AMWA Specification
AMWA Application Specification
AS-02 MXF Versioning
November 18, 2011 (version 1.0)

Executive summary

New routes to market for entertainment media are resulting in an explosion of new devices and channels that enable consumers to watch their favorite content. For the content owners and distributors, this is a double-edged sword.

- On the one hand, the overall market for content is increasing.
- On the other hand, the number of eyeball-contact minutes for any given distribution channel is decreasing.

This motivates content owners to search for more efficient ways of creating the media required for different distribution channels.

One common way of looking at the world is to consider the *media factory* shown in figure 1.

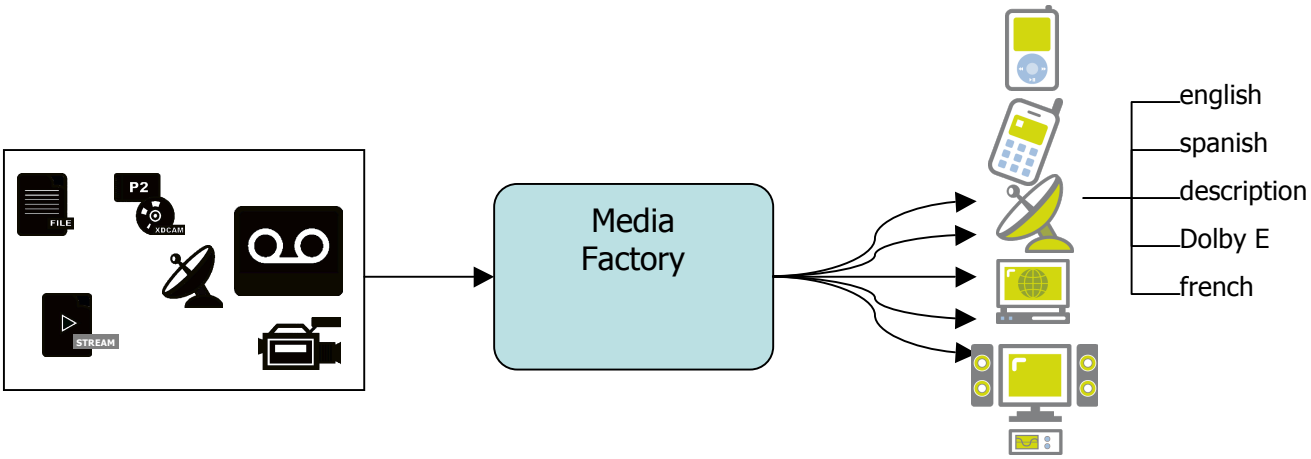

Figure 1: Media factory

The media factory accepts its input in the form of tapes and files. Its role is to deliver the revenue generating deliverables to the downstream services and devices. The media factory may be a business in its own right, or it may be a function within a larger facility. As facilities migrate to file-based working from tape-based working, the media factory concept is appearing in organizations around the world. To make this media factory efficient and measurable is a business decision, ensuring that costs and efficiencies can be calculated.

Today, many media organizations do not keep the kind of Key Performance Indicators (KPIs) that would be traditionally employed if the factory were making mechanical *widgets*. There have been a number of revolutions in the overall design of mechanical factories in recent years that we can use to help our overall design of a media factory. "Lean Manufacturing", "Kanban" scheduling and "Total Quality Management" are all hot topics in factory design. There is no universal right answer to the way in which a factory should be built, but all of these mechanisms are aimed at the one central goal:

- Reduce waste.

In order to achieve this, it is important to know what waste in a media factory actually is. The list can be quite long, but the high-level view includes:

- Moving media that does not need to be moved;
- Keeping media that does not need to be kept;
- Copying media that does not need to be copied;
- Processing media that does not need to be processed;
- Taking too long to process media that could be processed more quickly;
- Deleting data or metadata that needs to be recreated in a later process;
- Re-keying data or metadata that was previously known;
- Making a tape from a file that was previously a tape.

In general, waste in a media factory is the same as waste in any other factory. In achieving the result that the business needs, materials, time and/or resources are used unnecessarily.

The MXF file format was designed to associate metadata and media so that factories could be created that were much more efficient than tape-based factories. It seems, however, that these factories have not magically appeared because of a large number of factors that need to be considered when migrating from a tape-based world to a file-based world.

The MXF file format was developed to provide a wrapper agnostic to the number of audio channels. or the compression format, or the resolution of the video. It was developed to provide a metadata-rich way of performing business processes on an asset whilst leaving the fine detail to the underlying machines that manipulate the pixels.

To allow a step change in waste reduction to be realized, the AS-02 application specification defines an efficient way of using MXF within the media factory. This is also a step change in cost-efficiency of handling media.

A key factor in this cost-efficiency is that AS-02 is an application specification for MXF users to build workflows. Its design is oriented towards low-cost, high-efficiency media factories rather than towards any one vendor's view of the world. The underlying design brings the best from the MXF media world and the best from the IT world to allow standards-based workflows to be created without requiring expensive customization. AS-02 can be considered as a set of constraints on the SMPTE-MXF toolbox. It is not a new MXF standard. It is a way of making the best use of the tools that exist.

MXF was standardized in 2004 by the SMPTE. Since then, a large number of products, services and tools have been created that use MXF. The product types are split into three broad categories:

1. OP1a interleaved devices and tools – largely operating on files as if they were tapes;
2. Atoms / referenced file devices – largely operating on source elements;
3. Generic toolkits, SDK and transcoding tools – gluing the world together.

Neither the OP1a nor the atom way of using MXF is currently optimized for building a good factory.

Factories based on interleaved OP1a make the world look as much like a tape as possible to ease the migration from tape to file. This is a good thing because it introduces basic file-based working and the associated quick-win benefits. These benefits include faster than real time transfers and aggregation of large volumes of content onto single server, realizing genuine savings. These savings allow the concept of a media factory to be created in an otherwise tape-based world.

The OP1a format has one major design principle – interleave the video and audio so that you always have the data for one frame of video close to the data for the synchronized sound. This is great for an acquisition device like a camera or for a playback device - like a single channel playout server - for which AS-03 is a good fit. However, this can lead to serious waste for media factory operations. For example, if you need to create a stereo audio from the surround sound audio track, why should you have to transfer the video at the same time? 90% of the data transferred is not touched. This is waste.

The OP-Atom format (SMPTE 390M-2004), as used in Avid environments and Panasonic P2 cameras, was optimized for storing individual essence components in editing environments. The actual essence storage is almost perfect for a factory environment. The problem comes with the way in which the individual components are synchronized:

- OP-Atom synchronizes the components within each OP-Atom file, allowing for only one version;
- The dCinema community makes use of an XML structure for the same role.

The goal of the AS-02 work was to create a factory format that supports multiple versions and with no re-invention, making use of metadata-only MXF (SMPTE 377-1-2009) to provide component synchronization. With as few constraints as possible, we had to use the tools in MXF as they were written and implemented. One lesson learned in lean manufacturing is that if you get it right, it should feel like the design is just *on the edge*, where on the edge means:

- You could write some more specification, but it wouldn't be widely used;
- If you deleted any of the specification, you would lose many of the benefits.

MXF is only a file format. It is a tool that allows you to build a better media factory. Like all tools, it can be used well and it can be misused.

The AS-02 application specification contains many - but not all - of the rules that you need to build a media factory. It only contains the rules that relate to the file format. Other rules are required to put the system together, such as those to do with communication between AS-02 processes and AS-02 services. An interface specification that creates a common dialog for interoperable services between media factories is a likely topic for a future AMWA specification.

When a media asset is used within a media factory, there are several ways in which it can be used:

- The unique identifier of the asset. Sometimes a filename, a house number or an UMID.
- The standardized metadata of the asset. Sometimes a database record, a MXF header or a QT header.
- The custom metadata of the asset. Often an XML file that may be lost when the asset is exchanged.
- A component of the asset. The video data or the data in one of the audio tracks.
- A simple version of the asset. The English soundtrack Internet version or the HD version.
- A complex version of the asset. Edited for family or aircraft viewing.
- A group of versions of the asset. All the mobile phone versions or the region 1 DVD versions.

Note how we are using the word *asset* and not the word *file*. You will see this is important when we come to talk about real implementations later. Also note that most of the uses of the term asset refer to anything from abstract metadata to several physical files. Most asset management systems have developed ways to handle

this complexity inside their databases. Prior to AS-02, no one had yet standardized the factory format that allows any two vendors to build products to that specification and have their equipment work out-of-the-box.

AS-02 is a stable, specified way of using MXF that leads to efficient media factory design. It was originally designed by a group of ten companies to meet the needs of a single user who had a vision of an ultra-efficient media factory and has now been deployed in several facilities around the world. The uses of a media file given above were considered and application rules were written that allow devices to use MXF (was SMPTE 377M-2004, now updated to SMPTE 377-1-2009) to achieve all of the use cases.

Contents

1 Scope

AMWA's application specification AS-02 defines the building blocks of interoperable media factories, the media facilities where the versioning of assets takes place. The specification provides a lean and efficient approach to working with file-based media that is intended to reduce waste as part media facility operations, e.g. by avoiding unnecessary copy operations or the re-keying of metadata. Rather than specifying a new file format, AS-02 provides constraints on the structure and use of standardized MXF files to provide new opportunities for interoperability both within and between media facilities.

