Analysis of Loss Modes in Preservation Systems

Deliverable D2.2

DAVID identifier: DAVID-D2.2-Analysis-of-Loss- Modes-in-Preservation- Systems.docx

Deliverable number: D2.2

Author(s) and company: Martin Hall-May (ITInnov), Banafshe Arbab-Zavar (ITInnov), Jörg Houpert (CTI), Christian Tiensch (CTI), Hannes Fassold (JRS), Vegard Engen (ITInnov)

Abstract: This is a report on the way in which loss and damage to digital AV content occurs for different content types, AV data carriers and preservation systems.

Three different loss modes have been identified, and each has been analysed in terms of existing solutions and long-term effects. This report also includes an in-depth treatment of format compatibility (interoperability issues), format resilience to carrier degradation and format resilience to corruption.

Internal reviewers: Jean-Hugues Chenot (INA)

Work package / task: WP2 / Task 2.2

Document status: Final

Confidentiality: Public

DOI: 10.7800/304DAVID22
Acknowledgement: The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 600827.

Disclaimer: This document reflects only the author’s views and the European Union is not liable for any use that may be made of the information contained therein.

This document contains material, which is the copyright of certain DAVID consortium parties, and may not be reproduced or copied without permission. All DAVID consortium parties have agreed to full publication of this document. The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the DAVID consortium as a whole, nor a certain party of the DAVID consortium warrant that the information contained in this document is capable of use, nor that use of the information is free from risk, and does not accept any liability for loss or damage suffered by any person using this information.
# Table of Contents

1 Table of Contents ............................................................................................................................ iii

2 List of Figures ................................................................................................................................... v

3 List of Tables .................................................................................................................................... v

4 Executive Summary ......................................................................................................................... 6

5 Introduction ....................................................................................................................................... 7

5.1 Purpose of this Document .......................................................................................................... 7

5.2 Scope of this Document.............................................................................................................. 7

5.3 Status of this Document.............................................................................................................. 7

5.4 Related Documents .................................................................................................................... 7

6 Analysis of loss modes during preservation ................................................................................ 8

6.1 Loss Modes ................................................................................................................................ 8

6.1.1 Faulty Encoder ...................................................................................................................... 8

6.1.2 Physical damage to the file .................................................................................................... 9

6.1.3 Incompatible Encoder-Decoder pair ...................................................................................... 9

6.1.4 Inadequate Encoder .............................................................................................................. 13

6.1.5 Encoder is recording extra data ........................................................................................... 14

6.2 Existing Solutions ..................................................................................................................... 14

6.2.1 Faulty encoder ..................................................................................................................... 15

6.2.2 Physically damaged files ..................................................................................................... 16

6.2.3 Dealing with loss from an unknown source ........................................................................... 16

6.3 Selecting an appropriately effective solution ............................................................................ 18

7 Format compatibility analysis .................................................................................................... 29

7.1 General observations on MXF standard conformity and interoperability issues Error! Bookmark not defined.

7.2 Format compatibility analysis research Error! Bookmark not defined.

7.3 Format compatibility analysis during the IRT MXF PlugFest Error! Bookmark not defined.

7.4 Case Study on a collection of defective ORF MXF DF10 files Error! Bookmark not defined.

7.5 Lessons Learned from MXF preservation Error! Bookmark not defined.

8 Format resilience to carrier degradation ..................................................................................... 29

8.1 Video tape recorders .................................................................................................................. 29

8.2 Error correction and error concealment ................................................................................... 29

8.3 Digital BETACAM (DigiBeta) format ..................................................................................... 30

8.4 Analysis of test material originating from digital tape formats ................................................. 31

9 Format resilience to corruption .................................................................................................... 33

9.1 Automatic detection of corruption ............................................................................................. 33

9.1.1 Detection of structural and colour damage ........................................................................... 33
2 List of Figures

Figure 1. Faulty encoder loss mode ........................................................................................................... 9
Figure 2. Physical damage loss mode ........................................................................................................ 9
Figure 3. Incompatibility loss mode .......................................................................................................... 10
Figure 4. FFmpeg first frame output timeline .......................................................................................... 13
Figure 5. Inadequate encoder loss mode .................................................................................................. 14
Figure 6. Unauthorised metadata loss mode ............................................................................................. 14
Figure 7. Detecting and replacing a faulty encoder first approach ............................................................ 15
Figure 8. Detecting and replacing a faulty encoder second approach ...................................................... 15
Figure 9. Fixing physically damaged files first approach ........................................................................... 16
Figure 10. Fixing physically damaged files second approach .................................................................. 16
Figure 11. Flow chart for loss mitigation during ingest or migration ........................................................ 20
Figure 12. Flow chart for loss mitigation during access .......................................................................... 21
Figure 13: High level view of the mapping of metadata within the MXF container .................................. 25
Figure 14: Sample image, affected by luminance and chrominance dropouts. ....................................... 31
Figure 15. Automatic damage detection ................................................................................................... 34
Figure 16. Edge detection steps at each frame .......................................................................................... 34

3 List of Tables

Table 1. Recovery options and their impact ............................................................................................. 17
Table 2. Structural corruption categories ................................................................................................ 35
Table 3. Colour corruption categories ..................................................................................................... 35
Table 4. Structural and colour corruption category examples .................................................................. 35
Table 5. Frame damage categories and their weight ............................................................................... 36
Table 6. Video content damage categories ............................................................................................... 37
Table 7. Comparison between Experiments 2.2 and 3.2 ....................................................................... 43
4 Executive Summary

This deliverable reports on the work conducted in the DAVID project on analysing loss modes in preservation systems. This work includes identifying and analysing loss modes and existing solutions to the problems identified, as well as a more in-depth treatment of some specific issues: format compatibility issues, format resilience to carrier degradation, and format resilience to corruption.

In the analysis of loss modes, the key focus is on how loss of, or damage to, digital Audio-Visual (AV) content occurs. This work builds on DAVID Deliverable D2.1 ‘Data Damage and its Consequences on Usability’ [Chenot, 2013], looking more closely into the causes of damage, identifying the different modes of loss and methods for mitigation. Three loss modes have been identified:

1. Problematic encoder: e.g., faulty encoder, inadequate encoder or the encoder is recording additional data.
2. Physical damage to files: e.g., damaged carrier, bit rot and block read errors.
3. Inadequate encoder-decoder pair: e.g., ambiguous/inconsistent/mixed aspect ratio or ambiguous/inconsistent colorimetric spaces.

Each of the loss modes is analysed in terms of existing solutions and long-term effects. This deliverable shows that there are often different approaches available to remedy a damaged file or workflow, but that the choice of solution may have long term risks and consequences that should be investigated thoroughly before a mitigation strategy is put into place.

One of the areas identified as important for an in-depth treatment is format compatibility. Specifically to identify the underlying cause of interoperability problems with the media by exchanging between different products. Interoperability problems can occur at different levels; both the container/wrapper level as well as the bit stream level. Since this is a very wide topic, the focus in this deliverable is on MXF as this is currently the most widely used standard for professional TV broadcasting (and in broadcasting archives). A case study on a collection of defective ORF MXF D-10 files is considered, and the lessons learned from this investigation is shared in this report.

Digital dropouts are a major type of damage that is also addressed in detail within this report. Dropouts may occur when storing digital AV content on digital video tape carriers or when transferring this content to file based environments using Video Tape Recorders (VTR). Magnetic tape recording is imperfect and suffers from noise and dropouts. A dropout is defined as a short loss of a recorded signal due to faulty head-to-tape contact or tape imperfections.

In order to avoid large areas of an image to be missing, which can happen due to the loss of several consecutive samples, the data is broken up into separate coded blocks, recorded in a non-sequential order. At playback, this non-sequential order can be reassembled in its correct temporal order by using a controlled code. This will assist in error correction and error concealment. Since the DigiBeta format has been accepted as the standard for the broadcasting and production industry for several years already, it was chosen for an in-depth investigation that is reported on here.

A further detailed analysis of the ‘physical damage to files’ loss mode is also provided in this document. The physical damage in the form of silent data corruption is easily quantifiable, and the damage process can be simulated and the severity of damage evaluated. Silent data corruption includes random change of the bit values on the disk (bit rot). Another type of silent data corruption is sector- or page-sized regions of corrupted data which was observed in a study by CERN [Panzer-Steindel, 2007] to occur as frequently as the single bit errors. We analyse in this document the effects and the damage caused to the video content by such types of silent corruption of data. For this, the corruption is simulated manually by changing the value of random single bits (bit rot) or multiple bits within a sector (sector corruption). Experimental results are presented for three video files, one MP4 and two MXF files. The resulting video files are analysed for visual damage and a statistical analysis of the visual effects of damage is discussed.
5 Introduction

5.1 Purpose of this Document

This document is a report of the findings from the work carried out in Task 2.2 Understanding loss modes of the DAVID project, on the way in which loss of and damage to digital Audio-Visual (AV) content occur for different content types, AV data carriers and preservation systems. This work addresses Objective 1 of the DAVID project – Understanding how damage occurs and what impact it has on digital AV content. Objective 1 is to:

- Create a deeper understanding of how loss and damage occur for digital AV content and which degree of integrity is required for keeping resources usable in different usage contexts.
- Analyse damage and loss modes for specific encoding formats (e.g. impact on data corruption on compressed video), specific content types (e.g. wrappers, audio, video, metadata, subtitles), and specific content carriers (e.g. digital video tape).
- Base analysis on both reported studies in the literature, and on damage that has occurred in archives through a collection of real damaged samples from content producers and consumers, inside and beyond the consortium.
- Take a holistic systems approach to loss by recognising that AV content is stored using multiple copies, in different formats, on different technologies, in different locations and managed using different people and processes.
- Understand what loss modes can be tolerated in different contexts, e.g. reuse, regulatory compliance, online access.

5.2 Scope of this Document

This document provides an analysis of the way in which loss of and damage to digital AV content (including structural metadata) occur for different content types, AV data carriers and preservation systems. The loss modes that threaten AV content are defined, and existing mitigation strategies are analysed in terms of their long term effectiveness. This is discussed in Section 6. Section 7 addresses format compatibility issues, with a real-world case study on a collection of defective ORF MXF D-10 files. Format resilience to carrier degradation, with a focus on digital dropouts, is discussed in Section 8. Based on the physical damage loss mode discussed in Section 6, a more detailed investigation with experimental results is provided in Section 9 (format resilience to corruption). Section 10 concludes this document.

5.3 Status of this Document

The status of this document is final.

5.4 Related Documents

Before reading this document it is recommended to be familiar with the following document:

- D2.1: Data Damage and its Consequence on Usability [Chenot, 2013].
6 Analysis of loss modes during preservation

An earlier DAVID deliverable, D2.1 [Chenot, 2013], has looked into data damage in digital Audio-Visual (AV) archives and its consequences on usability. D2.1 provided a first assessment of the impact of loss on AV properties of the content and the frequency of occurrence of these problems. Here we will look more closely into the cause of these damages, identify the different loss modes, and investigate the methods which are currently in use to mitigate these situations. We will discuss the effectiveness of these mitigation strategies. It will be shown that often there are different approaches available to remedy a damaged file or workflow, but that the choice of a solution may result in long term risks and consequences, which should be investigated thoroughly before a mitigation strategy is put into place.

6.1 Loss Modes

Content loss can happen at the first instance of ingestion to the archive. Then errors may be introduced through transcoding or migration. The data already in the archive can also be the damaged (silent damage). The damage is often detected at the point of access or through routine fixity checks. Here 'access' is used as an inclusive term for any task which involves a software codec to decode the file. With this definition, access can be: playback, transcoding for file transfer, and migration to a new system. Equally, in the following, 'ingest' is not just the initial ingesting of the file into the archive, but is any point at which an encoder software is used to encode data with regards to a standard. For example, at migration we have both 'access' for the old archive system and 'ingest' for the new archive system.

