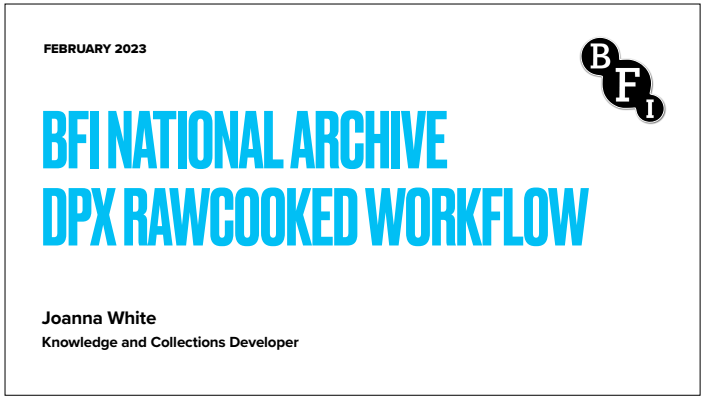
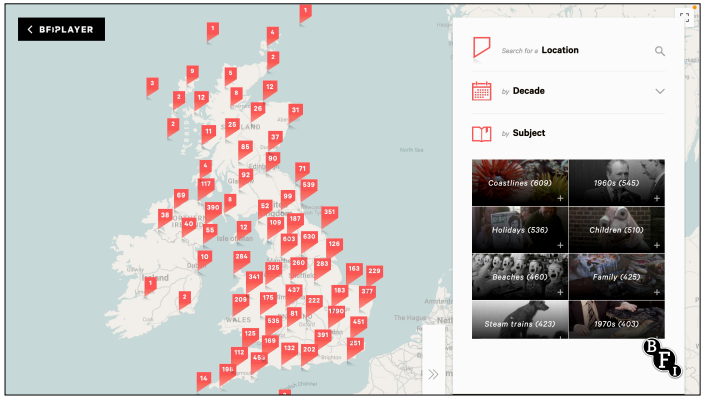
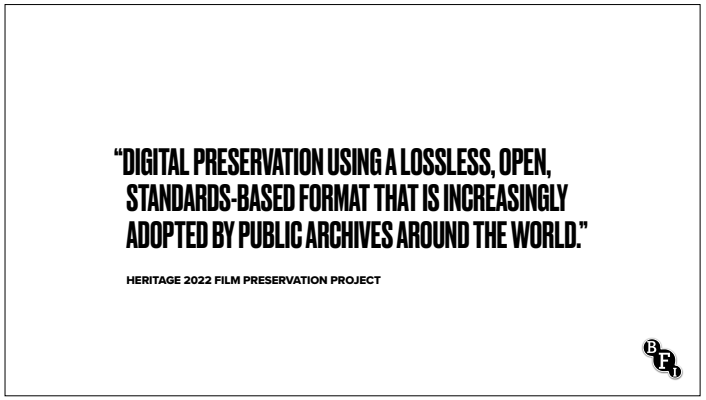


Hello, I'm Joanna White and I'm the Knowledge and Collections Developer at the BFI National Archive. I'm excited to present our DPX workflows to you today, using open source software RAWcooked. Thank you for this opportunity to present!

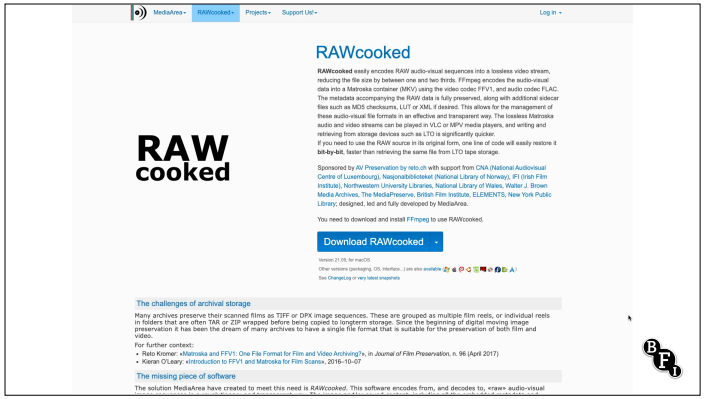


The BFI National Archive began the preservation of 3 peta-bytes of legacy DPX film scans held within its collection in 2019. These scans mostly originated on LTO tape and were generated during the Heritage 2022 Unlocking Film Heritage project.

To give you a sense of the scale of this project, the BFI's Britain on Film map shows footage gathered from across the country. Zooming in on London specifically you can see the many thousands of examples of film available to view for free from the BFI Player website.



The project's aim was 'digital preservation using a lossless, open, standards-based format that is increasingly adopted by public archives around the world.'