The design of AS-02 recognizes that individual facilities are likely to have some variations, specifically in the types of codecs and essence that are allowed. Also, variation is expected in the specific metadata rules and audio channel arrangements. For that reason, the AS-02 specification includes a *shim* specification that takes the rules of AS-02 and further constrains them for use in a facility. This has the goal that manufacturers can build MXF-compliant equipment, conforming to the AS-02 specification, and they know that properties subject to shim specification need to be exposed to the end user. This allows the end user to build a customized workflow by configuring a piece of generic software / equipment and yet retain the interoperability that comes with a constrained open standard.

Note: In a managed AS-02 environment within a single facility, the physical layout rules of AS-02 may be relaxed. The rules shall not be relaxed for business-to-business interchange.

2 Conformance language

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: the Introduction, any section explicitly labeled as "informative" or individual paragraphs that start with "Note:"

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword "reserved" indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword "forbidden" indicates "reserved" and in addition indicates that the provision will never be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

Unless otherwise specified, the order of precedence of the types of normative information in this document shall be as follows: Normative prose shall be the authoritative definition; Tables shall be next; followed by formal languages; then figures; and then any other language forms.

3 Reference documents

The following standards contain provisions that, through reference in this text, constitute provisions of this recommended practice. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this recommended practice are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

SMPTE 330M-2004 Television – Unique Material Identifier (UMID)
SMPTE 377-1-2009 Material Exchange Format (MXF) – File Format Specification
SMPTE 378M-2004 Television – Material Exchange Format (MXF) – Operational Pattern 1a
SMPTE 379-1-2009 Material Exchange Format (MXF) – MXF Generic Container
SMPTE 380M-2004 Television – Material Exchange Format (MXF) – Descriptive Metadata Scheme-1
SMPTE 381M-2005 Television – Material Exchange Format (MXF) – Mapping MPEG Streams into the MXF Generic Container
SMPTE 382M-2007 Material Exchange Format - Mapping AES3 and Broadcast Wave Audio into the MXF Generic Container
SMPTE 391M-2004 Television – Material Exchange Format (MXF) – Operational Pattern 1b
SMPTE 393M-2004 Television – Material Exchange Format (MXF) – Operational Pattern 2b
SMPTE 407M-2006 Television – MXF – Operational Patterns 3a and 3b
SMPTE 410-2008 Material Exchange Format - Generic Stream Partition
SMPTE 436-2006 Television – MXF Mappings for VBI Lines and Ancillary Data Packets
SMPTE 2029-2009 Uniform Resource Names for SMPTE Resources

W3C Extensible Markup Language (XML) 1.0 (Fifth Edition)
W3C Namespaces in XML 1.0 (Third Edition)
W3C XML Schema Part 0: Primer (Second Edition)
W3C XML Schema Part 1: Structures (Second Edition)
W3C XML Schema Part 2: Datatypes (Second Edition)

IETF RFC 1321 The MD5 Message Digest Algorithm
IETF RFC 1738 Uniform Resource Locators (URL)
IETF RFC 2104 HMAC: Keyed-Hashing for Message Authentication
IETF RFC 3174 US Secure Hash Algorithm 1 (SHA1)
IETF RFC 4122 A UUID URN Namespace

4 Overview

4.1 File format requirements (informative)

AS-02 addresses the problem of having a common file format in a facility that has to handle many input formats and make many output formats.

In an ideal world, a single essence representation would be standardized on. Unfortunately, over time, requirements for a facility change. The facility that started as SD introduces HD. The compression format of today gets replaced with a better one from tomorrow. Today's stereo audio gets upgraded to multi-channel and multi-lingual. In parallel, a facility operates with a constant pressure to reduce the cost of production, distribution and publishing.

In order to build systems that work at the MXF level rather than at the MPEG, JPEG2000 or DV level, AS-02 is an application specification providing a set of rules that MXF encoders and MXF decoders must follow. This

brings many advantages. It allows a device to query the image size without having to implement a decoder for each essence type – you simply read the MXF. It allows a business process to be specified with an action such as:

If (16:9) then transcode(centre-cut-out)

This action can be specified without the executor of process, human or system, needing to know the details of underlying codec type – that is something for the transcoder to figure out. In general, this allows business systems to deliver work order requests to content-manipulation devices, which are in turn configured to support delivery specifications.

4.2 General AS-02 and shims

To maximize commonality across applications, this specification is divided into general provisions that apply to all applications and specific constraint sets, called *shims*, that apply to defined applications.

General provisions shall apply to all AS-02 files and thus represent the maximum required capability of ingest servers, catch servers, playout servers, transcoders and other AS-02 compliant devices.

Each shim shall provide a further set of constraints that reduce the range of variability that may be needed in well-defined categories of applications. For example, these categories may address particular type of programming or programming genres, or they may address requirements of particular broadcast station groups.

Shims shall not be used to add new required capability to the general provisions. They shall be limitations on the general provisions. Thus, the general provisions may be non-restrictive in some areas and this is intentional.

In any given facility design or interchange specification based on AS-02, there may be one or more tightly defined shims. Typical examples are to have a shim for SD content, a shim for HD content and a shim for web content. Each of these shims tightly defines the codecs and layout for the particular business application.

In this specification, shim parameters shall be highlighted as follows:

Shim parameter	Definition
example	Semantic definition of the example shim parameter

4.3 Asset structure, versions and bundles

4.3.1 Introduction to the structure (informative)

MXF AS-02 is a component-based format. A single *essence component file* will exist for each video, audio and data element. Interleaving components together (as in SMPTE 386M D-10) is not allowed. A version file in MXF format represents individual versions. Customized metadata and application specific files may be stored within the folder structure in defined locations.

The general structure of an AS-02 asset is folder-based. The group of files is referred to as a *bundle*. The general structure of an AS-02 bundle is shown in figure 2 below.

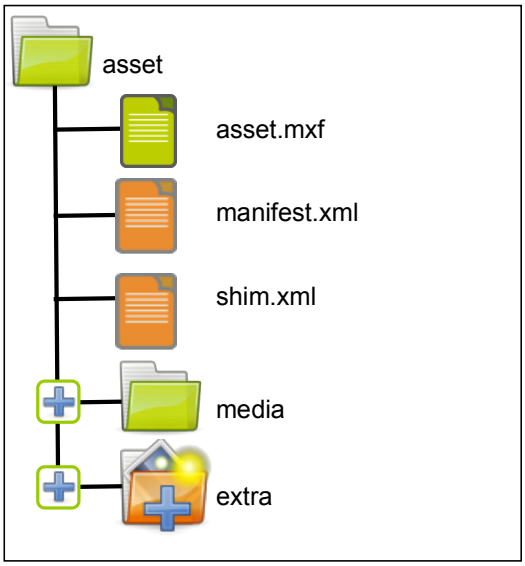

Figure 2: AS-02 file structure – a bundle

In IT file-based operations, reference to the asset is by its root folder. Within that folder will exist one or more version files that reference media stored in the media subfolder, as shown in figure 3.

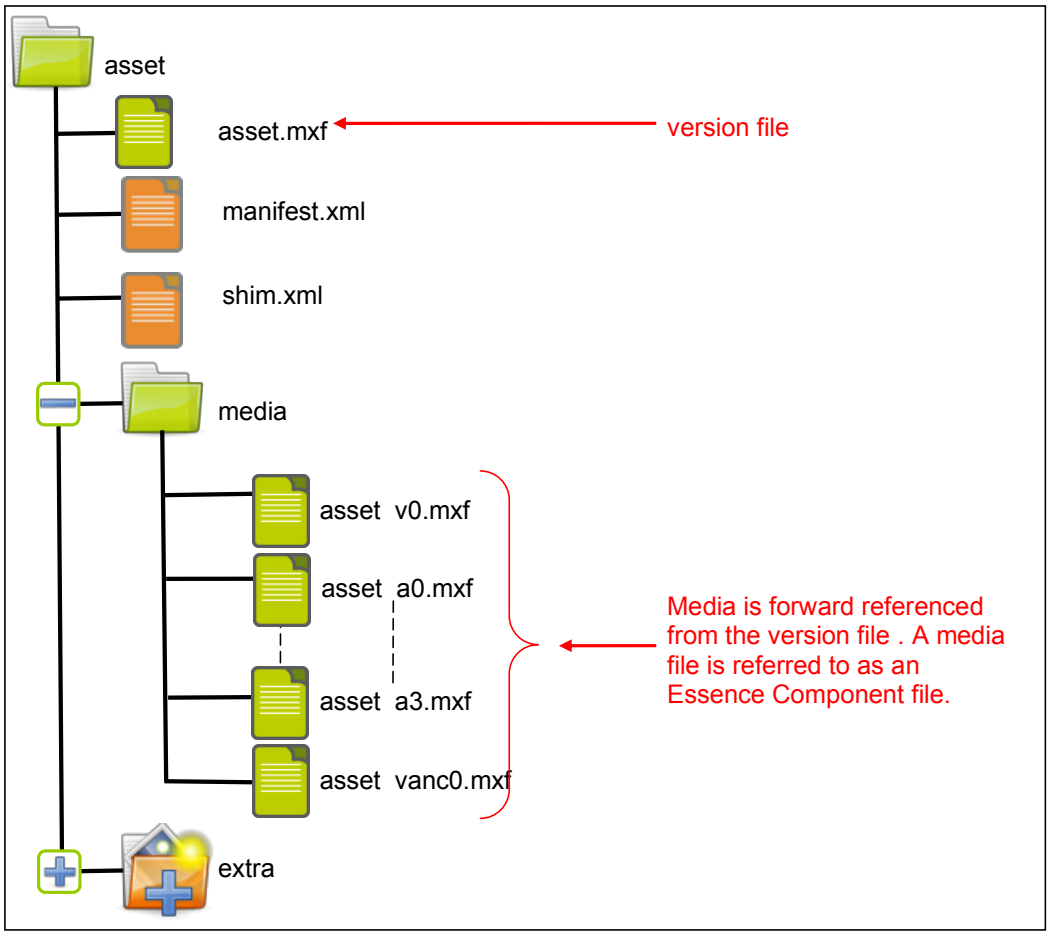

Figure 3: Media folder contains MXF essence

Extra metadata and content that is not MXF-wrapped is stored in the extra folder. An example of a typical extra folder is shown in figure 4. You can see in the figure that any customized metadata type is permitted within AS-02, but may be further restricted within a shim.