In this section, we will identify different categories of damage and keep track of the faulty or erroneous entities. When designing mitigation strategies, it is desirable to target these erroneous entities. Although the previous statement may sound obvious, as we shall discuss, it may be possible to remedy, for example, the loss caused by a faulty encoder using a compatible (but equally faulty) decoder. Also, please note the following considering the discussions and diagrams in this report:

- All discussions relate to a single format.
- Encoder x and Decoder x share the same core (codec) and are compatible.
- File x is encoded by encoder x and is readable by decoder x.
- File x is constructed using encoder x but is somehow altered or damaged afterwards.
- Faulty or otherwise problematic files or components are displayed in red.

6.1.1 Faulty Encoder

A faulty encoder is an encoder that is not fully compliant with its corresponding standard: all the necessary metadata information and essence data are encoded in the file, albeit in an erroneous form. This problem may go unnoticed while the decoder makes the same erroneous assumptions as the encoder. However, if the decoder is replaced or upgraded, the files may be decoded incorrectly. In this scenario, the files that are kept in the archive are considered faulty. Figure 1 shows a diagram illustrating this loss mode. As mentioned before, the erroneous entities are shown in red. Here the encoder is faulty and is shown in red. Furthermore, the faulty encoder produces faulty files which are also shown in red.
6.1.2 Physical damage to the file

Physical damage to the file may occur as a result of: (i) random change of the bit values on the disk (bit rot); (ii) damaged carrier; and (iii) block read errors. This damage is often silent and is only discovered when the file is accessed for playback, migration or transcoding. Routine fixity checks may help with detecting this damage prior to access. Different issues may arise within this loss mode. The most severe of which is the total loss of the content: when the decoder is unable to decode the file. Other problems may affect the picture, sound, duration of the video or size of the frames. These effects are investigated in detail in Section 9 on page 33. Figure 2 demonstrates this loss mode.

6.1.3 Incompatible Encoder-Decoder pair

The most problematic of the issues is perhaps the incompatibility between the encoder-decoder pairs, which is often a result of the ambiguities within the standard (see Figure 3, below). Where the standard has left room for different interpretations of metadata definition or data layout, different implementations
will arise. These, often incompatible, interpretations may stay hidden from the compliance checkers. Indeed they are standard compliant. Here, the erroneous entity is not the encoder or the decoder, but this problem can be traced back to the standard itself.

Amongst other reasons, ambiguities and inconsistencies may develop within a standard by: a) updating parts of the standard (for example in MXF-JPEG2000, the JPEG2000 standard may change independently of the wrapper, which is MXF); b) having inter-dependent metadata; and c) double definitions or redundant recordings of the metadata. D2.1 [Chenot, 2013] notes:

*Interoperability problems are quite often caused by implementation or parameterisation errors:*

- Non-consistent Presentation Time Stamps (PTS)
- Wrong/inconsistent/mixed aspect ratio
- Wrong/inconsistent colorimetric spaces
- Wrong/inconsistent field/frame wrapping, field dominance
- Valid, but unusual resolution (e.g. 704x576)

![Figure 3. Incompatibility loss mode](image)

These problems, which can arise at any stage when a software or part of a tool is changed within the workflow, cannot be easily detected since the file itself is not considered faulty. To keep track of these issues, sometimes a matrix of interoperability is maintained to help with selecting a compatible decoder for each file or file bundles with a specific encoding. INA’s current solution is to use a code within the filename that indicates which decoder to use. The file is sent to the appropriate decoder automatically as part of the workflow.

Here, an example of such incompatibility is investigated in detail and possible causes have been identified. ‘Inconsistent parameters’ have been mentioned as one of the causes of incompatibility. Inconsistency can occur when the parameter is defined in more than one location in the file. Such inconsistency may be more probable if the parameter has been defined in different standards. Such is the case for MXF-JPEG2000 files which are investigated here in depth for metadata concerning colour space.
MXF-JPEG2000 colour space case study

File Format: MXF-JPEG2000 (INA)
Encoder: OpenCube MXFTk Advanced (2.5.0.20130114)
Compatible Decoder: FFmpeg(20120513)
Incompatible Decoder: FFmpeg(20140313)
Description: Files were previously played back and transcoded correctly using FFmpeg(20120513). However, upgrading to FFmpeg(20140313) results in errors both with playback and transcoding. In this, it appears that the colour space is interpreted incorrectly.

Correct: First frame transcoded using FFmpeg(20120513)
Incorrect: First frame transcoded using FFmpeg(20140313)

The colour space related metadata is stored in multiple locations in MXF-JPEG2000:

1. MXF’s ‘Picture Essence Descriptors’.
2. JPEG2000 codestream, ‘Image and tile size (SIZ)’ marker segment
3. MXF header may also keep a copy of some of the JPEG2000 codestream metadata, including some of the SIZ parameters in ‘MXF JPEG 2000 Sub-Descriptor’.

Colour space is defined differently in each of these locations. Although it is not recorded explicitly in any of these locations, the colour space can be inferred from the information stored in each of these independently. Two different standards are involved here: (JPEG2000: ISO/IEC FCD15444-1) and (MXF: SMPTE 377M). These standards evolved separately, and so the definitions and specifications may change independently. What was once a file with no conflicting metadata information may become an inconsistent file with respect to its metadata due to the change of standard definitions. The colour space definitions are as follows:

MXF’s ‘Picture Essence Descriptors’: the use of one of the two picture essence descriptors have different implications regarding the colour space. Prior to digital cinema initiatives, ‘CDCI Essence Descriptor’ and ‘RGBA Essence Descriptor’ were used for YCrCb and RGBA video file essence, respectively. For Digital Cinema, the main colour space is XYZ. This colour space was later set to be defined in the MXF header using the pixel layout property in the RGBA Essence Descriptor.

This file is using:
MXF CDFI Essence Descriptor
- Component Depth = 10
- Horizontal subsampling = 2
- Vertical subsampling = 1
Inferred colour space: YUV, 4:2:2, 10-bit

JPEG2000 codestream, SIZ marker segment: various parameters within this marker segment give hints toward the colour space in use. These are:
• **Rsiz**: defines the profile of the JPEG 2000 codestream being used. In this, while profiles 3 and 4 define Digital Cinema profiles are used with XYZ colour space, there does not seem to be a colour space restriction for the other profiles.

• **XRsz[i]** and **YRsz[i]**: define the subsampling scheme for the components of the image. For example for the above video example, XRsz[1,...,3] = [1,2,2] and YRsz[1,...,3]=[1,1,1] denote that the first component has a full horizontal and vertical resolution while the two other components have a ½ horizontal resolution but still a full vertical resolution. Although this subsampling in itself does not give information about the nature of these components, we know that such subsampling is common with YUV colour space and is referred to as 4:2:2 and is commonly used for Digital Betacam. A similar inference scheme is used within FFmpeg to infer the colour space from these parameters.

This file is using:

JPEG2000 codestream, SIZ marker segment:
- Rsiz = 0
- Ssiz, XRsz[i] and YRsz[i] = 09 01 01 09 02 01 09 02 01

Inferred colour space: 4:2:2 -> YUV, 9-bit

**MXF JPEG 2000 Sub-Descriptor**: MXF can keep a copy of the values of Rsiz, XRsz[i] and YRsz[i] and, therefore, makes an independent inference as to the colour space being used. The sub-descriptor values for these parameters are considered to be copies from the JPEG2000 codestream. Thus, if there is any discrepancy between the two, those in the codestream take precedence.

This file is using:

MXF JPEG 2000 Sub-Descriptor
- Rsiz = 0
- Ssiz, XRsz[i] and YRsz[i] = 09 01 01 09 02 01 09 02 01

Inferred colour space: 4:2:2 -> YUV, 9-bit

Figure 4, below, demonstrates the timeline of the changes in the FFmpeg interpretations of this video file. Note that the error in the colour space appears for the first time between 18/04/2013 and 24/04/2013. This time frame corresponds to the addition of ‘JPEG 2000 decoder for DCinema’ to FFmpeg.

---

1 (ISO/IEC FCD15444-1 :) Precision (depth) in bits and sign of the \( i^{th} \) component. The precision is the precision of the component before the RCT (Reversible component transformation) or ICT (Irreversible component transformation) is performed. It is not necessarily the precision of the component plane coded in the file. The ICT or RCT can change the precision. This parameter signals the component precision that is in the codestream.

2 Note that the problem seems to have been resolved in versions since 08/2014.

3 The FFmpeg git commit entry: http://git.videolan.org/?p=ffmpeg.git;a=commit;h=c81a70638116eaf4251075475e5cbb600a33c5ec
Although it is still unclear after this analysis what has caused this specific error, it is perhaps more clear as to how the metadata can get out of sync. It would generally be advisable to:

- Define and encode the main metadata information explicitly.
- If possible, avoid double definitions of the metadata, which may become inconsistent or conflicting in different versions.
- Define the priority of usage clearly and explicitly, if there are double or redundant encodings of the metadata.

6.1.4 Inadequate Encoder

This case mainly occurs when some of the metadata is not stored because it is viewed as common practice or is implied (see Figure 5, below). As long as these common practices stay in place, the files that are in essence incomplete may go unnoticed. However, common practices may change. Also standards may change to accommodate different profiles that define how the file should be dealt with. As part of migration, a transcoding tool might strip the metadata during the process of format conversion.
6.1.5 Encoder is recording extra data

An encoder may make unauthorised use of reserved areas of the file to store proprietary data. Such files are essentially faulty but this fault will go unnoticed until the standard is changed to make use of these reserved areas (see Figure 6, below).

6.2 Existing Solutions

Clearly, there are existing solutions to remedy the effects of loss in AV archives. These solutions are reviewed here in attempt to detect the gaps where there is no mitigation strategy or where better, faster or less costly approaches may be possible. In the following, we presume that the source of the problem can be identified and the recovery strategy is targeted at the erroneous parts or entities. In Section 6.2.3 we shall discuss the risks and impact of choosing a recovery option without knowing the cause of the problem.
6.2.1 Faulty encoder

If there is a faulty encoder in the workflow it would be best to replace it by another encoder if possible. Figure 7 and Figure 8, below, show two approaches for detecting a faulty encoder. This is through observing the file or files that have been encoded using this encoder. The resulting file can be played back and the output can be inspected manually (Figure 7) or, alternatively, a compliance checker tool can check the file (Figure 8).

However, it is possible that the encoder cannot be replaced due to logistical reasons. In that case, the other option would be to try to fix the faulty file. Here, we may need access to the original input essence to extract the correct metadata independent of the faulty encoder. This metadata may be stored in a ‘Repair file’ and be used to fix the file. Keeping a repair file, which is basically a trusted independent source of meta-information, is not a common practice. The third option, which is not a desirable choice for long-term preservation as it does not scale, is to support a faulty decoder that can decode the faulty files correctly.
6.2.2 Physically damaged files

Figure 9, below, demonstrates the common method for dealing with physical damage. In this, the fixity checks are used to detect any changes in the bits of the file. If the fixity check fails, the replica is copied through and used as the main copy of the file. Clearly the storage requirements are increased by 100% for each copy that is retained. More granular fixity (e.g. frame-based checksums) allows identification of the position of the error. Replicas may only be kept for the 'important' parts of the file (see Figure 10, below). Partial replicas are more efficient at reducing risk, as they allow greater data safety in the same storage space.