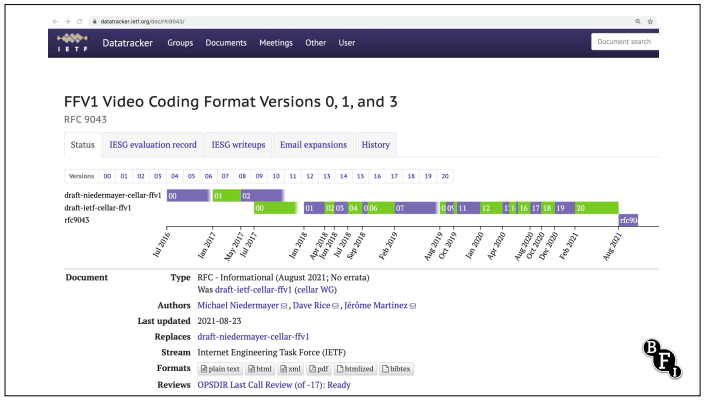


RAWcooked was chosen to encode the DPX sequences to FFV1 video codec in a Matroska container. RAWcooked is created by Media Area, a group of developer archivists led by Jérôme Martinez. This group produce audiovisual tools with preservation interests at their heart.

This software is free to use with a default series of encoding ‘flavours’, including 8-bit and 10-bit RGB DPX. Here at the BFI National Archive we have purchased a few additional encoding flavours including RGB 12-bit and 16-bit, Luma Y 10-bit and 16-bit.

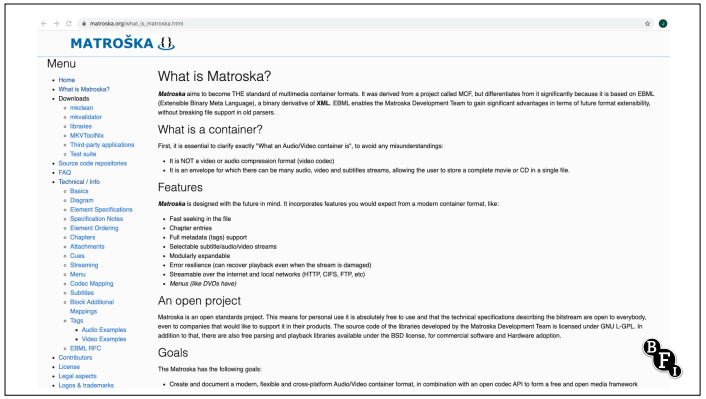
I should add, our DPX workflows do not process audio, but this software can do that very well. Our audio files associated with a DPX sequence are independently processed and ingested so I’m afraid we can’t reflect on this aspect of RAWcooked in this presentation.

https://mediaarea.net/RAWcooked



The FFV1 codec offers amazing features for audiovisual preservation including large file size reductions, frame slices that improve multithreading playback performance, and sliceCRCs, or cyclic redundancy checks, that make it possible for a decoder to detect bitstream errors.

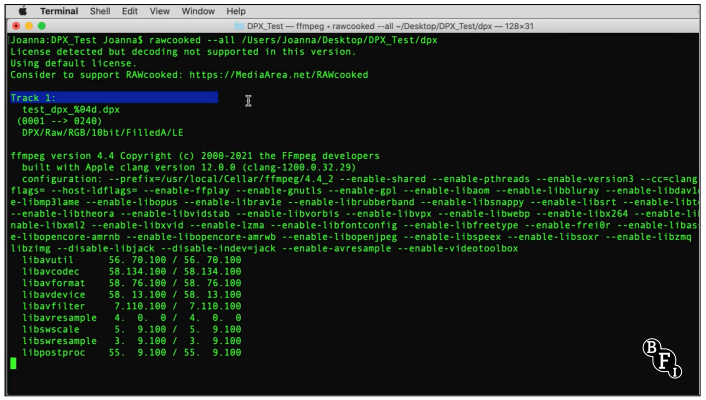
https://datatracker.ietf.org/doc/rfc9043/



FFV1 is the codec, and we use the Matroska wrapper to contain the FFV1 video file and attachments generated by RAWcooked. Both FFV1 and Matroska are proven, open and standards-based solutions created by a talented global audiovisual archiving community.

https://matroska.org/

For those unfamiliar with film, a digitised DPX image sequence has 1 high resolution image for every film's frame, so a 90 minute film at 24fps has almost 130,000 DPX files within it. This can create a sequences as large as 10TB if it's been scanned for 4K distribution. RAWcooked's lossless compression to FFV1 can reduce that overall file size by between one and two thirds on average, which not only saves money on storage solutions but reduces demand on networks. This is most noticeable when moving files to long-term data tape storage taking less time to write to, and retrieve from, tape libraries.