The root folder contains two XML documents that are used to manage the asset. The first of these is the manifest (section 9) that lists all the files in the bundles. The second of these is the shim (section 10) that provides an identifier for the shim that was used at the time the bundle was created or last modified.

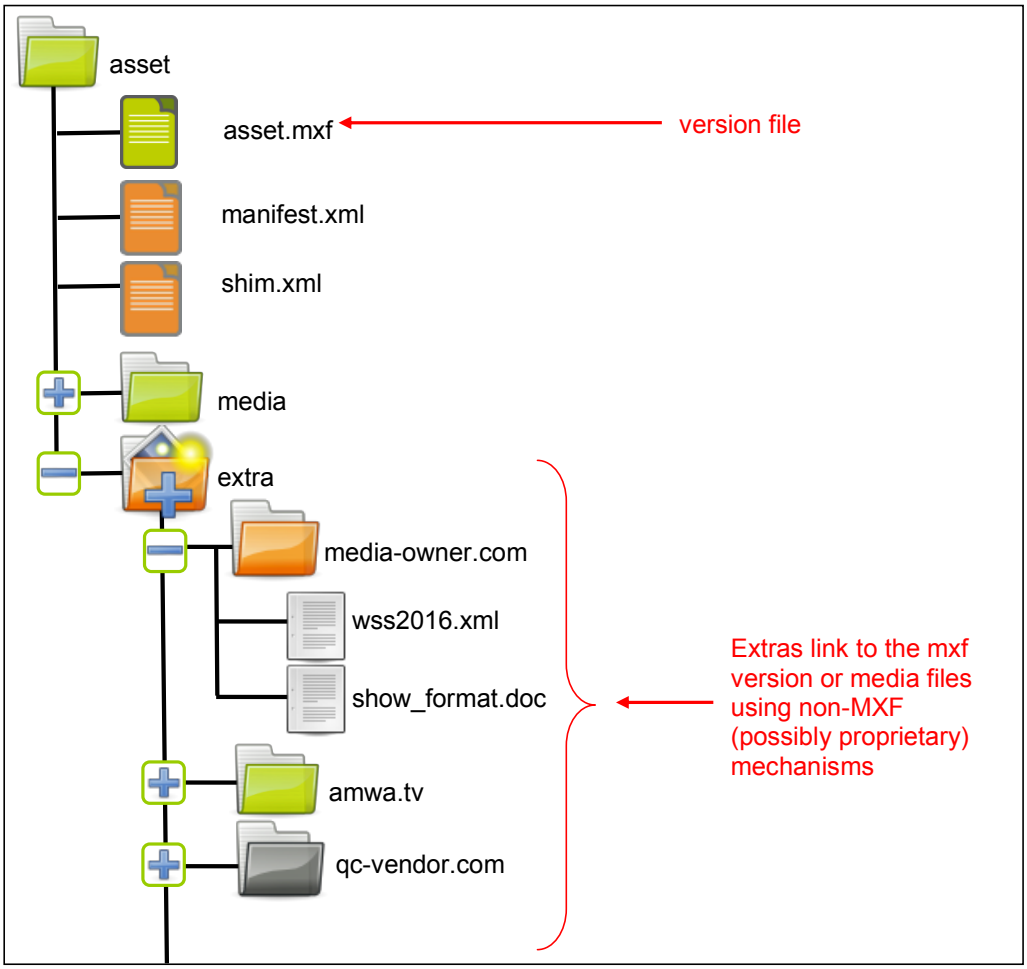

Figure 4: Extra folder contains non-MXF metadata and content

4.3.2 MXF and non-MXF operations (informative)

AS-02 shall permit both MXF operations and non-MXF operations.

An MXF operation is one that involves parsing the MXF metadata. Examples of MXF operations include: play; jog; seek; record; add track; remove track; create version; remove version and essence; find unreferenced essence components; extract simple version etc.. All MXF operations shall result in files that comply with the MXF suite of specifications.

A non-MXF operation is one that does not involve parsing the MXF metadata. Examples of non-MXF operations include: move bundle; transfer bundle (FTP); package bundle (zip); checksum bundle; purge extra files; add extra file; update manifest.

4.3.3 AS-02 asset structure

The various essence components and metadata collections of an AS-02 asset should be logically organized as a folder structure. The folder structure shall be referred to as an AS-02 *bundle*.

In a file system, the name of the root folder shall be used as the file-system name of the AS-02 bundle (see section 4.3.6). Systems may use this property to construct Uniform Resource Locators (URLs, RFC 1738) from the file system pathname of the bundle.

In an asset management system, the "BundleID" field of the manifest file shall indicate the identity of the bundle.

In addition to the essence component files, AS-02 assets include metadata that describe different versions of the asset. Each version shall be described in a separate version file.

4.3.4 Essence component file

An essence component file shall be a media file in the media folder with the following properties that are further described in section 5.

1. An essence component file shall be a mono essence MXF file.
2. An essence component file shall be frame wrapped for video and SMPTE 436M essence.
3. An essence component file shall be clip wrapped for PCM audio.
4. An essence component file shall be regularly partitioned when frame wrapped.
5. Each body partition in an essence component file shall have metadata, index tables or essence.
6. An essence component file shall be signaled as closed and complete in the header or footer partition.
7. An essence component file shall have distributed and complete index tables.
8. An essence component file shall be signaled as OP1a.
9. An essence component file should not have a system item.
10. An essence component file shall have a KAG size of 1.
11. An essence component file shall have the primary package set to the identity of the top-level file package within the essence component file. The UMID of the top-level file package in the essence component file shall be the unique identifier of the essence component.
12. For non-MXF operations, a filename specified as the pathname of the file relative to the root of the essence component file within a bundle shall be used as a substitute for the UMID. For example: "media/a03.mxf".

Note: By using the top-level file package identifier as both the primary package identifier and external identifier of an essence component file (as used in the manifest file), a decoder can locate a matching media MXF file for a specific component. As a result, it is not necessary to parse every essence component file in the bundle first.

4.3.5 Version file

A version file shall contain only a single MXF material package and shall have the following properties, each of which is fully explained in section 5.

1. A version file shall be a valid MXF file with external essence: OP1b, OP2b or OP3b.
2. An individual version of an AS-02 asset shall have the primary package property set to the UMID of the material package in the version file. The UMID of the material package in the version file shall be the unique identifier of that version.
3. For non-MXF operations, the filename of the version file within a bundle shall be used as a substitute for the UMID. Note: No path component is required as the version file is in the root folder.
4. If an "original" version, "primary" version or "default" version exists, it shall be stored in a version file that has the same filename as the root folder and shall have a ".mxf" extension.
5. AS-02 version files reference the file package UMIDs of the essence components. AS-02 version files shall not reference essence components outside of the bundle.
6. Essence components in the media folder may exist that are not referenced by any version file.

7. A version file may reference any of the essence components in the media folder.
8. Multiple references may be made to some essence components.
9. In some communities, the term "composition" is used as a substitute for the term "version". The term "AS-02 composition file" shall be treated as equivalent to the term "AS-02 version file".
10. In applications where non-MXF version information is required, it shall be stored in an XML file in the root folder and shall have the same name as the related .mxf file that stores the media synchronization information.
11. Version files may refer to multiple video, audio, and data tracks. In the case that they do, the precedence of the tracks shall be determined by the MXF track number property. Unique and non-zero track numbers shall be used.

4.3.6 Simple version file

A *simple version file* is intended for simple bundle applications that do not require segmentation (OP2b) or editing (OP3b). Additional constraints over and above section 4.3.5 are listed below and clarified further in section 5.

1. A simple version file shall be a valid MXF file with external essence: OP1b.
2. A simple version file should contain metadata to describe the intended usage of each essence component, e.g. audio languages, video keys or caption service.
3. All essence components shall be referenced only once.
4. All essence component files shall have the same duration measured in edit units as required by the OP1b specification.

Note: The intent of a simple version is that any software or device acting on that version shall use all of the referenced tracks.

Note: OP1b is specifically to be used as the operational pattern for simple version files, even if only a single essence component is contained in the bundle and the version file would also be valid as an OP1a file.

Note: With reference to point 4, the number of audio samples in the underlying essence container may vary slightly in the case of non-integer number of audio samples per video frame.

4.3.7 Bundle

All the files belonging to the asset shall be stored beneath a single root folder of a bundle. Figure 2 shows the logical structure of an AS-02 asset bundle.