![Figure 9. Fixing physically damaged files first approach](image)

![Figure 10. Fixing physically damaged files second approach](image)

6.2.3 Dealing with loss from an unknown source

If a problem is discovered during decoding of a file and if the problem persists with the backup copy, it might not be apparent which of the reasons listed in Section 6.1 is the cause for it. At this point it is not possible to change the encoder or it may not be possible to have access to the original files. If there are no other mitigation strategies put in place, there are two possible general classes of recovery approaches:
• Make the file (or files) compatible with the decoder of choice. This might involve a bespoke tool to fix the bit stream or container.
• Make the decoder compatible with the file. This might involve changing the decoder configuration (e.g., recompiling from source with different options, or simply invoking with different parameters), altering the decoder implementation (e.g., changing/patching the source code), or switching to an entirely new decoder (version or brand).

Looking back at the loss modes of Section 6.1, we can re-categorise them as:

1. Loss mode where the file is at fault;
2. Loss mode where the decoder is at fault.

Most of the loss modes fall within the category of faulty files. On the other hand, incompatibility issues can be attributed to the decoder. There is another loss mode where the decoder itself is faulty. This is perhaps the easiest loss mode to deal with since changing a decoder would have the fewest repercussions on the rest of the workflow. An easy method to detect which one of these high level categories the problem belongs to is to test the file against a set of decoders. If the majority of the decoders are capable of processing the file correctly then the fault can be attributed to the specific decoder used in the workflow. However, if the majority of the decoders cannot process the file correctly, the fault is most likely with the file.

Table 1 shows the potential impact of choosing a recovery solution given the cause of the problem. If the decision-maker does not know what the cause of the problem is (file or decoder), then they might make the wrong decision (red squares). However, if further investigation has revealed the cause of the problem, then the decision-maker can make the right choice.

<table>
<thead>
<tr>
<th>Problem is with the file</th>
<th>Make file compatible with decoder</th>
<th>Make decoder compatible with the file</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fix, rewrap, or re-encode; ensure standards-compliance</td>
<td>Change decoder configuration, alter decoder implementation, or switch decoder</td>
</tr>
<tr>
<td>Resulting file works well with decoder of choice and has a good chance of working with other (current and future) decoders.</td>
<td>Resulting decoder processes the problem file correctly, but it doesn’t necessarily work for other files. This introduces complexity, as different decoders or decoder implementations / configurations must be maintained for different file classes.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The problem is with the decoder</th>
<th>Make file compatible with decoder</th>
<th>Make decoder compatible with the file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resulting file works well with decoder of choice, but may have been ‘over-fitted’, with the result that it might not work well with other decoders.</td>
<td>Resulting decoder processes the problem file correctly. Regression testing must be used to check that the new decoder (or implementation / configuration) also works for other file formats.</td>
<td></td>
</tr>
<tr>
<td>Problem detected during decoding on ingest or migration (for archiving): File entering the archive is likely to cause problems if the decoder used for subsequent migration or access is different or is upgraded/changed. (High impact)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem detected during decoding on access (for playout): File going to playout is likely to be used only in the short-term and so over-fitting to the playout decoder might be OK. (Low impact)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.3 Selecting an appropriately effective solution

Figure 11 and Figure 12, below, show a decision process that covers the loss modes introduced above in Section 6.1 using existing solutions. It aims to show how a decision-maker could determine the nature of the problem and select the most appropriate and effective strategy.

The potential problems affecting ingest and migration are covered in Figure 11. On ingest or migration, a file is created (e.g., transcoded or normalised) or is ingested unchanged by the archive. Before the file is stored in the archive, the file format should be identified and the file should be checked for compliance with the appropriate standard. If this check fails, the problem might stem from the encoder used to create the file, which does not implement the standard correctly. Ideally, the encoder should be fixed (or a request made that the vendor fix the encoder); however, in some cases, it may not be possible to change the encoder. In this case, the organisation should consider switching to another (tested) encoder. Again, if it is not possible to switch encoder (e.g., owing to vendor lock-in or the high cost of exiting a contract or retraining), then the files could be altered using post-transcode fix-up to be compliant with the standard.

If, on the other hand, the file proves to be compliant with the standard, then decoding should be tested using tools representative of those that will be used in accessing the file contents (e.g., tools for playout or for transcoding to an access format). If this is successful, then the file can safely be stored, with the caveat that there is a small risk of incompatibility with other (untested) decoders, including versions of decoders yet to be released but that might become commonplace during the file generation’s lifetime. However, if the decoding reveals problems with playback, then other decoders should be used to determine the scope of the problem. If other decoders also reveal problems, then there could be a problem with the file that was not detected by the standards compliance checker; a newer or alternative compliance checker should be used. If the majority of other decoders successfully decode the file, then the problem would seem to be isolated to the primary decoder used. Similarly to the decision process regarding the encoder, it should be determined whether the primary decoder can be fixed, or whether it is possible to switch to one of the other good decoders. If neither of these options is available, the last resort is to alter the files such that they are compatible with the primary decoder. In the short term, this is a practical solution, but it means that otherwise standards-compliant files are being damaged and stored in the archive. This solution significantly increases the risks to long-term preservation of the file.

The potential problems affecting access are covered in Figure 12. In normal circumstances, decoding the file for playout or transcoding to an access format should be successful, provided that nothing has changed (the file or the decoder) since it was originally archived. However, corruption of the file may have occurred and/or the decoder may have been replaced or upgraded since that time. If decoding fails, the first thing to check is the integrity of the file (using fixity checksums). If this indicates that the file has changed, then the file should be replaced from a replica, if one exists, or repaired if not.

If this basic integrity check passes, then the file should be checked using recent standards compliance tools. Modern tools, possible involving more rigorous compliance checking, might reveal problems that were not discerned at the point of original storage. If a problem with respect to the standard is revealed at this stage and can be unambiguously identified, then it might be fixed using a bespoke tool. However, if the nature of the problem is not easily identified, or if it involves access to metadata that has been lost, then an independent, authoritative source of metadata for comparison could be used to develop a repair solution. Typically, such a ‘repair pack’ does not exist and is identified here as a gap in the preservation processes. In Figure 12, the gap is circled in red. In the case where a repair pack does not exist, the last resort is to process the file using commodity tools, e.g. to rewrap or re-encode it, in the hope that this results in a file that can be decoded without issue. This solution raises the risk of content quality loss, as there could be generation loss.

If the file proves to be standards compliant, then we must go through the process of testing alternative decoders to ascertain the scope of the problem. As above, if decoding using other tools does not suffer from problems for the most part, then we can either fix or replace the principal decoder used for access, or alter the files to play out (a risky strategy). In the event that decoding is problematic for many (or all) decoders, the repair strategy is not obvious, as the file appears to be compliant with the relevant standards and yet not with any implementations of these. In reality, the chance of this is very small.

The decision charts (Figure 11 and Figure 12) prioritise solutions that result in standards-compliant files, as these have greater likelihood of remaining accessible in the long term. Solutions that fix the problem
only for the immediate context (e.g. current software versions) are short-sighted and are not compatible with long-term preservation needs.
Ingest (or Migration)

1. File is created (encoded)
   - Check file against relevant standards (e.g., using MXF Analyzer)
     - Is standards-compliant?
       - yes
         - Test decoding with primary decoder (e.g., OpenCube)
           - Is decoding good?
             - yes
               - OK
             - no
               - N.B.: Risk of incompatibility with other (e.g., future) decoders
                 - wait for access...
           - no
             - Fix decoder / request fix
               - Can fix decoder?
                 - yes
                   - Fix decoder / request fix
                 - no
                   - Switch decoder
               - Can switch decoder?
                 - yes
                   - Switch decoder
                 - no
                   - Patch files to play out. N.B.: Risk of introducing damage to already standards-compliant files
     - no
       - Fix encoder / request fix
         - Can fix encoder?
           - yes
             - Switch to alternative encoder
           - no
             - Other encoders available?
               - yes
                 - Switch to alternative encoder
               - no
                 - Patch files to be compliant with standard

Figure 11. Flow chart for loss mitigation during ingest or migration
Figure 12. Flow chart for loss mitigation during access
7 Format compatibility analysis

During the preparation of Deliverable D2.1 [Chenot, 2013], an evaluation was performed on the size and current trends in evolution of size and format within collections held by the DAVID partners. Also, investigations were held into the scale of the impact of failure modes on usability and a means was developed to map failure modes relating to different content formats. As a result of this work, format compatibility analysis was identified as strongly in need of further investigation. At the same time, video test material that exhibited integrity problems was collected. These samples were made available for research and development for DAVID Work Packages 3 and 4. ORF detected a defective collection of MXF files, which originated from an earlier mass migration project of SD IMX video tapes. In total, about eight thousand hours of video material were affected to a degree that no further usage was possible.

Media files typically consist of the audio and video essence itself. For transportation and compact packaging, this digital essence is encoded into a bit-stream. This bit-stream is wrapped into a container format, which is then exchanged in file-based workflows. The codecs used and wrappers typically differ between the several stages in the media life-cycle, from production to post-production, delivery, contribution and archiving. For the format compatibility research, we focused on file formats that are typically used in TV broadcast delivery and archiving.

In short, the task of format compatibility analysis is to identify the underlying cause of interoperability problems with the media when exchanged between different products.

Interoperability problems can occur on the MXF container level as well as on the bit-stream level. Damage that occurs only within the essence layer typically results in technically correct files with audible or visual artefacts within the content. Metadata in the bit stream describing the format are replicated in the wrapper to ease the usability of the media file.

As format compatibility analysis of media files is such a wide area, we narrowed down its practical analysis research to the wrapper format that has the widest professional use in TV broadcast and broadcast archives; the Material eXchange Format (MXF). Vendor specific formats such as QuickTime from Apple Inc., or more consumer related formats like the open source Matroska (MKV) or OGG wrapper maintained by Xiph.Org Foundation were not researched as part of the format compatibility analysis. On the bit-stream level, the predominant broadcast format is MPEG4. Currently, mainly MPEG-2 and increasingly MPEG-4 AVC (known also H.264) with intra-frame-only encoding, commercially called AVC-Intra, are used. Broadcast preview formats (proxies) were not considered in this investigation, as proxy formats are encoded on demand and do not need to be preserved for long-term usage.

MXF is a vendor-neutral standardised exchange format for professional Audio-Visual (AV) moving image files. MXF was developed from 1999 on within the Pro-MPEG group. The aim was the development of a platform independent standard for the exchange of finalised video productions. In 2001 Pro-MPEG and the AAF-Association decided to join forces in this undertaking. The whole specification process took five years. The first release was published in 2004 by the Society of Motion Picture and Television Engineers (SMPTE). The underlying MXF data model consists of a section from the Advanced Authoring Format (AAF) data model, which is a professional file interchange format designed for the video post-production and authoring environment. Unlike other international standards bodies, SMPTE does not have a certification program up to now, nor are public reference implementations part of the usual standardisation work.

---

At the time of writing, practical MXF encoder/decoder implementations still face a huge variety of standard conformance and interoperability issues [IRT, 2013]. To understand the causes of the problems with MXF interoperability, one may consider aspects such as:

- complexity of standards
- standard comprehensibility and unambiguousness
- incompleteness of standards definition
- incompleteness of vendor implementations
- version conflicts with revisions of single standards documents within the suite of standards
- metadata variability
- structural variability
- diversity of adaption

7.1 General observations on MXF standard conformity and interoperability issues

In a perfect world, standard conformity and interoperability would be fully in sync (as a major aim of standardisation is interoperability). But even after more than 10 years of standardisation work on MXF, these two attributes still need to be validated separately.

