-file names]

2. Description
mkvtoolnix-gui is a Qt-based GUI for mkvmerge(1). It also implements mkvmerge(1)'s and mkvpropedit(1)'s functionality and adds extra to cover mkvmerge(1) as well. All settings (e.g. source files, track options etc) can be saved and restored.

mkvpropedit -- Modify properties of existing Matroska files without a complete remux

1. Synopsis
mkvpropedit [options] [source-file] [actions]
2. Description
This program analyses an existing Matroska file and modifies some of its properties. Then it writes those modifications to the existing file. Among the properties that can be changed are the segment information elements (e.g. the title) and the track headers (e.g. the language code, 'default track' flag or the name).

mkvinfo -- Print information about elements in Matroska files

1. Synopsis
mkvinfo [options] [source-file]
2. Description
This program lists all elements contained in a Matroska file. The output can be limited to a list of tracks in the file including information about the codecs used.

https://mkvtoolnix.download/docs.html
1. community
2. open source tools + resources
3. open, standards-based archival file formats
4. automation
Welcome

The Adlib API is a command line tool that can be used to interact with Adlib databases. Using this API you can easily build your own applications through one of the three available methods: Uri requests, Adlib.Data Windows dll or jQuery plugin.

This website offers extensive documentation, including an overview of all API commands and many examples on how to use them. You will also find links to other relevant documentation, downloads of sample code, demos and resources. A forum is also available to help you and share information with other Adlib API users.

Try the Adlib API now!

The following Uri request is an example of a Boolean search to retrieve paintings by Bernadino Fungai or William Etty

http://test2.adlibsoft.com/api/wwwopac.ashx?database=c

© Axiell ALM Netherlands, 2010-2018.

http://api.adlibsoft.com/
PUT imcc/v1/import/record/

Submit record(s) to add/update. In user defined XML form. IMCC will transform this into ImagenRecord(s) using the specified XSL (eXtensible Stylesheet Language) mapping file and add to the database. Records may also be updated with this method. This call will expect the body of the call to have Content-Type "text/xml" or "application/xml".

Resource URL

https://[HOST]/MM2/v3/REST/IMCC/v1/import/record/

Resource Information

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response formats</td>
<td>XML</td>
</tr>
<tr>
<td>Requires authentication?</td>
<td>Yes (user context only)</td>
</tr>
<tr>
<td>Required scope</td>
<td>modifyUsersData</td>
</tr>
</tbody>
</table>

Additional Request Headers

X-Imagen-Pathprefix

optional

The path to the directory where the media you want to ingest is stored. If specified then IMCC will add this to the paths specified in the media object elements, if not specified these paths will be used as is.

The X-Imagen-Pathprefix value should be a fully qualified path and can be a UNC path, local path, HTTP URL or HTTPS URL.

For example, if the Media Element OriginalFile is "image.jpg" and the X-Imagen-Pathprefix is "testing/myimagedir" then StorageService will look for the file in "testing/myimagedir/image.jpg" but if the Media Element OriginalFile is "01234 mp4" and the X-Imagen-Pathprefix is "testing/myvideos/record_" then StorageService will look for the file in "testing/myvideos/record_01234.mp4"

Parameters

Optional parameters should be supplied as query parameters.

Mapping

required
SDK Design Best Practices

Introduction
The purpose of this page is to provide BlackPearl client developers with guidance and tips on how to build an integration to BlackPearl using our Software Development Kits (SDKs). We strongly recommend you use our SDKs instead of using direct connection to BlackPearl's http-based Application Program Interface (API). For additional assistance, please post questions to our Google Group. The intended audience for this page is someone developing code with the BlackPearl SDKs or APIs.

Do NOT Use Traditional Amazon S3 PUT and GET Commands
A BlackPearl client should not use the traditional Amazon S3 PUT Object and GET Object commands to move files to and from BlackPearl. The client should use the Spectra S3 (formerly D3) Bulk Commands. The BlackPearl SDKs use these bulk commands. Use of traditional S3 PUT and GET commands will result in poor performance and/or errors when BlackPearl is used with a tape library. We will only certify integrations using the Spectra S3 commands, and we will only support certified integrations.

Group Files into “Jobs” When Archiving or Restoring
The Spectra S3 bulk commands described above group files into “jobs” when they are transferred. Grouping the files significantly increases the performance of the BlackPearl environment, especially when archiving or restoring with tape. Every effort should be made to put as many files into a job as possible to maximize performance. Single-file jobs should be avoided.

When transferring files in a job, multiple files or file parts must be uploaded in parallel to achieve maximum performance. See the “Bandwidth and Performance” section below.

Determine SDK Approach — Helper Classes
The architecture and workflows of the existing software being integrated to BlackPearl will help determine how best to use the SDKs. Some of the SDKs include “Helper” classes that simplify file movement and optimize performance. The client should use the Helper classes if they are available and will fit in the architecture/workflow.

To determine if the Helper classes can be used, first identify which Software Development Kit (SDK) will be used. The SDKs are available in five languages – Java, .NET/C#, Python, C, and Go.

https://developer.spectralogic.com/
Business Process Model and Notation (BPMN)

is the global standard for process modeling and one of the most important components of successful Business-IT Alignment.