The bundle of files shall have the following properties:

1. The name of the root folder shall be considered as the identification filename of the bundle.
2. All version files shall be placed in the root folder.
3. All essence components that can be referenced by a version file shall be placed in a subfolder called "media". Note: This includes unreferenced essence that may be re-linked at some later date.
4. All other files not referenced by a version file shall be placed in a subfolder called "extra".
5. Other subfolders may exist, but are not used in this application specification.
6. File system linking mechanisms shall not be used within the AS-02 bundle, i.e. all files under the root folder shall be physically stored there.

The maximum number of versions in a bundle may be constrained by the *max_versions* shim parameter.

Shim parameter	Definition
max_versions	Default value: **unlimited** Type: positive integer and unlimited Values: 1 to unlimited Maximum number of version files permitted in a bundle.

4.3.8 Simple bundle

A *simple bundle* shall be used for single version interchange and is restricted as follows:

1. All AS-02 constraints shall apply unless overridden here.
2. Only one simple version file shall exist in a simple bundle.
3. The simple version file shall reference all the essence components in the media folder.
4. The simple version file shall have the same name as the root folder.

Note: This bundle is intended for use in single version applications, such as on a playout server. It is intended for deterministic access, easy essence tracking and easy deletion.

In a simple bundle, each of the essence component files shall be referenced from at most one OP1b program version header.

Note: This provides for a simple and error-free content aging / content deletion strategy that does not require reference counting.

4.3.9 Extra folder

The *extra folder* shall provide a facility to associate additional metadata and associated files within the AS-02 bundle.

Metadata describing a show layout (position of the breaks), custom XML, QC reports and other associated files shall be placed in the extra folder or in a subfolder thereof.

In general, documents found in the extra folder should contain some linkage back to one or more of the MXF versions or components. This linkage may be via:

- Recommended practice - the UMID of the version file's material package;
- Common usage - a filename specified as the pathname of the file relative to the root;
- Permissible but very weak linkage - by the presence of the file in the bundle.

Subfolders within the extra folder shall be used to manage individual metadata files to prevent filename collisions and the inadvertent overwriting of files.

It is recommended that individual business should put their private metadata in a folder with a name corresponding an established Internet domain name. For example:

- AMWA metadata is stored in: extra/amwa.tv/
- Sony metadata would be stored in: extra/sony.com/
- BBC metadata would be stored in: extra/bbc.co.uk/

The management of the data in each business folder is outside the scope of this specification but may form part of an individual shim.

4.3.10 Extra folder – role of a file

In many workflows, tools from multiple vendors intrinsically share files that exist in the extra folder. The AS-02 specification provides a mechanism that shall be used for specifying the roles of these files within a manifest. These roles shall be as specified in Annex B. The use of these role identifiers is application-specific and should be mandated in the shim document if required.

A specific example of a role is the provision of caption source files. In a multi-lingual workflow, it may be required to keep a number of ".cap" or ".scc" or ".dxfp" files with the asset so that, at some point in its lifecycle, captions may be rendered to the broadcast, web or IPTV deliverables. These files may be stored in any appropriate subfolder on the extras folder. Their role should be signaled as *caption* in the manifest file.

4.3.11 Manifest file

The root folder of the bundle shall contain a manifest file (manifest.xml). The manifest file contains a list of all of the files and folders in the bundle, including the version and essence component files, with their identifiers and relationships. The manifest file shall also contain a unique identifier that shall provide the identification of the bundle as a whole. See section 9 for the specification of the format of this file.

4.3.12 Shim file

The root folder of the bundle shall contain a file (shim.xml) that contains all of the shim parameters and their values for the bundle. See section 10 for the specification of the format of this file.

5 Version file parameters and constraints

5.1 Provision categorization

Each provision within the general specification and within each individual shim is categorized as one of the following:

- Unconstrained: everything permitted by SMPTE 377-1-2009.
- Gently constrained: a range of values (for example, bitrates) or choices (for example, descriptive metadata schemes or essence types) is stated by the general AS-02 specification, that individual shims may further restrict.
- Strongly constrained: a range of values or set of choices is listed that individual shims must choose between.
- Fully constrained: a single choice or parameter value that all AS-02 applications will use identically

Shims shall always express constraints that are equal to or stronger than the general specification.

The sections below define the general provisions that shall apply to all AS-02 files. Named parameters are specified whose values shall define further constraints that apply in specific shims. A distinctive font is used for the names of ***parameters***. To enable automated processing of shims, the names are chosen to be unique across the AS-02 namespace, and are syntactically valid XML NCNames.

5.2 No essence in the version file

There shall be no essence containers in the version file.

Shim parameter	Definition
generic_streams_in_version	Default value: **false** Type: boolean Values: true, false When true, there may be generic stream containers in the version file.

Note: An *essence container* refers to the use of the MXF Generic Container to carry a stream of picture or sound essence. Use of the MXF Generic Container to carry generic data streams in version files is acceptable if the shim parameter defined above is set to true.

5.3 Partitions carrying header metadata in the version file

Header metadata in header and footer partitions shall follow the provisions of 377-1 section 7.5.

5.4 MXF header constraints in a version file

The following metadata constraints shall apply to the version file. A version file may contain segmentation and editing information.

<table>
<tr><th>Set name</th><th>Constraint</th></tr>
<tr><td>Preface</td><td>As per SMPTE 377-1-2009 revision. Primary package shall be a material package.</td></tr>
<tr><td>Identification</td><td>There shall be 1 identification set added by each application altering the file.</td></tr>
<tr><td>Content Storage</td><td>There shall be 1 material package, 0 or more top-level file package(s) per essence component, no restriction on lower level packages. Lower level source packages should be preserved in the version file by applications that edit or modify component files.</td></tr>
<tr><td>Essence Container Data</td><td>Shall not be present. All essence shall be externally referenced.</td></tr>
<tr><td>Operational Pattern</td><td>Shall be signaled as OP1b, OP2a or OP3a for single-track version files.
Shall be signaled as OP1b, OP2b or OP3b for multi-track version files.</td></tr>
<tr><td>Material Package</td><td>Shall be compliant with OP1b. References shall only be made to file packages within the version file. Unresolved UMID references shall not be permitted.</td></tr>
<tr><td>Source Package (File / Physical)</td><td>There shall be a copy of the file package from each of the essence component files in the version file.</td></tr>
<tr><td>Sequence (all cases)</td><td>For a closed file, the durations shall be correct in the header metadata. Open files may use distinguished values as described in 377-1.</td></tr>
<tr><td>SourceClip (Picture, Sound, Data)</td><td>The duration rules shall be the same as for the version file sequence as above.</td></tr>
<tr><td>Network Locator</td><td>One network locator shall exist for each file package. The path shall be a URI relative to the version file and shall be correct at the time that the file was created.
Decoders shall use the network locator(s) as a hint and the UMIDs shall always be used to verify the integrity of the network locator value. If the network locator does not point to the essence component file, the UMIDs shall be resolved via some other mechanism.</td></tr>
<tr><td>Text Locator</td><td>Optional.</td></tr>
<tr><td>Timecode
Timeline Track SubClip</td><td>The timecode properties shall be compliant with MXF 377-1-2009.
<table>
<tr><th>Shim parameter</th><th>Definition</th></tr>
<tr><td>sub_clip_limit</td><td>Default value: 1 frame
Type: free text
The smallest duration that can be used in a material package timeline source clip property.
When complex essence types are in use, such as for long-GOP MPEG, this parameter needs to specify whether or not sub-clips may start or end within GOPs / access units or other structural elements of the encoding scheme.</td></tr>
<tr><td>sync_cut</td><td>Default value: false
Type: boolean
When true, material package source clips are constrained such that all tracks shall cut synchronously. This effectively constrains version files to be OP2b butt-edits of pre-conformed clips. When false, the version file may become an OP3b edit decision list.</td></tr>
</table></td></tr>
</table>

5.5 MXF header constraints in a simple version file

The following additional constraints apply to a simple version file

Set name	Constraint
Content Storage	There shall be 1 top-level file package per essence component.
Material Package	Shall be compliant with OP1b.

5.6 Closed and complete metadata in the version file footer partition

During update of a version file, the header of the file may be in the progress of being updated. Applications reading the file should atomically read the closed and complete header metadata partition (it is small) and shall use that version of the metadata.

If it is found that the version file header is open and the footer is closed, then this should be indicative of an update in progress and appropriate caution should be taken.

5.7 No essence stream index tables in a version file

Essence stream index tables shall not be included in any partition of a version file.

Generic stream index tables may be included for data tracks.

5.8 Version file KAG size of 1

The KAG size of a version file shall be 1.

5.9 Minimum simple version duration

Shim parameter	Definition
min_sv_dur	Default value: **10 seconds** Type: floating point Values: gently-constrained The minimum simple version duration parameter ensures that in an application specification, performance criteria can be specified that reflects the operation of a facility. For example, simple versions with a duration value of a single frame may not play out correctly.

5.10 Media integrity check for essence components

Shim parameter	Definition
mic_scope	Default value: **essence_only** Type: enumeration Values: essence_only, entire_file, no_mic Version files may contain media integrity check information of the entire essence component file (entire_file), just the essence stream (essence_only) or not at all (no_mic). Essence component files shall only contain this metadata when the version files sign essence_only.
mic_type	Default value: **crc32** Type: list of permitted enumerations Values: HMAC-SHA1, crc32, crc16, md5 Version files may contain a media integrity check using one of the valid mechanisms.