While standardisation bodies seem to try their best to harmonise, the big players in the media business - as well as the “early birds” - shaped today’s reality of video archiving with commercial products, sometimes even before the standards documents were finalised. Some, mainly smaller companies, tried to build their MXF functionality based on open source implementations, such as the different FFmpeg branches. The result is a high diversity of implementations without any centralised point for quality control and standards compliance checking, and without any commercial pressure for standard compliant implementations. As the SMPTE standards documents have to be purchased in the SMPTE shop, some of the free MXF implementations are partly based on open available illegal copies of outdated standards documents, or by re-engineering of other existing MXF software.

7.2 Format compatibility analysis research

Over the period of one year, the CubeFTec team has analysed a few thousand MXF files; mainly broadcast flavours, like D-10, DVCPRO, XDCAM HD422, AVC-Intra, but also JPEG2000 and uncompressed bit-streams. For this, an automated analysing procedure was set up, with a reference MXF Analyser Professional [MXF Analyser Professional] from the Institut für Rundfunktechnik (IRT). In addition, a CubeFTec internal video bit-stream analyser was developed (CubeFTec Essence Stream Analyser). Of the MXF files that have been analysed, it is worth mentioning that they have been produced by a diverse list of products:

Adobe (Premiere Pro), Avid Technology (AirSpeed5000, Media Composer, Transfer Manager, MPI), Belden / Grass Valley (K2 9, Edius), Dalet / AmberFin (iCR), David Systems (MCL Player), EVS / OpenCube (OC Server, OC MxfTk, OpenCubeHD, XFReader), Fairlight (Pyxir, Xynergy), Oracle / Front Porch Digital (Samma), Hamburg Pro Media (MXF4Mac GC, Export FCP, MXF4Mac ImEx Suite), Harmonic (MIP-7600, Omneon Spectrum Server, Rhzet ProMedia Carbon), IBM (Arema), Imagine Communications / Harris Broadcast (Nexio Server), Marquise (MIST, Medway), Merging Technologies (Vcube, Wrap), MirriAd (ZoneSense), MOG Solutions (mxfSpeedrail), Quintel (SQEdit), Rohde & Schwarz / DVS (Venice Server), Sony (PMW-1000, Vegas, e-VTR, PDW-U1/2, PDW-HD1500), Telestream (FlipFactory, Vantage Transcode Pro, Pipeline Quad, Pipeline HD Dual),

Typical playback problems of video files with format compatibility issues are:

- Video Server refuses to playback an MXF file
- Server playback freezes with a still image while playing
- Server playback stutters with interrupts and jumps while playing the video
- Video player crashes while playing a file and needs a restart, or reboot.

Similar problems can occur in transcoder products or during archive ingestion. Other typical issues in reusing MXF files are:
• Picture playback with wrong aspect ratio, or even jumping aspect ratios
• Timecode jumps in player
• Incorrect timecode display
• Multi-channel audio with wrong format
• Audio playback is muted
• Playback with wrong audio channel alignment
• An encoded audio stream (e.g., Dolby E) is reproduced as PCM audio
• No playback of subtitles/captions

The most relevant documents for verifying or analysing MXF for standard compliance are:

• SMPTE Standards [SMPTE, 2014] on Format and Encoding
• SMPTE Standards on Operational Pattern
• SMPTE Standards on Essence Container Infrastructure
• SMPTE Standards on Descriptive Metadata
• SMPTE Recommended Practises (RP)
• SMPTE Engineering Guidelines (Tutorial Documents)
• SMPTE Product-specific Registered Disclosure Documents (RDD)
• EBU Technical Statements [EBU, 2014] on the use of MXF
• EBU Technical Recommendations on special MXF issues
• ISO/MPEG Standards [MPEG, 2014]
• ITU-R Standards
• AMWA Application Specifications [AWMA, 2014]

There are about 100 relevant documents to consider, which are highly inter-referenced, and also have a number of references to external standards bodies like W3C, or IETF.

It seems that also some implementers had problems in understanding the controlled vocabulary within the MXF standards documents. An example is the correct usage of the term ‘should’ and ‘should not’ in the SMPTE documents. This controlled vocabulary phrase indicates that, among several possibilities:

• One is recommended as particularly suitable, without mentioning or excluding others.
• OR indicates that a certain course of action is preferred but not necessarily required.
• OR indicates that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The MXF data model is a subset of the Advanced Authoring Format (AAF) data model, an object-based file format that wraps video, audio, and other bit-streams (“essences”) in a metadata-aware structure. Essences are wrapped in containers; implementations of the MXF Format Generic Container for specific encodings. Tracks represent the passage of time. Timeline Tracks (standard tracks) describe segments to form continuous sequences. Tracks are synchronised by storing them into a package. See Figure 13, below, for a high level diagram that shows the mapping of metadata within the MXF container.
7.3 Format compatibility analysis during the IRT MXF PlugFest

The Institut für Rundfunktechnik (IRT) in Munich is the research and development institute of the public broadcasters of Germany, Austria and Switzerland. As a user-driven organisation, IRT has been actively participating in the MXF specification, standardisation and implementation activities of the European Broadcasting Union (EBU), the Pro-MPEG Forum and SMPTE. Since 2006, IRT organises an annual event known as the MXF PlugFest. The last PlugFest (at the time of writing) in November 2013 was joined by 18 vendors with their products for MXF file exchange testing. In these two days of the PlugFest, two main test procedures were performed:

1. Interoperability tests between the participating products under test
2. Verifying the standard conformance based on the IRT MXF Analyser product

IRT has compiled a list of errors that most commonly appear in those MXF files that could not be successfully interchanged. The most common MXF wrapper errors are:

1. **Non-KLV** (Key-Length-Value) data. The existence of non-KLV data is not allowed according to the MXF standards. All data (e.g. video, audio, metadata, etc.) inside an MXF file must be KLV coded.
2. **Invalid BER** (Basic Encoding Rules) lengths. The MXF standard defines how many bytes can be used to indicate the Length of the following Value (e.g. video essence).
3. **Incorrect Data Definitions for Timecode tracks**. If the Data Definition is incorrect, a decoder can likely not process the Timecode.
4. **Invalid SMPTE** Unique Material Identifiers (UMID). Correctly encoded UMIDs ensure their uniqueness, this cannot be ensured by invalid UMIDs,
5. **Invalid SMPTE Universal Labels**. All SMPTE Universal Labels start with the same “sync word”, however some encoders create files that don’t obey this rule.
6. **Partition Pack does not start at a KLV grid line** of the previous Partition. All partitions that use the KLV Alignment Grid (KAG) shall ensure that the end of that partition is padded to a grid line (realised e.g. by fill items) defined by that partition.

7.4 **Case Study on a collection of defective ORF MXF D-10 files**

In 2003, ORF started a huge preservation project to transfer about 150,000 hours of analogue video content from Panasonic MII, 1-inch-B format and U-Matic video tapes to digital Sony IMX-tapes. In 2010, ORF decided that, instead of recording to Sony IMX-tapes, they would start generating MXF D-10 archive files. At that time, ORF did not have a Media Asset Management system for long-term preservation, so a Tape Library (based on LTO-4 tapes) was used as an interim stopgap for the backup. This way, more than 18,000 hours of analogue video material were transferred to MXF D-10 files. By the end of 2012, ORF had established its ORF Essence File-Storage. From that time on, media files were preserved directly to the new ORF File-Storage.

While starting the transfer of the MXF D-10 files from the interim back-up tape library to the new ORF file storage system, these files were often rejected or caused errors in the preview-production-process or other transfer-tasks. At first it seemed that only a few hundred files stored on the LTO-4 tape library had severe errors, then more than a thousand were identified, until ultimately the ORF engineers realised with great disappointment that as much as 50% of the files would not transfer properly to the new long-term storage system.

The ORF engineers managed to nail down the reason for the problem. It was not, as initially considered, a problem with the retrieval of the data tape from the tape robots; the problems were generated by an early version of the MXF capture software used to record the analogue video tapes into the MXF D-10 file format.

The errors were not detected earlier, as the files seemed to behave fine on the original recording system and, with the available measurement tools that ORF engineers used at that time, the errors did not present themselves. The defective collection was afterwards somehow isolated within the back-up data tape system, which restricted their reuse and also reduced the chance to detect the issue earlier. What were the possibilities to receive standard compliant and interoperable D-10 files from these 9,000 hours of ORF archive video content?

Re-running the digitisation process was a possibility, but it was not clear whether the old tapes would still play back. ORF had severe problems with a lot of the 1-inch B material. This approach was also not attractive, owing to costs and time constraints. Transcoding was also not an option, because of degeneration in visual signal quality (adding another generation of lossy coding to the video material with an already sub-optimal technical quality). As no repair tools were available on the market, the task was transferred into the DAVID research project. Cube-Tec engineers volunteered to run an analysis of the D-10 files based on a small batch of defective files.

The first step in this analysis phase was to collect a group of samples and detect where the issue originated from. The second step was to analyse whether there was a possibility for an automatic repair that would maintain the integrity of the AV essence. This was the most important aspect of the solution. Transcoding the media stream would not have been an appropriate tactic as this could have introduced degeneration in AV signal quality. These files had to be repaired without touching the media essence.

Based on the analysis of the batch of sample MXF files provided by ORF, the origin and symptoms of the errors were detected. The errors comprised combinations of issues in both the MXF container and random data corruption in the MPEG bit-streams. Fortunately, the encoded picture essence data was not damaged in most cases.

Given the relatively small number of tested samples, it was not clear whether all variants of the defective 9,000 hours of video material had been encountered during the initial analysis stage. Therefore, a **Repair as a Service** approach was sought. However, due to the geographic distance between ORF and Cube-Tec (Vienna and Bremen), transferring the data was not feasible because of unacceptable network speeds. Sending data tapes via post/courier service was not accepted by ORF either, so a hardware server was installed in the ORF data centre, near to the Tape Library. Thus, the tools were local, which resolved the issues regarding security and network speeds, although Cube-Tec engineers had remote access to control the server.
A complete migration workflow was set up, from selection of the right MXF files on the backup Tape Library by the ORF team, to copy them in chunks to the MXF Repair Server input watch folder and output all corrected MXF files including MD5 checksums for fixity supervision to the input watch folder for the final ORF File-Storage System. This way the Cube-Tec engineers had full remote control of the MXF repair process to deal with newly occurring file defects. When the Cube-Tec team detected additional systematic variability of the file damage structure, the repair software was updated. The analysis and repair data throughput was up to 2 Terabytes per day.

This approach has worked very well. In this process, more than 95% of the defective MXF files were corrected to be fully standard compliant. With the remote control interface, the process supervision via the Internet was a complete alternative to direct local access to the data. Phase 2 of this work started in October 2014 to address the remaining 5% of MXF failures that could not be repaired previously.

We believe that this version of ‘insourcing of technical expert services’ can also be an attractive blue print for other archives. With this approach, ORF has found a way to repair 9,000 hours of defective MXF files in a budget friendly way and with relatively low effort. Cube-Tec has gained deep knowledge in MXF file analysis and repair techniques with a large diverse media collection, which lead to new commercial offers. The MXF repair RTD results have already been incorporated by Cube-Tec into a commercial product named MXF Legalizer. Also the remote TV media QC and Repair concept is now available as ‘automated file analysis and repair as a remote service’, within the MXF Legalizer Online Service, which was released for non-commercial public beta testing by the time of writing. Leading on from this, a ‘Human expert file analysis & repair as a Remote Service’ is in evaluation for testing possible future commercial offers.

As this report is only focused on the analysis aspects of the ORF D-10 project, the full details of the repair methodology and the final achievements are not included. Instead, they are described in more detail in D4.2 ‘Final Tools for Damage Detection, Repair and Quality Improvement’. However, note that this deliverable is not available for public access.