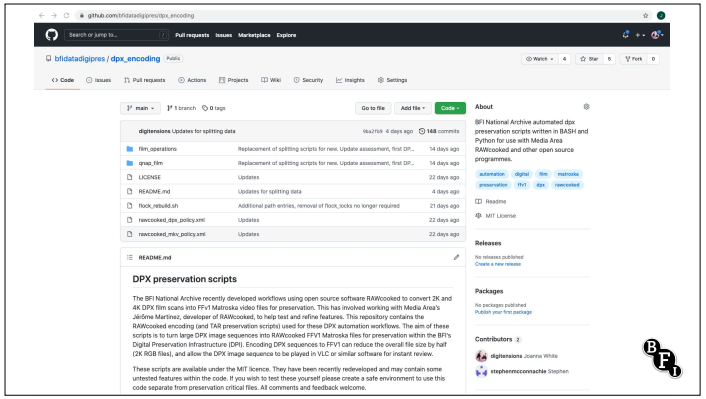
Let's quickly see how easily RAWcooked converts a single image sequence into an FFV1 video stream, using the 'all' command. The 'all' command combines several other important preservation commands, ensuring the highest standard of image sequence encoding. It can be used to transcode an image sequence to FFV1 Matroska, and it can be used to return an FFV1 Matroska back to an image sequence. A bit by bit perfect copy of the original, confirmed in the last statement 'reversibility was checked, no issues detected'.

As we have thousands of sequences to process within a tight project deadline, it was essential that bash shell scripts were written to automate the workflows around the clock. Our workflows started in 2019 with two bash scripts written by Head of Data and Digital Preservation, Stephen McConnachie.

These transcoding scripts maximised our transcoding throughputs by using open source software GNU Parallel which allows us to run transcodes in parallel, with transcoding processes spread efficiently across the threads of our transcoding server.

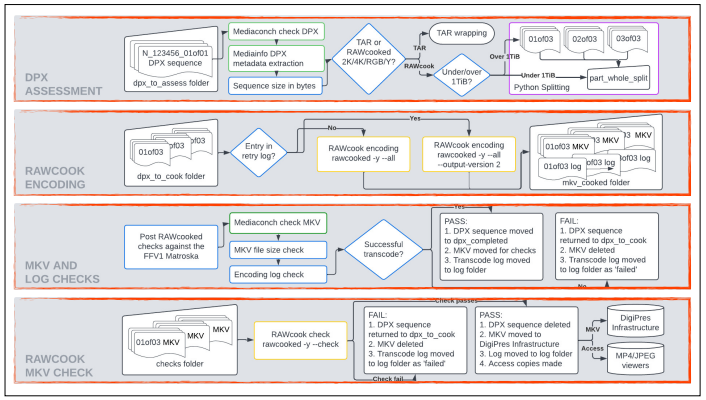
I joined the BFI in 2019 and spent the next two years evolving Stephen's scripts into four bash scripts, and two python scripts. These essentially manage the whole process of DPX assessment, file size management, transcoding to FFv1, conformance checking and deletion of the DPX sequence. There have been very many adjustments to our scripts during this development period, and in turn this has informed many developments within the RAWcooked software itself. The DPX files we have encountered in the project have really tested and expanded RAWcook's capabilities - and coined the term 'wild west DPX' for our legacy collections.

Most recently this workflow has been expanded to include 4K processing for our in house scanners, and we're currently working through huge folders full of DPX data which has seen a slowing in throughput but essential size reductions.



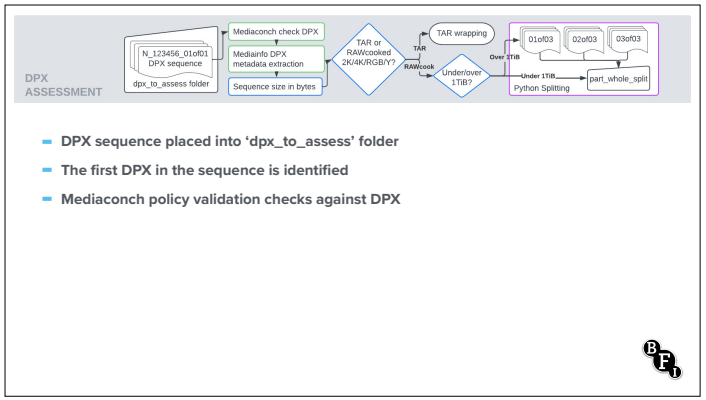
In the spirit of the open-source software and the community that this workflow is inspired by, we feel it's really important to share the scripts in full via our Data and Digital Preservation GitHub pages, which you can see here. You can find out more about all the code used in this workflow here, with detailed descriptions of how each script operates. This is our live code, so any changes made on a day to day basis are reflected immediately in this repository.

We also have recently put together a RAWcooked cheat sheet which details our approaches for optimising our automation processes. Including using GNU parallel for parallisation of transcodes. It also covers the importance of capturing your logs for review in automated transcode workflows.