5.11 Channel mapping with track numbers

Shim parameter	Definition
track_numbers	Default value: **false** Type: boolean Values: true, false When using audio components in an application that involves physical mapping of audio to ports on equipment, it is necessary to identify which of the audio essence components are mapped to which output channel. This is done via the track number property in the material package tracks. A true value indicates that track numbers shall be present and may be used by equipment. A false value indicates that track numbers may exist in the file, but should not interpreted by equipment

5.12 Timecode

Shim parameter	Definition
tc_mode	Default value: **file_specific** Type: enumeration Values: DF, NDF, file_specific. The timecode counting mode of a version file may be specified so that file writers create consistent outputs. Drop Frame (DF), non-Drop Frame (NDF) or file_specific values are permitted. When working with 50Hz material, the value NDF is recommended. When working with 60Hz or 24Hz material DF or NDF is recommended according to the working practises of the facility. A value of file_specific is intended for applications and facilities where DF or NDF cannot be known apriori.

5.13 Descriptive metadata parameters and constraints

5.13.1 SOM/EOM

Specific start-of-message (SOM) and end-of-message (EOM) metadata shall be provided in AS-02 version files using sequences of source references on MXF OP1, OP2 or OP3 essence tracks in the material package.

AS-02 version and essence component files may contain DMS-Segmentation metadata tracks.

Note: AS-03 files shall use only OP1a and may contain DMS-Segmentation metadata tracks.

5.13.2 Other descriptive metadata schemes

Private application-specification-defined schemes and extensions to SMPTE 380M DMS-1-based schemes shall be identified in the preface.

6 Essence component file parameters and constraints

Note: Shim constraints for essence components are expressed in the same manner as for version files in section 5.

6.1 Essence track parameters and constraints

6.1.1 General

AS-02 essence component files shall each contain a single essence component of a single essence type. They shall be subject to the general constraints listed in section 4.3.4 with the following additional constraints below.

- Header partition shall contain only metadata.
- On creation, an 8kByte fill shall follow the header partition to allow for in-place extension.
- When the essence is a specific variant, it should be annotated with metadata.
- All data sets shall follow MXF generation ID rules.

These parameters are described in the sections below. Parameters may be further constrained by shims as described in the annexes or external documents.

6.1.2 Mono essence

Only one essence track shall exist in the file package. Only one type of essence element shall be present (i.e. data, sound, picture or compound) in the essence container.

In AS-02 essence component files, bit 3 of byte 15 of the operational pattern identifier shall be set to zero to represent uni-track, signaling that every essence container has one and only one essence track.

6.1.3 Interleaving

AS-02 files shall not contain interleaved essence components.

6.1.4 Partitions

A partition in an AS-02 compliant essence component file shall only have one of the following types of data in it:

- Header metadata;
- Essence;
- Index table;
- SMPTE 410-2008 generic stream.

Shim parameter	Definition
partition_spacing	Default value: **60 seconds** Type: floating point Values: strongly-constrained for frame wrapped files Body partitions in a frame wrapped file shall occur at regular temporal spacing indicated by this parameter. Regular shall mean that any variation in the regularity shall be less than the smaller of 1 second or 10% of the period ***partition_spacing***.

Clip wrapped files shall have the following structure: a header partition; one body partition with an index table; one body partition with clip wrapped essence; a footer partition. This is shown in figure 5.

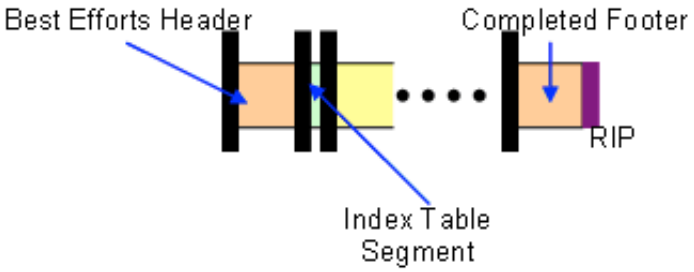

Figure 5: Clip wrapped file partition structure

A RIP shall be present in all essence component files.

6.1.5 Index tables

Frame wrapped essence shall have distributed index tables to aid process-during-write applications.

Shim parameter	Definition
index_strategy_frame	Default value: **lead** Type: enumeration Values: lead, follow or file_specific At each partition point in a given frame wrapped essence component file, the index partition shall either lead (i.e. precede) the essence partition that it indexes or follow the essence partition that it indexes. This shall be specified application by application. The “file_specific” option covers applications where the strategy cannot be defined.
index_strategy_clip	Default value: **lead** Type: enumeration Values: lead, follow or file_specific As above, but applies to clip wrapped files. When the index tables are calculated (PCM audio), lead means that the calculated index table immediately follows the header partition.

6.1.6 Video

Video essence in AS-02 files may be of one of a selection of compression families and acceptable variability should be defined by individual shims.

Shim parameter	Definition
picture_family	Default value: **none** Type: list of strings A list of supported video essence compression families including any family-specific elements such as GOP structures and profiles, for example “MPEG-2 H.264 JPEG2000”.
picture_bitrate	Default value: **none** Type: list of integer ranges A list of supported bitrates or ranges of bitrates for compressed video.
picture_format	Default value: **none** Type: list of formatted strings A list of permitted video frame sizes and rates, for example “1920/1080/50p/16:9”.
picture_component_limit	Default value: **1** Type: non-negative integer or unlimited When a number is given, specifies the maximum number of video essence component files that can be simultaneously decoded from an AS-02 bundle. Note: This parameter allows a facility to document the capacity of their decoding devices, such as playout servers.

Video files shall be frame wrapped with complete index tables.

6.1.7 Audio

PCM audio files shall contain either mono audio or stereo audio or multi-channel audio. All kinds shall be considered as mono-essence and shall be described by a single track in the file package. PCM audio channels from a different soundfield shall be in different files, for example for different languages.

Note: A *soundfield* is defined as the acoustical space in which the intended audio image is created. This definition is expected to appear in the SMPTE multi-channel audio specification. Therefore, you would not expect different languages to be present within the same soundfield.

Note: In existing digital cinema workflows, it is common to find six tracks of audio from a single soundfield combined into a single PCM stream.

Stereo and multi-channel audio may be represented as separate PCM audio files. The media file of a bundle may contain both a combined file and separate files for the same audio, with version files used to select the most appropriate representation.

Multi-channel audio carried as data in an audio file (e.g. inside a SMPTE 377-1-2009 data container) shall be in a single essence component file and shall be described by a single MXF track in the file package.

Audio from a compound essence containers representing more than one soundfield shall be extracted into a separate essence component files to ensure AS-02 compliance. The audio within a compound essence container shall be ignored in AS-02 operations.

Compressed audio files may contain multiple channels within a single soundfield, for example 5.1. Compressed audio files shall be described by a single track in the file package.

Shim parameter	Definition
audio_family	Default value: **none** Type: list of strings A list of supported audio essence families including bitrates, permitted soundfield arrangements, compression types and any format specific elements such as delay setting.
audio_file_arrangement	Default value: **none** Type: free text A description of the physical storage strategy for audio. Typically, this will be of the form “mono only files”, “stereo only files”, “stereo files with optional Dolby E in a file” etc.
audio_component_limit	Default value: **unlimited** Type: non-negative integer or unlimited When a number is given, specifies the maximum number of audio essence component files that can be simultaneously decoded from an AS-02 bundle. Note: This parameter allows a facility to document the capacity of their decoding devices, such as playout servers.

PCM audio files shall be clip wrapped with a calculated index table.

Constant bitrate compressed audio files shall be clip wrapped with a calculated index table.

Variable bitrate compressed audio files shall be clip wrapped with a variable bitrate (VBR) index table.

6.1.8 Closed captioning and subtitles

Closed caption data and subtitling data carried in VBI or VANC shall be frame wrapped with complete index tables and shall not be interleaved in the video file. This data shall be stored in a SMPTE 436M essence component file in the media folder. VBI data and VANC data may appear in the same SMPTE 436M essence component file. Non-caption data may also exist in the SMPTE 436M essence component file. Different languages for captions shall be carried in separate files.

The intent of the AS-02 specification is that SMPTE 436M shall be used for the carriage of captioning, subtitles, VANC, and VBI data. During the adoption period, it is understood that legacy formats will exist, where this data is contained in a large number of legacy file types.

The following shim parameters shall be used to indicate when legacy captioning or subtitle formats are used in the bundle.

Shim parameter	Definition
CC_data_essence	Default value: **none** Type: list of strings A list of supported data essence types for closed captioning including specific parameters, such as VBI lines.
CC_custom	Default value: **false** Type: boolean When true, VBI and / or VANC for closed caption data shall be encapsulated in the video essence using a defined method (e.g. carriage in MPEG picture user data). This data shall also be placed in a separate SMPTE 436M essence component file. Note: This provision is to support legacy playback devices and should be avoided wherever possible as it can leads to the SMPTE 436M captions and it custom copy changing independently.

Shim parameter	Definition
CC_render	Default value: **false** Type: boolean When true, VBI data for closed captions shall be encoded as active video within the video image. This data should also be placed in a separate VBI essence component file. Usually, this is only true for SD images that are coded as “tall MPEG” (i.e. the VBI area is in the active picture).
CC_component_limit	Default value: **unlimited** Type: non-negative integer or unlimited When a number is given, specifies the maximum number of closed caption files that can be merged when decoding an AS-02 bundle. Note: This parameter allows a facility to document the capacity of their decoding devices, such as playout servers.