### 7.5 Lessons Learned from MXF preservation

Based on the above format compatibility analysis, we have extracted the following lessons learned to help minimising risks with interoperability and future playability of MXF files:

- **MXF playback devices** - especially consumer software video players are usually highly tolerant when playing back damaged MXF files – such tools should not be used to validate playability of MXF files. The fact that an MXF player plays a file today as expected is no more than an indication that this file is not heavily corrupted.

- When, in a few years’ time, the harmonisation on how to use the MXF standards will have reached a higher level of conformance, one should not expect that all these early day file variations of the past 10 years will still be supported within future equipment.

- If different MXF devices behave differently with an MXF file, this can be an indication that the file is not standard compliant and has internal attributes (format describing metadata) that are not consistent and are interpreted differently by the MXF devices used.

- Internal file elements like multiple timecodes, captions or subtitles, etc., are more prevalent in broadcast collections and if they are preserved within the MXF file they can be very useful for future forensic analysis of the file coding history.

- There are some recommendations to vendors to build MXF tools that are highly tolerant with MXF errors on the input side, but provide a strict MXF flavour at the output. This strategy should help to cure MXF collections while being migrated and reused. The SMPTE MXF standards documents describe how a compliant encoder/decoder has to behave. They usually cannot standardise all aspects on how a decoder/transcoder must handle MXF non-standard-compliant files. In this aspect vendors have to get creative; they usually implement highly different strategies to handle compliance issues within the MXF file. The Format Compatibility RTD team could not find evidence that this strategy is better than the straightforward strategy to refuse non-standard compliant files, so that they can be corrected as early as possible in the media life cycle.
• One should keep an eye on emerging MXF Application Specifications like AS-07 developed for long-term preservation which support rich metadata.

• The cost of media preservation depends on the cost of future format migration processes and this depends on whether the media collection allows homogenous technical access and the media is standard compliant. The percentage of “difficult media” within a collection defines widely the future access cost to the file collection. Without easy automatic accessing possibilities to a collection, reuse will be severely restricted and the practical value of the collection will decrease.
8 Format resilience to carrier degradation

As a basis for long-term preservation, Audio-Visual (AV) content held on video tapes is transferred to a file based storage system. Digital dropouts are a major type of damage which may occur when storing digital AV content on digital video tape carriers or when transferring this content to file based environments using Video Tape Recorders (VTR).

8.1 Video tape recorders

For analogue or digital AV tape recording a moving magnetic tape in contact with a rotating recording/replay head is required [Ward, 2000]. Due to the large amount of data needed to represent a frame, recording all in one single linear tape track is not possible. The solution is dividing frames into fields and segmentation of the field. Within most analogue VTRs, several recording heads are mounted in a rapidly-spinning drum that is pressed against the moving tape. The heads move across the tape in a transverse or diagonal path, recording the video signal in consecutive parallel "tracks" [Wikipedia, 2014], which effectively increases the bandwidth that can be recorded.

As digital audio and digital video are just different kinds of data, Digital Video Tape Recorders (DVTs) record both using the same heads and tape tracks [Watkinson, 1995]. The video data is increased to make room for the audio data, which is stored using the same data rate as the video [Watkinson, 1995]. Furthermore, increasing the data rate allows the insertion of edit gaps, addresses and redundancy into the data stream [Watkinson, 2013].

Magnetic tape recording is imperfect and suffers from noise and dropouts. A dropout is defined to be the short loss of a recorded signal due to faulty head-to-tape contact or tape imperfections [Ward, 2000]. Several reasons could lead to problems and imperfect reproduction of video and audio on replay:

- Dust gets between the tape and the head, or tape imperfections or missing magnetic coating cause the head-to-tape contact to be disturbed.
- Tape and head rotation may not be continuously stable and some minute variations in speed may occur. This will affect the accuracy of the recording on replay.
- Errors in data compression: data compression is required, since digital signal processing allows a smaller signal-to-noise ratio when recording, while requiring a much higher bandwidth [Ward, 2000].
- The frequent appearance of dropouts on playback is an indication that the tape or recorder is contaminated with debris and/or that the tape binder is deteriorating.

To avoid a number of consecutive samples causing loss of a large area of an image, prior to recording, an interleaving process divides the video and audio signals in packets of data which are shuffled. The data is thus broken up into separate coded blocks, recorded in a non-sequential order. This is done by using a controlled code that can be used to re-generate correct spatial-temporal order when decoding. On replay, the blocks are decoded and arranged in their correct sequence by a subsequent de-interleaving process. This effectively separates the potentially missing or corrupted data, ensuring that consecutive values are less likely to be lost [Ward, 2000]. This assists in error correction, as discussed in the following section.

8.2 Error correction and error concealment

DVTs use error correction and concealment to deal with missing or corrupted data. Commonly, in digital recording compressed video and audio are recorded in packets of data. The digital video data is encoded using an error correction system that is able to replace some lost data completely. Values of missing pixels, e.g., due to noise or poor signal quality are restored by using Error Correction Codings (ECCs) to compute exactly the value of the wrong bit(s). In the error concealment step, the data is shuffled. The data, which is divided into separated coded blocks, is rearranged and recorded out of sequence. This is done so that a large area of missing or corrupted data is spread out over the picture.

This error correction technique is only useful for a very few consecutive samples [Ward, 2000]. However, some errors are just too great to be corrected, such as when a head clogs up. In this case, error concealment is used.
Error concealment cannot compute exact pixel values, but instead estimates probable values from correct pixels nearby. In most cases the estimate is remarkably good and goes unnoticed by viewers of the content, but concealment is potentially visible.

The error strategy of DVTRs is based on several ideas:

1) Data distribution: Data is distributed over two or more heads and tracks working in parallel. The distribution is done by feeding a particular head with alternate pixels in rows and columns. If one head clogs, the original picture is still left, but sampled less often. Interpolation can be used to estimate values of the missing pixels.

2) Shuffling: dropout usually causes an area of severe damage in the picture, surrounded by intact areas. If the dropout is correctable, this is no problem, but if area is large and concealment has to be used, the concealed area will be obvious to the viewer. The solution is a shuffle process that changes the order of the pixels or blocks before recording and then re-aligns them on playback. When shuffling is used, an un-correctable dropout results in pixels or blocks requiring concealment that are randomly spread across the image. Concealment is then less visible.

3) Product codes: Product codes are an error correction strategy where pixels are formed into a rectangular array prior to recording. Check-words are then added both to the rows and the columns of the array prior to recording. In replay there is a two stage process. The rows are checked for error, and small errors are corrected. Large errors, however, are used to produce flags which go into the array. When columns are read from the array, the second stage of correction is informed by the flags where the errors are.

Additional correction support is to record duplicate, additional data which is redundant if there are no errors but can be used to compare or replace any missing or incorrect data [Ward, 2000].

8.3 Digital BETACAM (DigiBeta) format

For nearly two decades, the mainstay for recording in production environments have been DigiBeta, IMX, DV, HDCAM, and some more digital video formats. DigiBeta (Digital BETACAM) in particular, has been accepted as a standard for the broadcasting and production industry in Standard Definition (SD). It is a high-quality archiving format with first mass migration projects (from tape to file) starting in the near future [Fassold, 2013].

In the DigiBeta format, one-field video data and 4-channel audio signals of each format are recorded on six helical tracks (program tracks) [Matrix AV, 2014]. It records a 2 to 1 DCT-compressed digital component video signal at 10-bit YUV 4:2:2 sampling in NTSC (720×486) or PAL (720×576) resolutions at a bit rate of 90 Mbit/s plus four channels of uncompressed 48 kHz / 20 bit PCM-encoded digital audio. A fifth analogue audio track is available for cueing, and a linear time code track is also used on the tape [Wikipedia, 2014].

Head-to-tape speed is three times higher than for BetaSP, and luminance and chrominance are recorded at different locations. Head clogging, dirt on the heads, or other impacts (as discussed above) can lead to luminance or chrominance dropouts, respectively on one field. A sample image, affected by luminance and chrominance dropouts is shown in Figure 14.
Figure 14: Sample image, affected by luminance and chrominance dropouts.

Two types of dropouts originating from DigiBeta tape format are illustrated: first, luminance dropouts of size 8x8 pixels and second, chrominance dropouts of size 16x8 pixels. It seems that there is a geometrical pattern for both types, which can be explained by the ‘block shuffling’ procedure of the DigiBeta format. By analysing all available test material we have produced a classification of digital dropouts, by using dropout properties, such as appearance colour, size, contrast to background, relations to neighbourhood /other dropouts, etc.

8.4 Analysis of test material originating from digital tape formats

For the digital tape dropout detection problem JRS did a comprehensive analysis of available test data obtained mainly from ORF and INA.

From the defect region, a first grouping was done into 8x8 pixel defects, 16x8 pixel defects (and defects with different size). Within these groups, we further sub-divided into 3 types, dependent on the missing pixels within the blocks: We found a chessboard structure, stripes (in both cases only one field was affected) or full block damages (2 fields affected).

Generally, there seems to be no temporal coherence, i.e., it is very unlikely that digital dropouts appear on the same positions in consecutive frames. A spatial neighbourhood is possible, vertically and horizontally, but it is rather rare. It seems that there is a geometrical pattern at least for the DigiBeta format.

Finally, we defined the following 6 individual classes of digital dropouts.

Class 1: 8 x 8 chessboard luminance dropouts

These are visible only in the Y channel, have a chessboard structure and only one field is affected.

Class 2: 16 x 8 chessboard chrominance dropouts
These are visible only in the U and/or V channel, have a chessboard structure as well and only one field is affected.

**Class 3: 8 x 8 line dropouts (stripes), one field affected**
Mainly the Y channel is affected.

**Class 4: 16 x 8 luminance/chrominance line dropouts (stripes), one field affected**
It is not clear yet if only U and/or V channel is affected or Y as well.

**Class 5: 8 x 8 luminance/chrominance block dropout class**
Both fields are affected.

**Class 6: 16 x 8 luminance/chrominance block dropout class, both fields affected**

After a comprehensive discussions among the DAVID partners (JRS, ORF, CTI, INA), it was concluded that the presence of a geometric relationship (due to shuffling) between the affected blocks of a frame would be a valuable, discriminative feature for DigiBeta defects. With this finding we can also explain the spatial relational pattern that we have already recognised for class 1 and 2.

The bigger size of the second class was explained due to the sub-sampling of the chrominance signal. Thus, class 1 dropouts mainly occur due to damages in the luminance (Y) channel and class 2 dropouts because of defected chrominance (U and V) information. The DAVID partners suspected that luminance and chrominance are recorded in different locations. The different colours and patterns of class 2 dropouts can be explained by the 4:2:2 sampling pattern.
9 Format resilience to corruption

The physical damage loss mode described in Section 6.1.2 is investigated in more detail here. The physical damage in the form of silent data corruption can be quantified, the damage process can be simulated and the severity of damage evaluated. A statistical analysis of the visual effects of damage is presented here.

First, it is worth repeating what we mean by silent data corruption. This includes random change of the bit values on the disk (bit rot). Another type of silent data corruption is sector- or page-sized regions of corrupted data which was observed by a CERN study [Panzer-Steindel, 2007] to occur as frequently as the single bit errors\(^5\). The objective of this study is to analyse the effects and the damage caused by these types of silent corruption of data to the video content. For this, the silent corruption of data is simulated manually by changing the value of random single bits (bit rot) or multiple bits within a sector (sector corruption). The resulting video files are then analysed for visual damage.