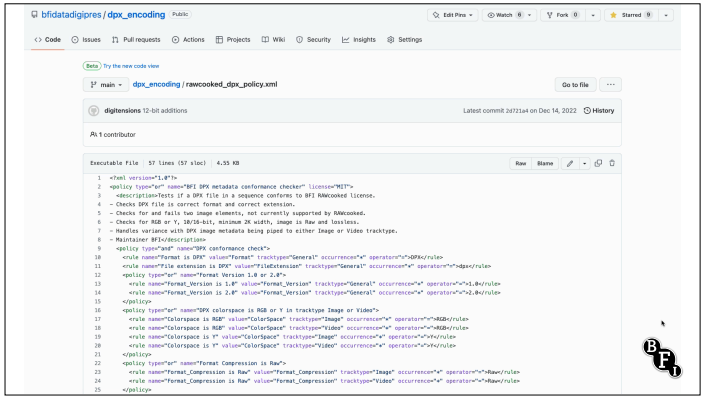


So to give an overview of the workflow I thought it best if we step through it from a DPX sequences perspective.

I've broken the workflow into four stages, the DPX assessments phase, the RAWcooked encoding of the DPX into an FFV1 Matroska, the encode log and Matroska file checks, and finally the RAWcooked checks that trigger DPX deletion.

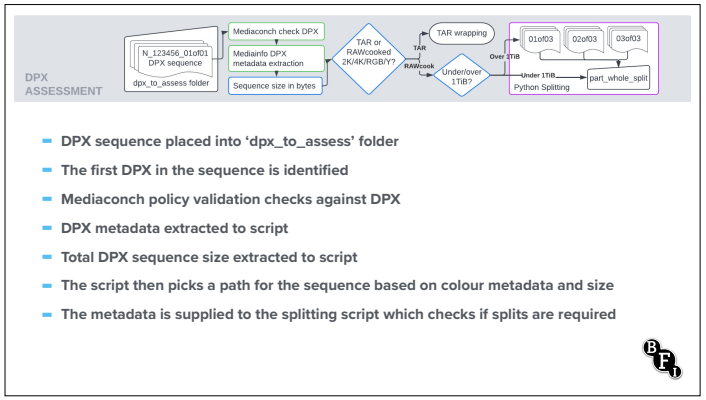


The assessment starts when a DPX sequence is manually placed into the dpx_to_assess folder. This should be the only manual intervention in these workflow scripts. Within that DPX sequence folder the first DPX is identified. This DPX is used to check a few following steps. Firstly, the Mediaconch DPX policy conformance check using Media Area's MediaConch policy checking software.



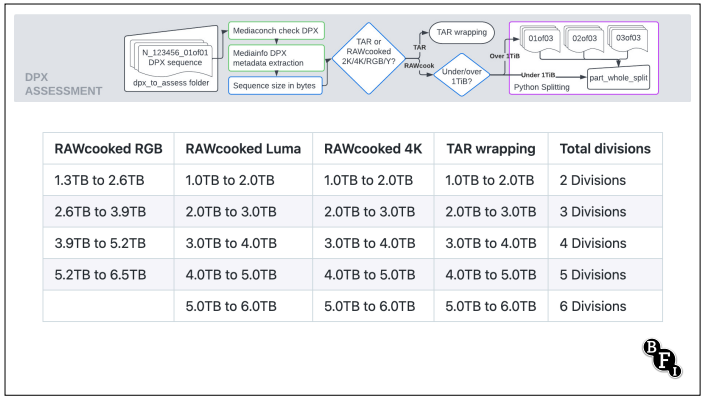
As demonstrated in Michael and Stephen's presentation, a Mediaconch policy will apply certain rules to the assessment of a supplied file, where all rule are found to be true a policy is passed. This is our DPX Mediaconch policy which is kept up to date in our open GitHub repository.

It checks for obstacles that will prevent RAWcooked encoding, such as if there are two image elements in a DPX (caused sometimes by alpha channel inclusion), and if the DPX file conforms to our current RAWcooked license requirements.



Next the scripts extract DPX metadata into the script, and for this we use MediaArea's MediaInfo tool. This DPX metadata is also written into a text file which is stored as an attachment within the finished MKV file's wrapper.

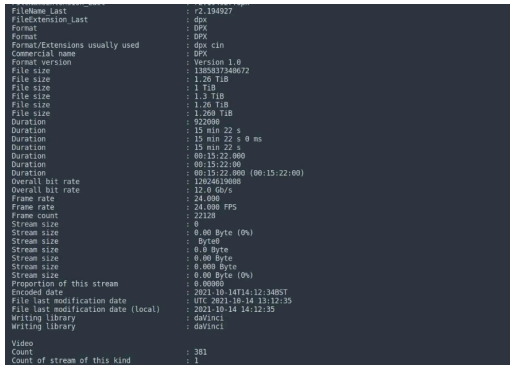
The total DPX sequence size is also extracted. The script checks if the sequence should be RAWcooked or TAR wrapped, and identifies if a files is Luma Y, RGB, 2K or 4K. This data is supplied next to the splitting script which decides if any splitting actions are required.