6.1.9 Other VBI and ANC data

Other VBI and VANC data shall be carried in SMPTE 436M, as per section 6.1.8. Any data in a separate SMPTE 436M file shall override any data of the same type carried within the video file.

Note: To preserve data that is intimate with a timeline essence component, copy the VBI and VANC data into a separate file and when creating an AS-02 bundle. The original data should be deleted from where it is originally sourced to avoid it leaking back into the workflow.

Different types of VBI and VANC data should be split up into separate files. Decoders should be able to merge different kinds of data, for example on playout.

The following shim parameters shall be used to indicate when legacy VBI or VANC formats are used in the bundle.

Shim parameter	Definition
VBI_data_essence	Default value: **none** Type: list of strings A list of supported data essence types including specific parameters such as VBI lines supported.
VBI_custom	Default value: **false** Type: boolean When true, VBI data shall be encapsulated in the video essence using a defined method (e.g. carriage in MPEG picture user data) as well as being present in a separate VBI essence component file.
VBI_render	Default value: **false** Type: boolean When true, VBI data shall be encoded as active video within the video image. This data should also be placed in a separate VBI essence component file. Usually, this is only true for SD images that are coded as “tall MPEG” (i.e. the VBI area is in the active picture).
ANC_data_essence	Default value: **none** Type: list of strings A list of supported data essence types including specific parameters such as ANC packet types supported.
ANC_custom	Default value: **false** Type: boolean When true, ANC data shall be encapsulated in the video essence using a defined method (e.g. carriage in MPEG picture user data) as well as being present in a separate ANC essence component file.

Shim parameter	Definition
ANC_render	Default value: **false** Type: boolean When true, VBI data shall be encoded as active video within the video image. This data should also be present in a separate ANC essence component file.
data_component_limit	Default value: **unlimited** Type: non-negative integer or unlimited When a number is given, specifies the maximum number of VBI or VANC files that can be merged when decoding an AS-02 bundle. Note: This parameter allows a facility to document the capacity of their decoding devices, such as playout servers.
data_separation	Default vale: **true** Type: boolean When true, indicates that all data essence component files for VANC and VBI data shall be split into separate files. When set to false, all VANC and VBI data is merged into a single essence component file. Note: This specification does not permit a mixture of approaches.

6.2 Header metadata and operational pattern constraints

6.2.1 Baseline operational pattern

The operational pattern of an AS-02 essence component file shall be signaled as OP1a.

The qualifier bits of the operational pattern in AS-02 essence component files shall be set as follows:

- bit 1: internal essence set to "0" to signal internal essence only.
- bit 2: streamable set to "0" to signal that the essence in the file is streamable.
- bit 3: uni-track set to "0" to signal that the essence in the file is atomic, i.e. there is only one track.

Note: This provision is intended to maximize compatibility with legacy MXF decoder applications.

6.2.2 Essence container label

AS-02 essence component files shall each specify essence container labels that shall correspond to the essence container used in that file.

6.2.3 System item

The generic container system item may be present but should not be used by AS-02 files. Metadata in the system item that is to be used in an AS-02 operation should be copied or moved to other containers such as the header metadata or SMPTE 436M data tracks. The following recommendations may be applied in a shim:

Shim parameter	Definition
sys_item_tc	Default value: **false** Type: boolean Indicates whether it is acceptable for the essence component file to carry timecode in the generic container system item. This provision is intended to capture source timecodes and does not affect the material package of any version file. Whatever the parameter is set to, timecode in any generic container system shall be copied to the file package header metadata, including any discontinuities therein.

6.2.4 Timecode

The timecode of an AS-02 version shall be controlled in the version file. There may be several timecode tracks in the header metadata of an essence component. Timecode mode - drop-frame or non-drop frame - may be specified in each shim.

6.2.5 Random index pack

All AS-02 version and essence component files, when closed and complete, shall contain a random index pack as per SMPTE 377-1-2009.

6.2.6 KAG size

AS-02 essence component files shall have a KAG size of 1 unless this requirement conflicts with an underlying essence container specification. When a conflict exists, the value in that essence container specification shall be used.

6.3 Header metadata parameters and constraints

The header partition of an essence component file shall start at the first byte of the file, with no run-in. Header metadata shall exist in the header partition and may have optional fill. All files shall comply with SMPTE 377-1-2009.

The footer of an essence component file may contain a copy of the header metadata and it shall be marked closed and complete. All files shall have a footer partition unless:

1. the file is in the process of being written;
2. the underlying essence container specification forbids the existence of a footer.

Note: Once a published standard, KLV extension syntax (SMPTE 377-2) shall be used for MXF metadata extensions.

Any specific constraints on the header metadata shall be:

<table>
<tr><th>Set name</th><th colspan="2">Constraint</th></tr>
<tr><td>Preface</td><td colspan="2">As per SMPTE 377-1-2009. Primary package shall be file package.</td></tr>
<tr><td>Identification</td><td colspan="2">Each device altering the file shall add 1 identification set.</td></tr>
<tr><td>Content Storage</td><td colspan="2">1 material package, 1 top-level file package, no restriction on lower level packages. Lower level source packages shall be preserved by applications that edit or modify component files.</td></tr>
<tr><td>Essence Container Data</td><td colspan="2">One single entry corresponding to the single generic container. BodySID and IndexSID shall be present and shall be different.</td></tr>
<tr><td>Material Package</td><td colspan="2">Compliant with OP1a mono essence – shall contain only a single essence track.

Shim parameter | Definition
track_tag_policy | Default value: none
Type: free text
Specifies how material package tracks in the essence component file are tagged with metadata, describing a business rule of the presence of absence of tags. This specification may be a private xml specification or a SMPTE specification.
For example: "SMPTE MCA language", "SMPTE MCA language & spatial", "none".</td></tr>
<tr><td>Source Package (File / Physical)</td><td colspan="2">There shall be a single essence descriptor for the essence.</td></tr>
<tr><td>Track (Timeline)</td><td colspan="2">The edit rates property of a track for material that is frame coded shall be equal to the rate of content packages in the essence. For progressive material, this is the underlying frame rate. For MPEG interlaced material this is usually the frame rate. For interlaced J2k field coded material and for interlaced MPEG field coded material, this is usually the field rate.
Clip wrapped audio shall have the edit rate set to the audio sampling rate.</td></tr>
</table>

Set name	Constraint
Event Track	Only used for descriptive metadata tracks. Note that metadata on an event track shall have its event duration equal to the validity of the metadata on the timeline. This may often be the duration of the track.
Static Track	Metadata may be present on a static track. This is discouraged. To prevent metadata morphing, static metadata should be placed on an event track with the duration set to the duration of the file. Applications manipulating files should be aware that metadata described as static may no longer be static after splicing and editing operations.
Sequence (all cases)	For a closed file, the durations shall be correct in the header metadata. Open files may use distinguished values as described in 377-1.
SourceClip (Picture, Sound, Data)	Duration values shall use the same rule as for sequences.
MPEG Descriptor	Shall be present for all MPEG video component files, shall include picture essence coding, display width and height, aspect ratio, constant Bframes, MaxGOP, BpictureCount, Bitrate, ProfileAndLevel, VerticalSubsampling.
Wave Audio Essence Descriptor	Shall be present for all uncompressed PCM audio component files.
VBI Data Descriptor	Shall be present for all VBI components.
ANC Data Descriptor	Shall be present for all ANC components.
Multiple Descriptor	Shall be present only for component files that have VBI and ANC data present, otherwise it shall not be used.
Network Locator	Not present in essence component files.
Text Locator	Not present in essence component files.
Timecode	This recommendation constrains the EBU recommendation on timecode in MXF. The material package timecode and the top-level file package timecode with the lowest value of TrackNumber shall be identical. Any captured timecode - which may be discontinuous - shall be captured in the source package that is referenced by the top-level file package.

Shim parameter	Definition
ingest_TC	Default value: **none** Type: extensible enumeration Values: “LTC”, “VITC”, “DVITC”, “external”, “source” or <value> Indicates that ingest (file creation) applications shall start the material package timecode and the top-level file package timecode at a given value, or derive it from the specified source.
lead_TC	Default value: **FP** Type: enumeration Values: MP, FP Indicates that an application requiring the timecode of the component file shall use the lowest numbered by TrackNumber file package timecode track (default) or the material package timecode track of the essence component file.

The header metadata shall always be valid in a component file. This means that the process shown in figure 6 shall be followed:

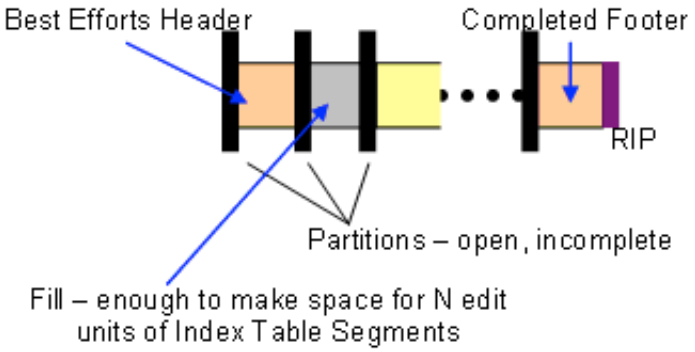

Figure 6: Valid header metadata process

Open and incomplete partitions are created during file writing operation. When the writing operation is finished, the header or footer shall be marked closed and complete. The footer may contain a copy of the header metadata and it shall be marked closed and complete. A random index pack shall follow the footer.