9.1 Automatic detection of corruption

The aim here is to detect the visible damage to the video file as the result of bit rot or sector corruption. For this, each frame is compared to the corresponding reference frame of the un-corrupted video. The difference between these frames is then classified as a structural and/or colour damage. Finally, the video file is classified into one of 17 categories of damage based on the reoccurrence and severity of the damage in its frames. We will refer to the categories below in respective sections.

9.1.1 Detection of structural and colour damage

The structural damage to the content of a frame is evaluated by comparing the position of edges in the potentially damaged frame to the uncorrupted frame. The number of matching values between the edge pixels (when both or neither show an edge) are then summed up and normalised by the total number of pixels in the image. This is referred to as the structural score, \(S_{st}\):

\[
S_{st} = \frac{\sum_{i}^{N} \sum_{j}^{M} \neg(E_d(i,j) \oplus E_r(i,j))}{N \times M}
\]

where \(N\) and \(M\) are the number of rows and columns in the image while \(E_d\) and \(E_r\) denote the Edge maps of the damaged frame and the reference frame respectively. The structural score takes on values in the range \([0,1]\), where 0 denotes no similarity and 1 is the score of a decoded frame with the exact same edges as the reference frame.

A reference value is also computed, which is the structural score of the reference frame with a black frame (no edges). This is referred to here as the ‘Structure base score’. Any frame with a score below this value is considered to have lost all the structural information within the frame. The colour score, \(S_c\), is computed as the mean of the pixel-wise Euclidean distance of the RBG channels for RGB colour space\(^6\):

\[
S_c = 1 - \frac{\sum_{i}^{N} \sum_{j}^{M} d(C_d(i,j), C_r(i,j))}{N \times M}
\]

where \(d(p,q)\) denotes the Euclidean distance between the two points \(p,q\). The colour score is also normalised to the range \([0,1]\), where 0 denotes no similarity between the colours and 1 is for same colours at all pixels.

Figure 15, below, shows a damaged frame and its corresponding reference frame along with their edge maps. The difference map and the structural and colour scores are also shown on the right. A structural

\(^5\) Note: This point is to some extent ambiguous in this paper. They report the frequency of these two errors as percentage of overall error. However it is not clear whether the sector errors which may include multiple single bit changes count once towards the overall error or count depending on the actual single bit changes within them.

\(^6\) Only the hue and saturation channels are used for the HSV colour space.
damage usually causes a colour disparity as well. As such, the ratio of the colour score over the structural score is considered to evaluate the colour damage. Furthermore, for the frames that have severe structural damage the colour damage is regarded as invalid and is not considered at all.

Figure 15. Automatic damage detection

The edge detection is performed in three stages: i) Gaussian smooth filtering on the luminance channel; ii) Sobel edge detection; iii) thresholding to obtain a binary edge map (see Figure 16, below). Note that due to the pixel-wise comparison of the edge points, the computed structural score is sensitive to translation (shifted to a new location) and rotation of the main features within a frame.

Figure 16. Edge detection steps at each frame.

9.1.2 Structural and colour damage categories

Multiple thresholds on the value of structural and colour scores determine the category of damage. In training, a number of frames are manually labelled for the severity of their structural and colour corruptions and are used to determine the thresholds for the categorisation of the damage. For this, a sample video file in MP4 format is corrupted by single bit flips. This video is comprised of 25 frames. Two approaches are taken for selecting the bits to be flipped: i) the most significant bit of all bytes of the header; ii) the most significant bit of a randomly selected byte. Since it is impractical to examine the effects of bit flip corruption on all the bytes of the file, 5% of the total number of bytes are selected at random to be examined. Note that for each experiment, only a single byte is corrupted. It is worth noting that similar patterns of damage emerge in the corrupted files. Also, the main damage appears to be structural, while the cases of incorrect colours with high structural score are limited. Table 2 and Table
3, below, list the structural and colour damage categories. Examples of different structural and colour damage categories are shown in Table 4, below.

**Table 2. Structural corruption categories**

<table>
<thead>
<tr>
<th>Cat.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No corruption</td>
</tr>
<tr>
<td>2</td>
<td>Small/ Not visible corruption</td>
</tr>
<tr>
<td>3</td>
<td>Noticeable corruption</td>
</tr>
<tr>
<td>4</td>
<td>Poor frame reconstruction</td>
</tr>
<tr>
<td>5</td>
<td>All structure is lost</td>
</tr>
</tbody>
</table>

**Table 3. Colour corruption categories**

<table>
<thead>
<tr>
<th>Cat.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No or very little corruption</td>
</tr>
<tr>
<td>2</td>
<td>Small corruption</td>
</tr>
<tr>
<td>3</td>
<td>Noticeable corruption</td>
</tr>
</tbody>
</table>

**Table 4. Structural and colour corruption category examples**

**Structural corruption categories examples:**

2. Small/ Not visible corruption:

- $S_{st} = 0.9989$
- $S_{c} = 0.9969, \frac{S_{c}}{S_{st}} = 0.998$

- $S_{st} = 0.9989$
- $S_{c} = 0.9980, \frac{S_{c}}{S_{st}} = 0.999$

3. Noticeable corruption:

- $S_{st} = 0.9720$
- $S_{c} = 0.9842, \frac{S_{c}}{S_{st}} > 1$

- $S_{st} = 0.9709$
- $S_{c} = 0.9709, \frac{S_{c}}{S_{st}} = 1$

4. Poor frame reconstruction:

- $S_{st} = 0.8702$
- Structure base score = 0.8626

- $S_{st} = 0.9012$
- Structure base score = 0.8626
5. All structure is lost:

\[ S_{st} = 0.8525 \]
Structure base score = 0.8626

\[ S_{st} = 0.8371 \]
Structure base score = 0.8626

Colour corruption categories examples:

3. Noticeable corruption:

\[ S_{st} = 0.9676 \]
\[ S_c = 0.9238, \frac{S_c}{S_{st}} = 0.955 \]

\[ S_{st} = 0.9947 \]
\[ S_c = 0.9486, \frac{S_c}{S_{st}} = 0.954 \]

9.1.3 Frame damage categories

Each frame is classified into one of the nine categories based on its structural and colour corruption. A weight is assigned to each damage category denoting its severity. In this, any type of small corruption has a weight of ‘1’. The weight increases by one for each more severe corruption category (see Table 5).

<table>
<thead>
<tr>
<th>Cat.</th>
<th>Description</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No error</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Small colour corruption</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Noticeable colour corruption</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Small structural corruption</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Small structural + colour corruption</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>Noticeable structural corruption</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>Noticeable structural + colour corruption</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>Poor frame reconstruction</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>All structure is lost</td>
<td>5</td>
</tr>
</tbody>
</table>

9.1.4 Video damage categories

A video is a collection of frames where each frame might have been affected in a different manner by the corruption of the file. A temporal histogram of frame-wise damage categories across all frames of the video file is then considered. For this, the temporal bins are as follows:

- Not considerable: \( \leq t_1 \)
- Small reoccurrence: \( t_1 < \) and \( < t_2 \)
- Large reoccurrence: \( \geq t_2 \)
- Almost all of frames: \( \geq 90\% \) of all frames
The values of $t_1$ and $t_2$ can be chosen considering the required sensitivity, the video format and the frequency of the I-frames. The damage of the video content is finally classified into 16 corruption categories and one size error category. These 16 categories are based on both the corruption at the frame level and the rate of reoccurrence of each of these frame-level corruption types. It is possible that different frame-level corruptions happen within a damaged video sequence. In such cases, a weight is calculated for each video damage category which is the weight of the corresponding frame-level damage (see Table 5, above) multiplied by the reoccurrence weight, which goes from zero to three for the temporal bins mentioned above. The damage with the highest weight is then reported as the corruption category for the video file. Table 6 shows the video damage categories.

<table>
<thead>
<tr>
<th>Cat.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No error</td>
</tr>
<tr>
<td>2</td>
<td>Small structural or colour corruption in small number of frames</td>
</tr>
<tr>
<td>3</td>
<td>Small structural or colour corruption in large number of frames</td>
</tr>
<tr>
<td>4</td>
<td>Noticeable structural or colour corruption in small number of frames</td>
</tr>
<tr>
<td>5</td>
<td>Small structural or colour corruption in almost all frames</td>
</tr>
<tr>
<td>6</td>
<td>Noticeable structural and colour corruption in small number of frames</td>
</tr>
<tr>
<td>7</td>
<td>Poor frame reconstruction in small number of frames</td>
</tr>
<tr>
<td>8</td>
<td>Noticeable structural or colour corruption in large number of frames</td>
</tr>
<tr>
<td>9</td>
<td>All structure is lost in small number of frames</td>
</tr>
<tr>
<td>10</td>
<td>Noticeable structural or colour corruption in almost all frames</td>
</tr>
<tr>
<td>11</td>
<td>Noticeable structural and colour corruption in large number of frames</td>
</tr>
<tr>
<td>12</td>
<td>Poor frame reconstruction in large number of frames</td>
</tr>
<tr>
<td>13</td>
<td>Noticeable structural and colour corruption in almost all frames</td>
</tr>
<tr>
<td>14</td>
<td>All structure is lost in large number of frames</td>
</tr>
<tr>
<td>15</td>
<td>Poor frame reconstruction in almost all frames</td>
</tr>
<tr>
<td>16</td>
<td>All structure is lost in almost all frames</td>
</tr>
<tr>
<td>17</td>
<td>Size error</td>
</tr>
</tbody>
</table>

9.2 Experimental results

Experiments have been carried out on three video files. One of these files, as discussed before, is in the MP4 format, while the other two are in the MXF format. Two general approaches for corrupting the files were considered:

i) byte-size corruption (single bit flip)

ii) sector-size corruption (multiple bit flips)

For simulating the byte-size corruption, the worst-case scenario with flipping the most significant bit of the selected byte is performed. Random bits are flipped within a sector for the sector-size corruption. It was observed that the corruption of the header, where the metadata of the file is stored, may cause more persistent and severe damage, while the size of the file header is generally much smaller than the data part of the file. Therefore two general approaches have been used to select the to-be-corrupted bytes/sectors. The first step was to corrupt all the bytes/sectors of the header one by one and evaluate the resulting video files for the potential damage. Then, the corruption of the rest of the video data has been considered. In that case, 5% of the total volume of data is corrupted at random at a byte- or sector-wise corruption level.
9.2.1  Experiment 1: MP4

Video file:
Format: MP4  
Sample rate = 25/1  
Frame count: 25  
Width: 640 pixels  
Height: 360 pixels  
Header byte count: 925  
File byte count: 75900

Experiment 1.1:  
Corruption type: bytes-wise corruption  
Corruption location: header (all bytes)  
Test set size: 925 samples  

Run time:
• Corruption time: 27 min  
• Evaluation time: ~ 2 hrs

Results:
Cases of all frames decoded: 681 (73.6%)  
Cases of all frames decoded correctly: 668 (72.2%)  
Cases of some corruption: 13 (1.4%)  
Noticeable structural or colour corruption in small number of frames: 2 (0.2%)  
Noticeable structural or colour corruption in large number of frames: 8 (0.9%)  
Noticeable structural or colour corruption in almost all frames: 2 (0.2%)  
Poor frame reconstruction in almost all frames: 1 (0.1%)  
Cases of truncation: 240 (25.9%)  
Cases of no frames decoded: 153 (16.5%)  
Cases of truncation but correct decoding of frames: 71 (7.7%)  
Cases of some corruption: 16 (1.7%)  
Small structural or colour corruption in small number of frames: 2 (0.2%)  
Noticeable structural or colour corruption in small number of frames: 10 (1.1%)  
Noticeable structural or colour corruption in large number of frames: 2 (0.2%)  
All structure is lost in small number of frames: 1 (0.1%)  
All structure is lost in almost all frames: 1 (0.1%)  
Cases of incorrect frame size: 4 (0.4%)  

Discussion:  
Note that the main effect of the corruption of the header seems to be the inability to decode frames altogether, with 25.9% of samples having un-decoded frames. In comparison, the visual errors in the decoded frames are small, at 3.5% (1.4%+1.7%+0.4%) overall.