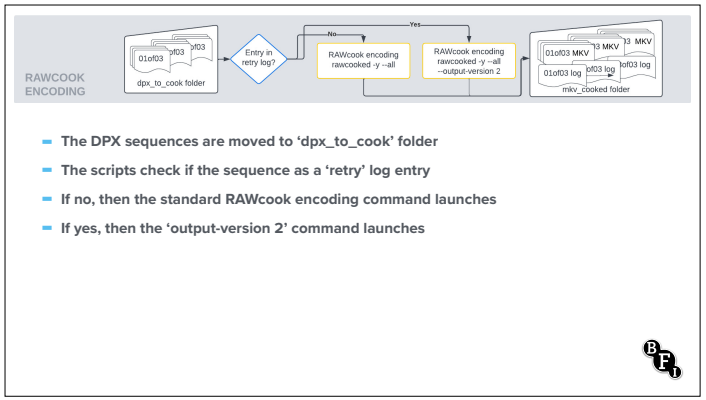
RAWcooked RGB	RAWcooked Luma	RAWcooked 4K	TAR wrapping	Total divisions
1.3TB to 2.6TB	1.0TB to 2.0TB	1.0TB to 2.0TB	1.0TB to 2.0TB	2 Divisions
2.6TB to 3.9TB	2.0TB to 3.0TB	2.0TB to 3.0TB	2.0TB to 3.0TB	3 Divisions
3.9TB to 5.2TB	3.0TB to 4.0TB	3.0TB to 4.0TB	3.0TB to 4.0TB	4 Divisions
5.2TB to 6.5TB	4.0TB to 5.0TB	4.0TB to 5.0TB	4.0TB to 5.0TB	5 Divisions
	5.0TB to 6.0TB	5.0TB to 6.0TB	5.0TB to 6.0TB	6 Divisions

The Python splitting scripts are complicated and manage to many complexities encountered when replicating and sharing massive DPX sequences across multiple, identical folders. If you'd like to know more about how they function please visit our open source GitHub repository, link to follow.

This table shows the splitting size choices applied in the splitting script code. If a file is RAWcooked RGB 2K (the left column), then a DPX sequence can be up to 1.3TB in size before it requires splitting. This is because a 2K RGB sequence will reduce by at least one third when in FFV1 form. This isn't the case for RAWcooked 2K Luma Y or RAWcooked 4K items where the reductions are less predictable, so splits have to happen at the 1TB limit. If a file is over 1TB or 1.3TB then the whole DPX sequence will be split and the DPX files will be moved into identical folders with a filename additional part whole increase to account for the folder addition. So far the scripts can manage files up to 6TB in size, but may require further adjustments if 10TB sequences appear for RAWcooked processing.

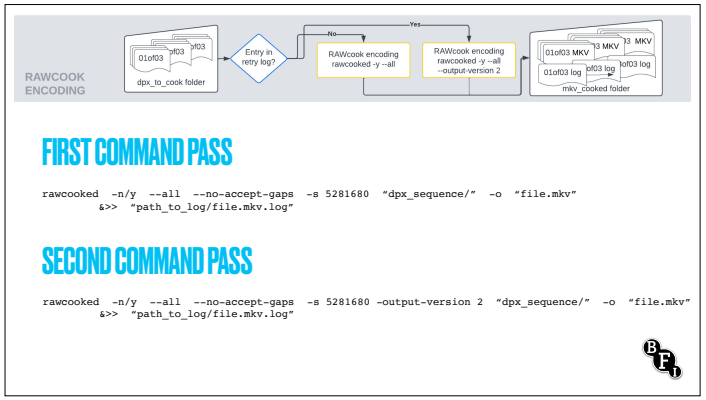


Here's a quick look at an example metadata extraction from a DPX file within a sequence, using MediaArea's MediaInfo software. This is saved into the compressed MKV file as an attachment, and can be extracted from the file at any time using open source software such as MKVtoolnix.



So to the encoding phase. The DPX sequences once split successfully are moved into 'dpx_to_cook' under automation.

The scripts check if the sequence has a failed a previous encoding pass attempt with a 'retry' entry in a special retry log.
- If no, then the standard RAWcooked command launches with '—all' command
- If yes, this is a retry encoding then the all command has an additional 'output version 2' flag. We need this retry command to manage DPX sequences that have extra data in the DPX padding, where usually zeros are found. It ensures this extra data is safely stored in a RAWcooked reversibility attachment so the sequence can be perfectly rebuilt from the FFV1.