6.3.1 Package labeling

All applications that modify an MXF file shall create a new identification set in the file and shall preserve existing identification sets. The new identification set provides a generation ID that may be used to identify which elements of the file have changed. Full details are given in SMPTE 377-1-2009.

Note: This feature may be used to audit metadata modification in the essence component file. It can therefore allow the detection of differences between "master" metadata and "derived" metadata when an MXF process modifies a file. This information may be used to flag the fact that metadata reconciliation may be required.

7 Legacy bundles

This section specifies AS-02 structures that have been created prior to the approval of this AS-02 specification document. It is recommended that file readers be able to accept files with these structures.

7.1 Legacy bundles

When AS-02 was initially proposed, no root folder was included in the specification. To retain backwards compatibility with the initial rollout and to provide a migration path for original equipment, a *legacy bundle* is defined as a simple bundle with the following additional restrictions:

1. A legacy bundle shall not have a root folder.
2. A legacy bundle shall be identified by the filename of its simple version file.
3. Many legacy bundles may exist in the same folder.
4. The media subfolder shall be named "media.dir".
5. Many legacy bundles may share the same "media.dir" folder.
6. There shall be no extra folder in a legacy bundle.
7. Video essence in a legacy bundle shall be long-GOP MPEG2 wrapped in SMPTE 381M.
8. Audio essence in a legacy bundle shall be uncompressed broadcast wave format (BWF) wrapped in SMPTE 382M.
9. Legacy bundles are not required to have a manifest file.
10. Legacy bundles are not required to have a shim file.

The disadvantage of legacy bundles is that non-MXF operations are very difficult. Most move, copy, FTP and other operation require the parsing of the version file and the potential re-naming of the essence components.

7.2 Legacy shim identification - compliance ID

This section describes the way in which shims were signaled in files created prior to the first publication of the AMWA AS-02 specification. It is included for file readers that need to be backwards compatible with legacy files.

A legacy bundle shall contain one or more DMS identifiers indicating the constrained DMS in use, for example a SMPTE 380M based scheme, or some private MXF scheme.

At least one DMS identifier in the legacy bundle shall be used in order to indicate the compliance ID of the AS-02 file (section 7.2.1).

The descriptive metadata scheme identifiers shall be carried in the preface set of the MXF file according to SMPTE 377M. For example, using a constrained SMPTE 380M scheme would result in the following label in the preface set:

DMS1 06.0b.0e.2b.34. 04.01.01.01. 0d.01.04.01. 01.02.01.01

7.2.1 DMS-AS compliance ID

Each AS-02 shim shall define a compliance ID string. The string should identify the version of the shim to which the file was created.

Note: The compliance ID may change over time within a shim. For example, image resolutions may change, audio arrangements may change and codecs may change.

An application that is comparing ***compliance_id*** strings shall perform a UTF-16 case sensitive comparison that ignores whitespace at the start and end of strings.

Shim parameter	Definition
compliance_id	Default value: **none** Type: UTF-16 string It is recommended that the organization creating the shim uses an XML namespace identifier URI structure as a ***compliance_id***. This minimizes the chances of a ***compliance_id*** clash with other organizations. Publishing the shim document at that URL may also encourage other organizations to copy the shim and hence minimize unnecessary invention in the industry. `e.g. http://amwa.tv/AS-02-shim/2010/hd-j2k.txt`

The ***compliance_id*** shall be stored as FrameworkThesaurusName property of a SMPTE 380M Production Framework.

8 Generic shim

8.1 Shim identifier

To aid system designers and equipment vendors, the empty AS-02 generic shim is tabulated in this section.

A shim identifier (***shim_id***) shall identify the shim. This identifier is intended to signal the version of the shim that was in use when the bundle was created or modified.

Shim parameter	Definition
shim_id	Default value: **none** Type: UTF-16 string It is recommended that the organization creating the shim uses an XML namespace URI as a shim_id. This minimizes the chances of a shim identifier clash with other organizations. Publishing the shim document at that URL may also encourage other organizations to copy the shim and hence minimize unnecessary invention in the industry. e.g. `http://amwa.tv/AS-02-shim/2010/hd-j2k.txt`

The sections below are organized as a sheet to be filled in by a system designer who is constraining their system. An example of a completed shim is given in annex A. Not all the parameters are relevant in every design. Some parameters can be left as none or not applicable (n/a).

The entry in the parameter name column is intended for use as the name of an XML element in a machine-readable template for the shim.

8.2 Shim

Dimension	Description	Parameter name	AS-02 constraint	AS-02 values	Shim-specific value
Shim ID	The identifier of this shim.	***shim_id***	Strong		

8.3 General essence

Dimension	Description	Parameter name	AS-02 constraint	AS-02 values	Shim-specific constraint	Shim-specific values
Partition spacing	Regular partition spacing.	***partition_spacing***	Strong	60s		
Indexing strategy	Frame wrapping strategy.	***index_strategy_frame***	Strong	lead		
Indexing strategy	Clip wrapping strategy.	***Index_strategy_clip***	Strong	lead		

8.4 Picture components

Dimension	Description	Parameter name	AS-02 constraint	AS-02 values	Shim-specific constraint	Shim-specific values
Picture essence schemes	What picture signal schemes (compression, sampling or other) are encountered in programs?	***picture_family***	Gentle	none		
Picture bitrate	How many bits/second at real time?	***picture_bitrate***	Gentle	none		
Picture format	Picture raster and aspect ratio?	***picture_format***	Gentle	none		
Picture component limit	Maximum simultaneous video components?	***picture_component_limit***	Moderate	1		

8.5 Sound components

Dimension	Description	Parameter name	AS-02 constraint	AS-02 values	Shim-specific constraint	Shim-specific values
Sound essence schemes	What sound signal schemes (compression, sampling or other) are encountered in programs?	***audio_family***	None	none		
Audio file arrangement	Physical storage strategy for audio.	***audio_file_arrangement***	None	none		
Audio component limit	Maximum simultaneous audio components?	***audio_component_limit***	Moderate	unlimited		

8.6 Caption components

Dimension	Description	Parameter name	AS-02 constraint	AS-02 values	Shim-specific constraint	Shim-specific values
CC essence schemes	What caption essence types are allowed?	***CC_data_essence***	None	None		
CC custom formatting	Captions to be inserted in video in a custom manner?	***CC_custom***	Moderate	false		
CC in-vision render	Captions inserted into the active picture?	***CC_render***	Moderate	false		
CC component limit	Maximum simultaneous caption components?	***CC_component_limit***	Moderate	unlimited		

8.7 Other VANC / VBI components

Dimension	Description	Parameter name	AS-02 constraint	AS-02 values	Shim-specific constraint	Shim-specific values
VBI essence schemes	What VBI essence types are allowed?	*VBI_data_essence*	None	None		
VBI custom formatting	VBI data inserted in video in a custom manner.	*VBI_custom*	Moderate	false		
VBI in-vision rendering	VBI data to be rendered into the active picture.	*VBI_render*	Moderate	false		
ANC essence schemes	What VBI essence types are allowed?	*ANC_data_essence*	None	None		
ANC custom formatting	Captions to be inserted in video in a custom manner.	*ANC_custom*	Moderate	false		
ANC in-vision rendering	VBI data to be rendered into the active picture.	*ANC_render*	Moderate	false		
Data component limit	Maximum simultaneous data components?	*data_component_limit*	Moderate	unlimited		
Separate data component files	Different types of VBI/VANC in separate files?	*data_separation*	Moderate	true		

8.8 Version files

Dimension	Description	Parameter name	AS-02 constraint	AS-02 values	Shim-specific constraint	Shim-specific values
Version file complexity	Simple version files only?	*simple_versions_only*	Gentle	false		
Version file bloating	Can generic streams be put in a version file?	*generic_streams_in_version*	Strong	false		
Short clip limit	Shortest length of a clip in an EDL.	*sub_clip_limit*	Gentle	1s		
Version file complexity	Is an edl a series of synchronous butt edits?	*sync_cut*	Gentle	false		
File length	Minimum duration of a simple version.	*min_sv_dur*	Gentle	10s		
Media integrity scope	What is the scope of the media integrity check?	*mic_scope*	Moderate	essence_only		
Media integrity type	What algorithm is used for MIC?	*mic_algorithm*	Moderate	crc32		

Dimension	Description	Parameter name	AS-02 constraint	AS-02 values	Shim-specific constraint	Shim-specific values
Channel ID	Is channel mapping done with track numbers?	***track_numbers***	Gentle	false		
Timecode	Timecode counting mode.	***tc_mode***	Strong	<shim specific>		

8.9 Header metadata

Dimension	Description	Parameter name	AS-02 constraint	AS-02 values	Shim-specific constraint	Shim-specific values
System item timecode handling	Is the system item timecode copied to the header?	***sys_item_tc***	Moderate	false		
Track tag metadata	How are tracks tagged?	***track_tag_policy***	None	None		
Ingest timecode handling	What timecode source shall be dominant when creating files?	***ingest_TC***	Strong	none		
Timecode precedence	What timecode source shall be dominant when using essence files?	***lead_TC***	Strong	FP		

8.10 Bundle

Dimension	Description	Parameter name	AS-02 constraint	AS-02 values	Shim-specific constraint	Shim-specific values
Maximum versions	Maximum number of versions.	***max_versions***	None	None		

Note: Setting this property to 1 and the *simple_version* property to true is used to indicate that only simple bundles are acceptable.