Experiment 1.2:  
Corruption type: bytes-wise corruption  
Corruption location: 5% of all the bytes  
Test set size: 3796 samples  

Run time:
• Corruption time: 2 hrs 22 min  
• Evaluation time: ~ 11 hrs

Results:
Cases of all frames decoded: 3781 (99.6%)
  Cases of all frames decoded correctly: 2227 (58.7%)
  Cases of some corruption: 1554 (40.9%)
    Small structural or colour corruption in small number of frames: 490 (12.9%)
    Small structural or colour corruption in large number of frames: 388 (10.2%)
    Noticeable structural or colour corruption in small number of frames: 234 (6.2%)
    Noticeable structural and colour corruption in small number of frames: 39 (1.0%)
    Poor frame reconstruction in small number of frames: 2 (0.1%)
    Noticeable structural or colour corruption in large number of frames: 168 (4.4%)
    Poor frame reconstruction in large number of frames: 186 (4.9%)
    All structure is lost in small number of frames: 6 (0.2%)
    Noticeable structural and colour corruption in large number of frames: 7 (0.2%)
    Poor frame reconstruction in large number of frames: 186 (4.9%)
    All structure is lost in large number of frames: 34 (0.9%)

Cases of truncation: 14 (0.4%)
  Cases of no frames decoded: 5 (0.1%)
  Cases of truncation but correct decoding of frames: 7 (0.2%)
  Cases of some corruption: 2 (0.1%)
    Noticeable structural or colour corruption in small number of frames: 1 (0.0%)
    Noticeable structural or colour corruption in large number of frames: 1 (0.0%)

Discussion:
Visual error appears when the body parts of the video data is corrupted. Here the un-decoded frames are negligible at 0.4%. However, even a single bit corruption in the body part of the data in 17.8% of times will cause a visual damage that is not small (40.9%−(12.9%+10.2%)).

9.2.2 Experiment 2: MXF, lossless

Video file:
    Container format: MXF
    Image format: JPEG2000
    Sample rate = 25/1
    Frame count: 26
    Width: 1920 p
    Height: 1080 p
    Header byte count: 9728
    Compression: Lossless @ ~370Mbps

Experiment 2.1:
    Corruption type: bytes-wise corruption
    Corruption location: header
    Test set size: 9725 samples (header byte count is 9728. 3 samples have been erroneously left out)

  Run time:
    • Corruption and frame extraction time: 39 (second per video sample)
    • Evaluation time: 10 (second per video sample)

  Results:
    Cases of all frames decoded: 8709 (89.5%)

Cases of all frames decoded correctly: 8709 (89.5%)
Cases of truncation or elongation: 1016 (10.4%)
  Cases of no frames decoded: 638 (6.6%)
  Cases of truncation but correct decoding of frames: 1 (0.0%)
  Cases of some corruption: 377 (3.9%)
    Poor frame reconstruction in large number of frames: 322 (3.3%)
    Poor frame reconstruction in almost all frames: 55 (0.6%)

Discussion:
There is no visual corruption. The 3.9% detected corruption is due to the fact that truncation has disturbed the frame correspondence between the reference frames and the decoded frames. The main damage here is due to the un-decoded frames.

Experiment 2.2:
Corruption type: sector-wise corruption
Corruption location: 5% of the file sectors
Test set size: 4923 samples

Run time:
- Corruption time: (not measured)
- Evaluation time: (not measured, similar to above)

Results:
Cases of all frames decoded: 4920 (99.9%)
  Cases of all frames decoded correctly: 1722 (35.0%)
  Cases of some corruption: 3198 (65.0%)
    Small structural or colour corruption in small number of frames: 1943 (39.5%)
    Noticeable structural or colour corruption in small number of frames: 265 (5.4%)
    Noticeable structural and colour corruption in small number of frames: 273 (5.6%)
    Poor frame reconstruction in small number of frames: 677 (13.8%)
    All structure is lost in small number of frames: 40 (0.8%)
Cases of truncation: 3 (0.1%)
  Cases of no frames decoded: 0 (0%)
  Cases of truncation but correct decoding of frames: 0 (0%)
  Cases of some corruption: 3 (0.1%)
    Poor frame reconstruction in large number of frames: 3 (0.1%)

Discussion:
Similarly to experiment 1.2, the corruption of the body part does not result in many un-decoded frames. In terms of the severity of the damage, 25.5% of the corrupted video files have noticeable or more severe damage, while 35% of the video files are decoded correctly and the damage to 39.5% of the files is not visually detectable.

Examples:

Noticeable:
9.2.3  Experiment 3: MXF, lossy

**Video file:**
- Container format: MXF
- Image format: JPEG2000
- Sample rate = 25/1
- Frame count: 26
- Width: 1920 p
- Height: 1080 p
- Header byte count: 9728
- Compression: Lossy @160Mbps

**Experiment 3.1:**
- Corruption type: byte-wise corruption
- Corruption location: header
- Test set size: 9726 samples (header byte count is 9728. 2 samples have been erroneously left out)

**Run time:**
- Corruption and frame extraction time: 39 (second per video sample)
- Evaluation time: 10 (second per video sample)

**Results:**
- Cases of all frames decoded: 8710 (89.5%)
  - Cases of all frames decoded correctly: 8710 (89.5%)
- Cases of truncation or elongation: 1016 (10.4%)
  - Cases of no frames decoded: 638 (6.6%)
  - Cases of truncation but correct decoding of frames: 1 (0.0%)
  - Cases of some corruption: 377 (3.9%)
    - Poor frame reconstruction in large number of frames: 323 (3.3%)
Poor frame reconstruction in almost all frames: 54 (0.6%)

**Discussion:**
Very similar results to experiment 2.1 are obtained here; again no visual corruption is observed and the 3.9% detected corruption is due to the truncation that is disturbing the frame correspondence between the reference frames and the decoded frames. Also the main damage is again due to the un-decoded frames.

---

**Experiment 3.2:**
Corruption type: sector-wise corruption
Corruption location: 5% of the file sectors
Test set size: 2142 samples

**Run time:**
- Corruption time: (not measured)
- Evaluation time: (not measured, similar to above)

**Results:**
Cases of all frames decoded: 2139 (99.9%)
- Cases of all frames decoded correctly: 496 (23.2%)
- Cases of some corruption: 1643 (76.7%)
  - Small structural or colour corruption in small number of frames: 616 (28.8%)
  - Noticeable structural or colour corruption in small number of frames: 471 (22.0%)
  - Noticeable structural and colour corruption in small number of frames: 69 (3.2%)
  - Poor frame reconstruction in small number of frames: 448 (20.9%)
  - All structure is lost in small number of frames: 39 (1.8%)
- Cases of truncation: 3 (0.1%)
  - Cases of no frames decoded: 0 (0%)
  - Cases of truncation but correct decoding of frames: 0 (0%)
  - Cases of some corruption: 3 (0.1%)
    - Poor frame reconstruction in small number of frames: 1 (0.0%)
    - Poor frame reconstruction in large number of frames: 2 (0.1%)

**Discussion:**
A point to point comparison of experiments 2.2 and 3.2 are shown in Table 7, below. These two experiments compare the effects of lossy compression on the robustness to data corruption.

**Examples:**

![Noticeable Corruption Example](image1)

![Noticeable Corruption Example](image2)
Poor:

All lost:

Table 7 shows the comparison between the experiments 2.2 and 3.2 where the effects of random sector-wise corruption of the data are compared for a lossless and a lossy compression of MXF file format. As expected, the lossless compression is more robust to corrupted sectors. It has higher percentage of corrupted video samples decoded correctly. Also when errors occur they are less severe. In this, for the lossless video samples 60.8% (1943/3198) of the 65.0% errors that are detected are small, while only a 37.5% (616/1643) of the 76.7% errors detected for lossy video samples are small corruptions and the rest are more severe forms of damage.

Table 7. Comparison between Experiments 2.2 and 3.2

<table>
<thead>
<tr>
<th></th>
<th>Lossless @ ~370Mbps</th>
<th>Lossy @160Mbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cases of all frames decoded correctly</td>
<td>35.0%</td>
<td>23.2%</td>
</tr>
<tr>
<td>Cases of truncation</td>
<td>0.1%</td>
<td>0.1%</td>
</tr>
<tr>
<td>Cases of corruption:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small structural or colour corruption</td>
<td>39.5%</td>
<td>28.8%</td>
</tr>
<tr>
<td>Noticeable structural and/or colour corr.</td>
<td>10.9%</td>
<td>25.2%</td>
</tr>
<tr>
<td>Poor frame reconstruction</td>
<td>13.8%</td>
<td>20.9%</td>
</tr>
<tr>
<td>All structure is lost</td>
<td>0.8%</td>
<td>1.8%</td>
</tr>
</tbody>
</table>

9.2.4 Summary and conclusions

In this section, a method has been introduced to evaluate the visual damage to video files due to physical corruption associate with bit flips and sector corruption. As shown, a physical corruption to a file does not always translate to a visual damage in the rendered video. Furthermore, the different parts of the file have different sensitivity to physical damage. In this regard, a corruption in the header is more likely to cause a noticeable damage in the rendered video. Also the choice of image format and video container as well as the level of compression impacts the robustness of the resulting video file to physical damage. The method above can be used in a systematic way for robustness evaluations of the files in the planning stages of an archive.
10 Conclusions

This deliverable builds on the work reported in D2.1 [Chenot, 2013], by identifying how loss of/damage to digital Audio-Visual (AV) content occurs. Three loss modes have been identified:

1. **Problematic encoder**: e.g., faulty encoder, inadequate encoder or the encoder is recording additional data.

2. **Physical damage to files**: e.g., damaged carrier, bit rot and block read errors.

3. **Inadequate encoder-decoder pair**: e.g., ambiguous/inconsistent/mixed aspect ratio or ambiguous/inconsistent colorimetric spaces.

For each loss mode, existing solutions have been analysed, and the main findings are:

1. **Problematic encoder**: acceptable existing solutions exist.
   - Detecting the problematic encoder:
     - Using various decoders.
     - Using compliance checkers.
   - Remedy the loss:
     - Replace the encoder.
     - Repair the file.

2. **Physical damage to files**: acceptable existing solutions exist.
   - Detecting the problematic file:
     - Using fixity and partial fixity.
     - Detection on access.
   - Restore the file:
     - Using replicas and partial replicas.

3. **Inadequate encoder-decoder pair**: incomplete solutions.
   - Detect the incompatibility:
     - Detection on access.
   - Remedy the loss:
     - No systematic remedy.

In addition to the analysis of the loss modes, this deliverable has also reported on a format compatibility analysis, and format resilience to both carrier degradation and corruption. For this part of the deliverable, real-life case studies and experimental results have been reported.

Regarding format compatibility issues, several recommendations have been made in this deliverable to help minimise the risks with interoperability and future playability of MXF files. These recommendations have been made on the basis of a case study on a collection of defective ORF MXF DF10 files. In this case study, ORF engineers discovered issues with a large batch of analogue video content (18,000 hours’ worth) that had been converted to MXF D-10 files in 2010. Approximately 50% of the files could not be transferred to a new long-term storage system, which stemmed from problems with an early version of an MXF capture software.