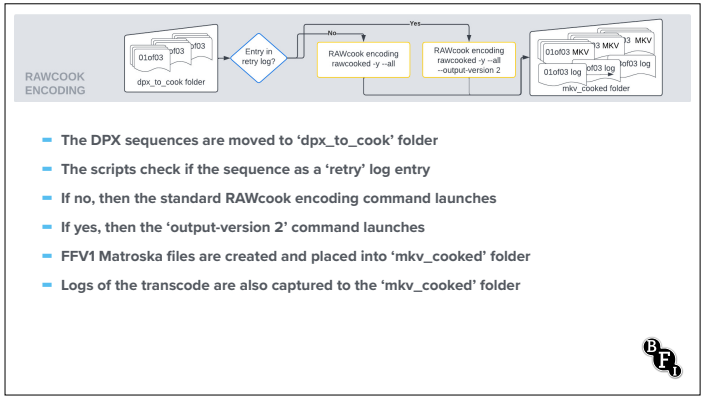


These are the two RAWcooked command currently in use. As you can see these are identical except one has an 'output-version 2' option.

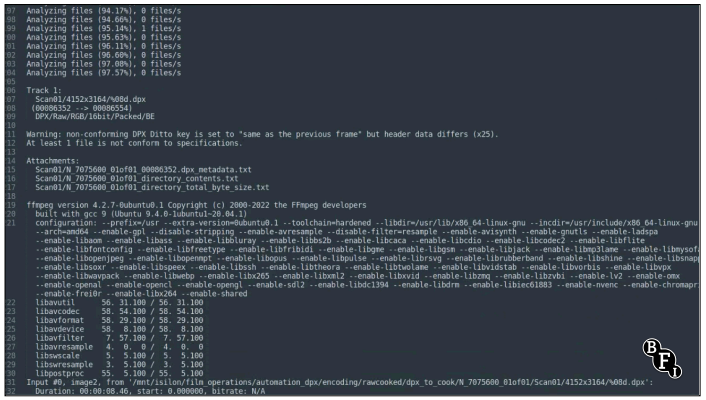
rawcooked: command launched the RAWcooked software
-n / -y: answer no or yes to any enquiries that RAWcooked/FFmpeg asks
- -all: the excellent preservation command new to RAWcooked v21.
- -no-accept-gaps: a flag we use to stop gaps passing through without failing the encoding when the '-n' flag is active. Essential in our new 4K workflows, but where we're working with legacy DPX files from LTO tape we leave the flag '-y' to allow

gaps to pass through.
-s 5281680: sets the maximum attachment size limit up to 5MB. The default in the code is 1MB and too small for our attachments.
- -output-version 2: in the second command ensures that files with large padding data are managed.
‘dpx_sequence’ -o ‘file.mkv’: is the input sequence, and output file name for the new Matroska
&>> ‘path_to_log/file.mkv.log’: ampersand and two right arrows ensures all console outputs are saved into a log file, listed in the path following. This captures

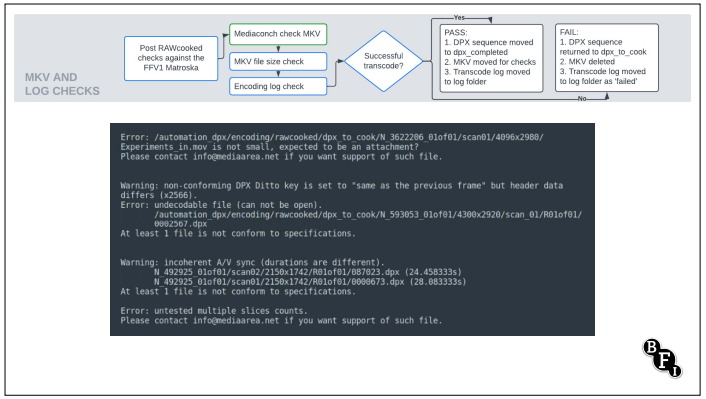
---



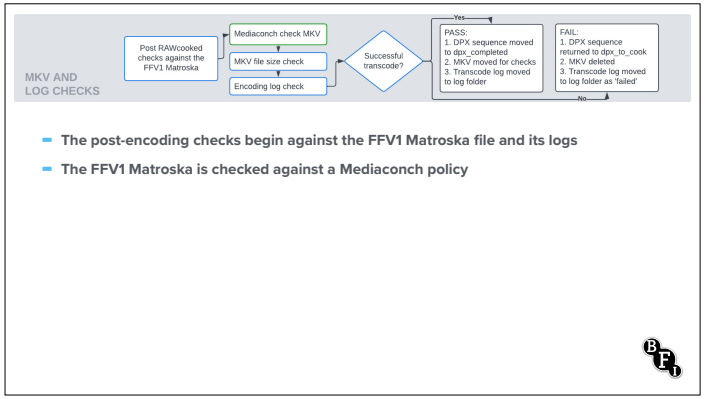
The encoding begins and an FFV1 Matroska file is created and placed into ‘mkv_cooked’ folder. Logs are also captured to the ‘mkv_cooked’ folder. These logs contain RAWcooked and FFmpeg messages about the transcoding process.