More and more organizations are using BPMN and in more and more universities BPMN is taught as a subject. These are the reasons:

- **Standard**: BPMN is not owned by a certain enterprise but by an institution (OMG), which is already established through other world-wide standards, e.g., UML. The standard is supported by many software products; you are less dependent on any particular vendor’s products.

- **Simplicity**: The principle behind BPMN is rather simple which is why you can start working with this notation very quickly.

- **Power of expression**: If necessary, you can describe precisely how a process functions with BPMN. However, this is more difficult than only roughly describing this process. This way of precise modeling is possible, but not mandatory.

- **Implementation in IT**: BPMN has been primarily developed to support technical implementation of processes (“Process Automation”). The more important the IT is in a company, the more helpful the use of BPMN becomes.

A simple flow in BPMN

Let’s begin our BPMN tutorial with a rather simple process diagram:

![Diagram](https://camunda.com/bpmn/)

This diagram shows a simple process triggered by someone being hungry. The result is that someone must shop for groceries and prepare a meal. After that, someone will eat the meal and have his or her hunger satisfied.

**Best Practice: Naming Conventions**

When naming tasks, we try to adhere to the object-orientated design principle of using the [verb] + [object] pattern. We would say “acquire groceries,” for example, not “first take care of shopping for groceries.”

Events refer to something that has already happened regardless of the process (if they are catching events) or as a result of the process (if they are throwing events). For this reason, we use the [object] and make the [verb] passive in voice, so we write “hunger noticed.” BPMN does not require you to model start and end events for a process - you can leave them out - but if you model a start event, you must model an end event for each path. The same is true for end events, which require start events. We always create our models with start and end events for two reasons: first, that way it’s possible to determine the process trigger, and second, you can describe the final status of each path end. We only sometimes abandon this practice with sub-processes. More on this later.

Pizza Collaboration

This example is about Business To Business Collaboration. Because we want to explicitly model the interaction between a pizza customer and the vendor, we have classified them as “participants”, therefore providing them with dedicated pools.

![Pizza Collaboration Diagram](https://camunda.com/bpmn/)

Please note that there are no default semantics in this type of modeling, which means you can model collaboration diagrams to show the interaction between business partners, but also zoom into one company, modeling the interaction between different departments, teams or even single workers and software systems in collaboration diagrams. It is totally up to the purpose of the model and therefore a decision the modeler has to make, whether a collaboration diagram with different pools is useful, or whether one should stick to one pool with different lanes, as shown in the previous chapter.
BFI legacy data ingest BPMN
1. transcode to ffv1 matroska with framemd5 validation
2. split into single programmes
3. detect aspect ratio and triage into correct autoingest workflow
autoingest

1. validate file against policies
2. ingest using correct transcode workflow
3. confirm bit-perfect storage to data tape x 2
4. delete file from network storage
5. log actions and errors at every stage

- Ingest files to ingest:
  - creates Ingest record if object number not already in Ingest
  - updates Ingest record if object number already in Ingest
  - creates Media record if object number not already in Media
  - updates Media record if object number already in Media

- Ingest successful?
  - Log files will be saved to the autoingest folder in each share, every time autoingest completes processing its files. In each share will show only folders originating from that share, so users focus on their own share's work.

- Ingest failed?
  - Log files will be saved to the autoingest folder in each share, every time autoingest completes processing its files. In each share will show only folders originating from that share, so users focus on their own share's work.

- This autoingest folder structure will be replicated in each share in batch.
- Users must place ingest-ready files into the correct folder, based on media type and transcoding requirements.
- Detailed guidelines to be provided, but briefly:
  - autoingest: ingest based on simple file extension rules, no intelligence around transcoding, 4.3 or 16-bit audio for PPL source
  - proxy: use video folder for moving image other than PPL source
  - proxy: use 4x3 or 16x9 for PPL source
  - store: use audio for film soundtracks digitisation to WAV
  - store: use documents for film image digitisation to PDF / Tiff
  - store: video has no current use case, all video ingest is proxy
Mediatheque at BFI Southbank

The new BFI Southbank Mediatheque has reopened with thousands more titles and a fresh new look.

The Mediatheque is evolving, with constantly growing content, a completely new digital platform and new facilities.

Thanks to continuing digitisation work at the BFI National Archive, the content available on the new Mediatheque service is constantly growing in both depth and breadth.

Beautiful Southwark and Lambeth on screen

As part of the Mayor's Story of London project we explore two vibrant boroughs on BFI Southbank's own doorstep.

Related links

Mediatheque Films around the UK

bfi mediatheque – how it works

1. Every night, the MTQ asks CID: give me all records for newly ingested assets with proper restriction applied.

2. CID sends its records, with media identifiers, and MTQ caches them in its own db.

3. All MTQ assets available to CMS.

4. Curator-approved assets only.

5. Over 15 years, unreviewed, entire content.

6. When a user views an image or video in MTQ, it requests the media from Imagen API.

7. Imagen API gets media from DPL proxy server, sends to MTQ via HTTPS.

8. Collection, Asset, Age rating.

9. Under 15, curated content only.
Questions?

‘Not a question really more of a comment’?