A Repair-as-a-Service approach with Cube-Tec engineers was conducted, with encouraging results. It was important not to touch the media essence as the integrity of this was paramount. Approximately 95% of the errors could be detected and repaired (fully standard compliant), which were sourced to both the container and bit stream levels. As part of this case study, the software from Cube-Tec was already expanded to deal with new issues identified, and further work will address the remaining 5% of the errors.
Regarding digital dropouts, this has been investigated specifically in this deliverable as it has been identified as a major type of damage that can occur when storing digital AV content on digital video tape carriers or when transferring this content to file based environments using Video Tape Recorders (VTR). Based on data provided by ORF and INA, a comprehensive analysis has been conducted, in which we have defined 6 individual classes of digital tape dropouts (including DigiBeta):

- **Class 1**: 8 x 8 chessboard luminance dropouts
- **Class 2**: 16 x 8 chessboard chrominance dropouts
- **Class 3**: 8 x 8 line dropouts (stripes), one field affected
- **Class 4**: 16 x 8 luminance/chrominance line dropouts (stripes), one field affected
- **Class 5**: 8 x 8 luminance/chrominance block dropout class
- **Class 6**: 16 x 8 luminance/chrominance block dropout class, both fields affected

We found that that the presence of a geometric relationship (due to shuffling) between the affected blocks of a frame is a valuable discriminative feature for DigiBeta defects. With this finding we can also explain the spatial relational pattern that we have already recognised for class 1 and 2. The bigger size of the second class was explained due to the sub-sampling of the chrominance signal. Thus, class 1 dropouts mainly occur due to damages in the luminance (Y) channel and class 2 dropouts because of defected chrominance (U and V) information.

Regarding the physical damage loss mode, the format resilience to corruption has also been investigated in greater detail. With the aim of automatic detection of corruption as a result of bit rot or sector corruption, experiments have been carried out on three different video files (one MP4 file and two MXF files; one with lossy and the other with lossless compression). One of the key aspects of these experiments is the comparison of the effects of lossy compression on the robustness to data corruption. As expected, the lossless compression is more robust to corrupted sectors. Firstly, it has a higher percentage of corrupted video samples decoded correctly, but also when errors occur they are less severe. Moreover, as a general observation across all files, corruption in the header is more likely to cause a noticeable damage in the rendered video.

With this deliverable, we have furthered the understanding on how loss and damage occurs with many existing solutions. However, there is a gap for a) availability of an independent, trusted ‘pack’ of metadata, and b) robust workflows (and change to workflows). This gap is, however, addressed in the DAVID project in Task 3.3 and will be reported on in May 2015.
11 References


## Glossary

Terms used within this deliverable, sorted alphabetically.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4:2:0</td>
<td>A colour sampling pattern, where chrominance is sub-sampled horizontally and vertically: only one pixel out of two holds the colour information, only one line out of two. Used by DV in 50Hz environments (Europe).</td>
</tr>
<tr>
<td>4:2:2</td>
<td>A colour sampling pattern, where chrominance is sub-sampled horizontally: only one pixel out of two holds the colour information. Used by most SD and HD recording formats.</td>
</tr>
<tr>
<td>AMWA</td>
<td>Advanced Media Workflow Association.</td>
</tr>
<tr>
<td>Aspect ratio</td>
<td>Ratio of the expected width divided by height of the picture content of a programme. Usually 4:3 in SD, 16:9 in HD...</td>
</tr>
<tr>
<td>AV</td>
<td>Audio-Visual.</td>
</tr>
<tr>
<td>AVC</td>
<td>Advanced Video Coding.</td>
</tr>
<tr>
<td>BER</td>
<td>Basic Encoding Rules.</td>
</tr>
<tr>
<td>Carrier</td>
<td>Physical media, holding video, or files.</td>
</tr>
<tr>
<td>Codec</td>
<td>“Coder-Decoder”; device or computer program capable of encoding or decoding a digital data stream or signal.</td>
</tr>
<tr>
<td>Colour space</td>
<td>System by which the colours in the contents are represented. A colour is represented by three coordinates in a colour space, but if an error is made changing from one colour space to another, or if the wrong colour space is specified, representation of colours will be affected. (a short list of colour spaces’s: RGB, R'G'B', CYMK, CIE XYZ, L&quot;u&quot;v&quot;, L&quot;a&quot;b&quot;, Y'C'B'R, YcCbcCr, Y'P'B'R, Y'UV, Y'IQ...)</td>
</tr>
<tr>
<td>Container</td>
<td>See Wrapper.</td>
</tr>
<tr>
<td>DAVID</td>
<td>Digital AV Media Damage Prevention and Repair.</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete Cosine Transform. Mathematical transformation of data, used in lossy compression.</td>
</tr>
<tr>
<td>DigiBeta</td>
<td>Digital Betacam.</td>
</tr>
<tr>
<td>Drop-out</td>
<td>Loss of Radio Frequency (RF) signal read by one (or more) of the microscopic magnetic playback heads, reading at high speed (several m/s) the recorded magnetic track, resulting in temporary loss of signal.</td>
</tr>
<tr>
<td>DV</td>
<td>The first of the DV digital video tape family, uses intra-frame video compression scheme. Uses 4:1:1 or 4:2:0, 8 bits, at 25Mbps, stored in cassettes, standalone files, of MXF (or others) wrappers.</td>
</tr>
<tr>
<td>DVC PRO</td>
<td>Professional Panasonic version of DV. Uses MP.</td>
</tr>
<tr>
<td>DVTR</td>
<td>Digital Video Tape Recorder.</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBU</td>
<td>European Broadcasting Union.</td>
</tr>
<tr>
<td>ECC</td>
<td>Error Correction Codings</td>
</tr>
<tr>
<td>FFmpeg</td>
<td>A popular free, open source, software commonly used for decoding, encoding, transcoding digital AV content.</td>
</tr>
<tr>
<td>H.264</td>
<td>See MPEG-4 AVC.</td>
</tr>
<tr>
<td>HD</td>
<td>High Definition.</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force.</td>
</tr>
<tr>
<td>IMX</td>
<td>Sony Betacam IMX video format, using MPEG2 4:2:2 intra-frame compression at 30, 40, or 50Mbps. Available as cassettes, but can be exported as MXF files (known as MXF-D10).</td>
</tr>
<tr>
<td>IRT</td>
<td>Institut für Rundfunktechnik.</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardisation.</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology.</td>
</tr>
<tr>
<td>JPEG</td>
<td>Joint Photographic Experts Group, a lossy compression format for digital images.</td>
</tr>
<tr>
<td>JPEG2000</td>
<td>The second version of the JPEG standard, created in 2000.</td>
</tr>
<tr>
<td>KAG</td>
<td>KLV Alignment Grid.</td>
</tr>
<tr>
<td>KLV</td>
<td>Key Length Value.</td>
</tr>
<tr>
<td>Lossless</td>
<td>A class of image compression that does not result in data / quality loss, unlike 'lossy' – see below.</td>
</tr>
<tr>
<td>Lossy</td>
<td>A class of image compression, which results in some lost data and quality from the original version.</td>
</tr>
<tr>
<td>LTO</td>
<td>Linear Tape Open.</td>
</tr>
<tr>
<td>Mbps</td>
<td>Mega-bits per second: one million bit per second.</td>
</tr>
<tr>
<td>Metadata</td>
<td>Data about the data. Information about the contents.</td>
</tr>
<tr>
<td>Matroska</td>
<td>An open source multimedia container/wrapper.</td>
</tr>
<tr>
<td>MD5</td>
<td>Message Digest function 5.</td>
</tr>
<tr>
<td>MKV</td>
<td>The file extension for Matroska (see above).</td>
</tr>
<tr>
<td>MP</td>
<td>Metal Particle. Tapes where the particles holding the magnetic information are made of very finely ground metallic particles. Allow for a higher storage density than oxide.</td>
</tr>
<tr>
<td>MP4</td>
<td>A short name for MPEG-4.</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving Picture Expert Group.</td>
</tr>
<tr>
<td>MPEG-2</td>
<td>DCT-based standard for storing as files or bitstreams audio-visual contents. Can support a wide range of intra or inter-frame compression ratios.</td>
</tr>
<tr>
<td>MPEG-4 AVC</td>
<td>A very common standard video compression format, also known as</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>H.264</td>
<td>which is used in Blu-ray.</td>
</tr>
<tr>
<td>MXF</td>
<td>Material eXchange Format.</td>
</tr>
<tr>
<td>NTSC</td>
<td>National Television System Committee, an analogue television standard, common in many American countries.</td>
</tr>
<tr>
<td>OGG</td>
<td>A free, open, digital AV container.</td>
</tr>
<tr>
<td>OP1a</td>
<td>One of the MXF Operational patterns used to store, in a single file, one video track, with several synchronous audio tracks.</td>
</tr>
<tr>
<td>Pixel</td>
<td>A Picture element. A sample of the picture; may have RGB, YUV, or luminance-only value.</td>
</tr>
<tr>
<td>PTS</td>
<td>Presentation Time Stamp: in a coded programme, specifies the specific time in the timeline where the referred chunk of data/video/audio should be presented.</td>
</tr>
<tr>
<td>QC</td>
<td>Quality Check/Control.</td>
</tr>
<tr>
<td>RAID</td>
<td>Redundant Array of Independent Disks.</td>
</tr>
<tr>
<td>RGB</td>
<td>Red Green Blue – see ‘Colour space’.</td>
</tr>
<tr>
<td>RGBA</td>
<td>Red Green Blue Alpha – see ‘Colour space’.</td>
</tr>
<tr>
<td>SD</td>
<td>Standard Definition.</td>
</tr>
<tr>
<td>Silent damage</td>
<td>Damage to content that can occur over time without the content being accessed or manipulated in any way. This relates to damage such as bit rot. This kind of damage is only discovered when the file is accessed for, e.g., playback, migration or transcoding.</td>
</tr>
<tr>
<td>SIZ</td>
<td>Image and Title Size.</td>
</tr>
<tr>
<td>SMPTE</td>
<td>Society of Motion Picture &amp; Television Engineers.</td>
</tr>
<tr>
<td>UMID</td>
<td>Unique Material Identifiers.</td>
</tr>
<tr>
<td>VTR</td>
<td>Video Tape Recorders.</td>
</tr>
<tr>
<td>WAV</td>
<td>A Wrapper for audio contents.</td>
</tr>
<tr>
<td>Wrapper</td>
<td>A standard describing how different media elements and metadata coexist in a computer file (e.g. MXF, QuickTime, Matroska, WAV, MPEG-4…).</td>
</tr>
<tr>
<td>XDCAM</td>
<td>Sony line of recording media products, based on Blu-Ray, within shell. Uses MXF-OP1a, can store DVCAM, IMX, XDCAM HD, XDCAM EX and XDCAM HD422 can be accessed as generic storage device.</td>
</tr>
</tbody>
</table>
## Partner Acronyms

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTI</td>
<td>Cube-Tec International GmbH, GE</td>
</tr>
<tr>
<td>HSA</td>
<td>HS-ART Digital Service GmbH, AT</td>
</tr>
<tr>
<td>INA</td>
<td>Institut National de l'Audiovisuel, FR</td>
</tr>
<tr>
<td>ITInnov</td>
<td>University of Southampton - IT Innovation Centre, UK</td>
</tr>
<tr>
<td>JRS</td>
<td>JOANNEUM RESEARCH Forschungsgesellschaft mbH, AT</td>
</tr>
<tr>
<td>ORF</td>
<td>Österreichischer Rundfunk, AT</td>
</tr>
</tbody>
</table>