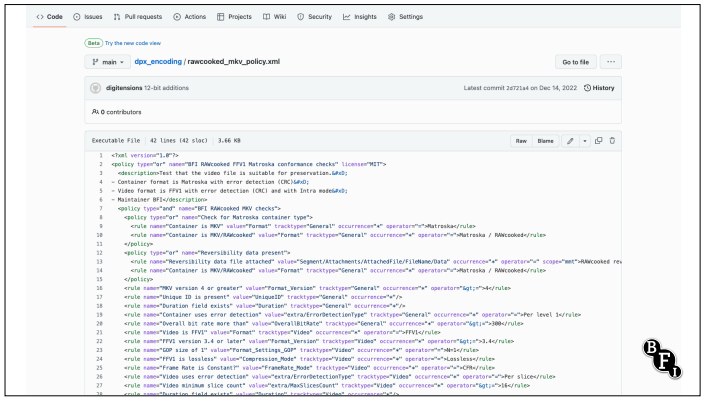
This is an example log for a standard DPX transcode. The first outputs you see are the RAWcooked analysis outputs. When it gets to ‘track 1’ FFmpeg is beginning to run. Here you can see the file information and metadata attachments you’re adding into the MKV file. The next steps are the FFmpeg transcoding logs. You then have a repeated RAWcooked analysis output which culminates in the statement ‘Reversibility was checked, no issue detected.’



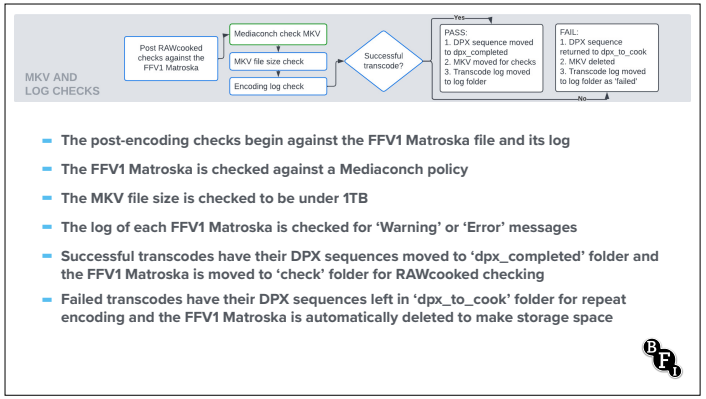
Just to illustrate, we don't always get perfect transcodes, sometimes some errors crop up, and they carry the prefix 'Error:' or 'Warning'. Errors will generally stop a transcode proceeding, but some warnings let you know of an issue perhaps with DPX metadata, but may not stop encoding or reversibility of your sequence.



Following transcoding, there are a series of automated checks that occur now to ensure a file has successfully transcoded. The Matroska file is checked against an FFV1 Matroska Mediaconch policy.



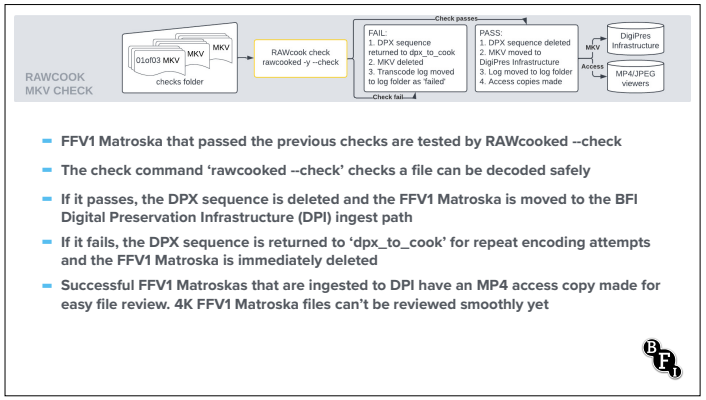
Again you can view this policy at our open source repository, where it shows all our necessary checks against the MKV file. It checks that the RAWcooked reversibility data is attached, checks that the file is lossless, has all the slice CRCs and error detection present, and that the duration field is present. If it's missing it's an indication that a file can be truncated during creation.



Next the MKV file size is checked to ensure it's under 1TB for our DPI ingest. Then those logs are assessed for any error or warning messages. If a specific warning is found to indicate that the transcode failed because of large padding data, then a log output is made to control which RAWcooked encoding pass is made next.

Successful transcoded MKV files are moved to the 'check' folder for the next stage RAWcooked checks. The DPX sequence folder is moved to a 'dpx_completed' folder to stop it being lined up for repeated encoding. The log is moved to a 'log' storage folder and we keep this file.

If an FFV1 failed the Mediaconch policy or Errors indicate a failed encoding attempt then the DPX sequence is left in place in 'dpx_to_cook' for a repeated attempt, the FFV1 Matroska is automatically deleted, and the transcode log is saved in 'log' storage, but prepended 'Failed'.



Finally we have the deletion checks. This stage was added recently to run a RAWcooked '- -check' command test against the file, as we previously kept both the DPX and the MKV files on storage until the MKV had been ingested to our tape library. Now we're handling so much 4K material we quickly fill storage, so we added a RAWcooked check to allow for earlier deletion of the DPX sequence.

This checks that the FFV1 Matroska can be correctly decoded, ensuring that a decoded file matches its original content by checking the checksum and file information stored in RAWcooked's reversibility data, stored in the MKV.

If a file passes this test then the DPX sequence is deleted and the MKV file is moved to our Digital Preservation Infrastructure ingest folder.

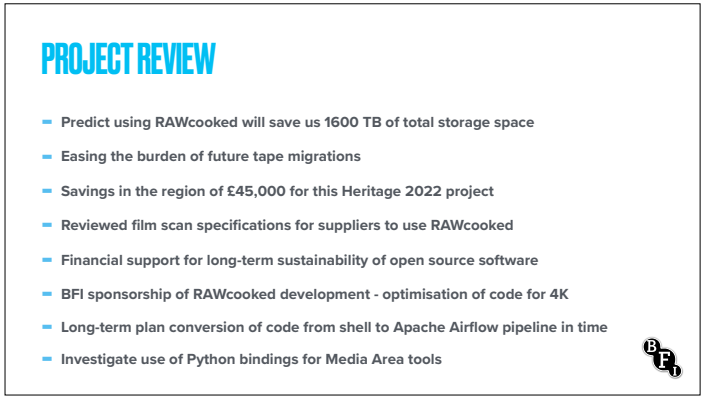
If a file doesn't pass this 'check' then the DPX sequence is returned to the 'dpx_to_cook' folder for repeated encoding, and the FFV1 Matroska is deleted.

Once ingested the FFV1 Matroska has an MP4 access copy made for easier viewing of the compressed content by our technical teams. 4K MKV files can't be played easily by any of our workstations at this time, so an MP4 transcode provides a quick an easy method for reviewing previously unseen DPX data in storage.

So that's a swift overview. I thought you'd like to see some of the RAWcooked FFV1 Matroska files we've ingested to our Digital Preservation Infrastructure. They include films from the Socialism on Film from the Educational and Television Films collection.

Many of these items have colourful optical audio tracks making them beautiful to look at and technically fascinating.

Before RAWcooked encoding we would TAR wrap all DPX sequences, so the ability to review our RAW DPX in this way is one if the best features provided by using RAWcooked.



So to review how successful this workflow has been?

- We predict that using RAWcooked for this Heritage 2022 project will save us a total of 1600 TB of storage space.
- By reducing the content's storage footprint by more than half we ease the burden of future tape migrations.
- Our projected savings will be in the region of £45,000 for this project alone.
- We recently undertook a review of our film scan specifications, using these findings to better inform our suppliers of safe parameters for successful RAWcooked encoding. This also resulted in the creation of a DPX Mediaconch policy which is available to view in MediaArea's public policies web page, and the creation of a sub-licensing feature in RAWcooked allowing us to share our license with our DPX suppliers allowing them the option to create and supply FFV1 Matroska files directly to us, rather than DPX sequences.
- These open source tools are not expensive to implement, but open source shouldn't be thought of as free. Users of these tools need to work together to establish financial support for their long-term sustainability.
- To that end the BFI has sponsored several features and is currently investing in developments to help optimise RAWcooked for our 4K workflows.

In the future:

I'd like to see redevelopment of this workflow so it is managed within an Apache Airflow pipeline. This will allow for greater optimisation as transcoding processes can be shared across multiple 'node' servers. This will also allow realtime monitoring of operating tasks from a web user interface. At present fault finding can be a series of log chases across all the different server shares, and it can take some time!

Finally I'd like to investigate the use of MediaArea's python bindings provided for MediaInfo and MediaConch software.

RAWcooked, FFV1 and Matroska offer unique and exciting solutions for long-term audiovisual preservation, providing much needed format stability and financial savings. I'd like to thank Jérôme Martinez for his patience and support during the development of these scripts, and thank the wider community whose blogs, technical guides and cheat sheets continue to aid us in the creation and maintenance of our workflows. Thank you.