8.11 Descriptive metadata

Note: The shim identifier property "*compliance_id*" is included for legacy files only, as described in section 7.2.1.

Dimension	Description	Parameter name	AS-02 constraint	AS-02 values	Shim-specific constraint	Shim-specific values
Shim identifier	The legacy compliance ID of this file.	***compliance_id***	None	None	Fully	

9 Manifest file format

Each AS-02 bundle shall have a single manifest file, located in the root level folder of the bundle. The manifest file contains a list of all of the files and folders in the bundle, including the version and essence component files.

The manifest shall be encoded as an XML document (W3C XML 1.0). The manifest file shall be named `manifest.xml`.

9.1 Manifest structure

The top-level element in the manifest file shall be designated `Manifest`, and is illustrated in figure 7. See the XML schema declaration in section 9.3 of this document for the formal element definition.

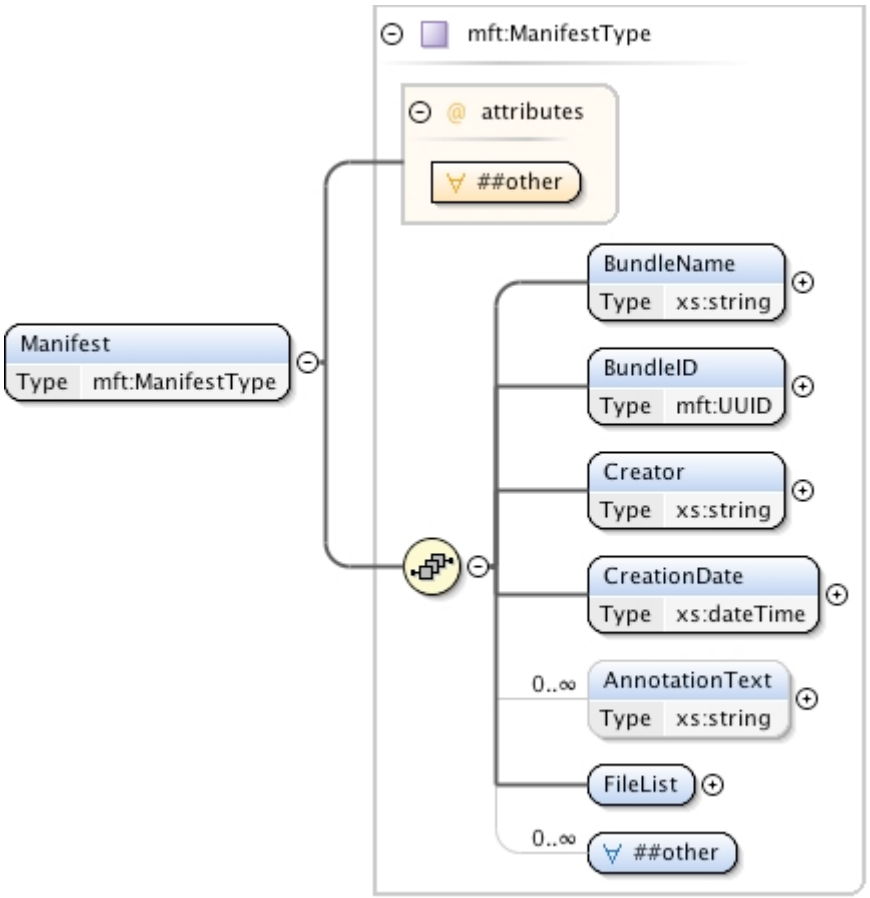

Figure 7: Manifest structure element

Note: In figure 7, elements shown with a grey outline are optional.

9.1.1 Bundle name element

The bundle name (`BundleName`) element uniquely contains the file name of the root-level folder of the bundle.

9.1.2 Bundle identifier element

For overall management of the asset, the bundle identifier (`BundleID`) element shall uniquely identify the bundle. Each unique bundle shall have a distinct bundle identifier. The bundle identifier shall be encoded as a URN-encoded UUID, as defined in IETF RFC 4122.

Note: This will allow easy differentiation between bundles.

9.1.3 Creator element

The creator (`Creator`) element shall be a free-form, human-readable annotation describing the person, facility or system (hardware/software) that created the bundle.

Note: The creator property is intended only for display as guidance to a user.

9.1.4 Creation date element

The creation date (`CreationDate`) element shall be set to the time and date at which the bundle was created. The creation date shall be encoded as `xs:dateTime` type.

9.1.5 Annotation text element (optional)

Annotation text (`AnnotationText`) elements may be present and shall be a list of zero or more free-form, human-readable annotations describing the bundle.

Note: Annotation text elements are intended only for display as guidance to a user.

9.1.6 File list element

The file list (`FileList`) element shall contain the list of `File` elements that each describe the files and folders contained in the bundle. The structure of the `File` element is described in section 9.2. The order of `File` elements in the list shall not be significant.

9.2 File element

A manifest shall contain a list of files and folders. A file (`File`) element shall represent any file or folder that exists in the AS-02 bundle. Each file shall be described by a file element, as illustrated in figure 8. See the XML schema declaration in section 9.3 of this document for a normative definition.

The manifest shall not include a file element entry for the manifest itself.

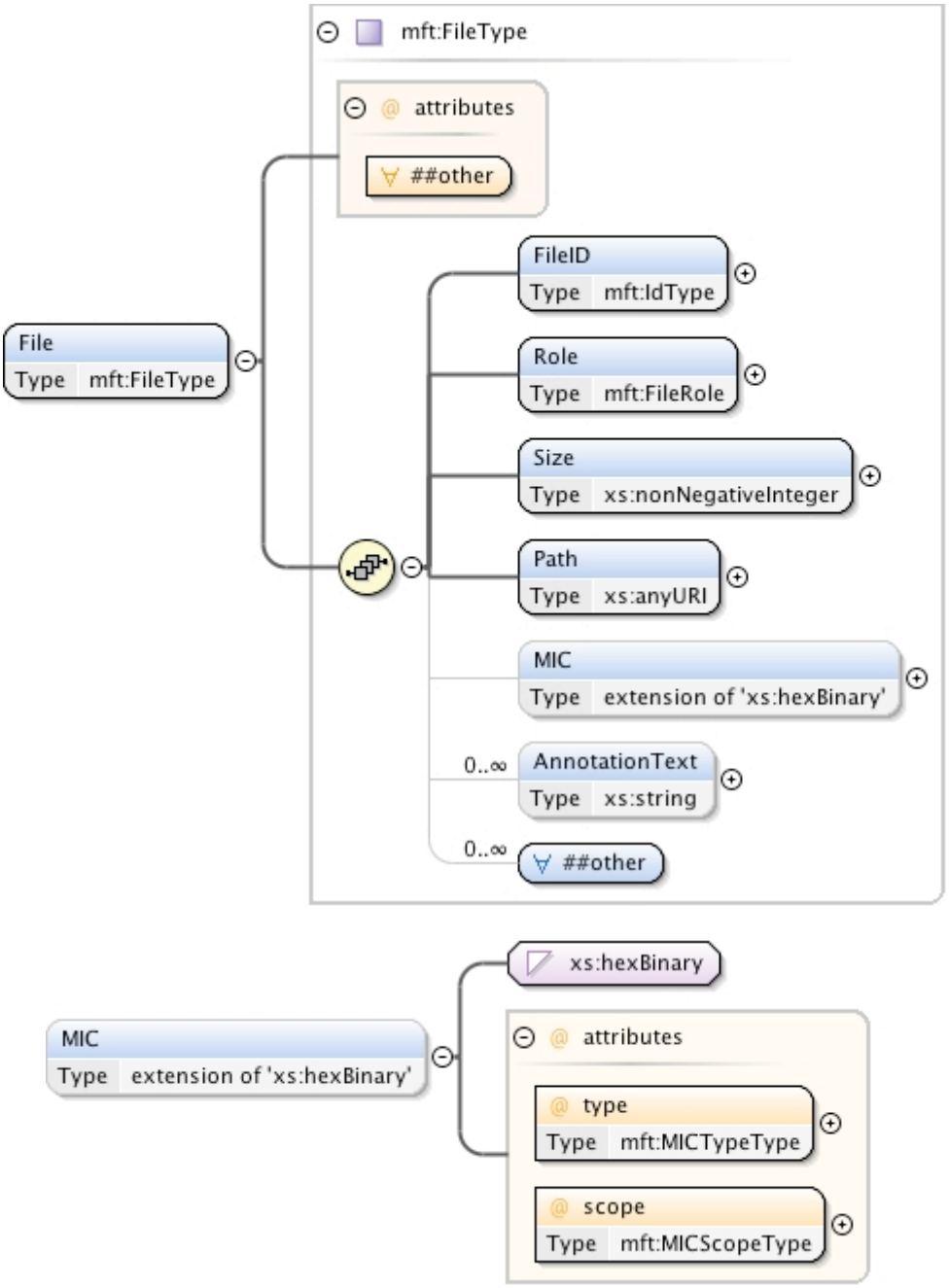

Figure 8 – File element with optional media integrity check

Note: In figure 8, elements shown with a grey outline are optional.

9.2.1 File identifier element

The file identifier (`FileID`) element shall represent the unique identifier associated with the described file. Where the file has its own unique identifier, such as the primary package identifier of a version file or essence component file, this value shall be extracted from the file and used in the manifest. If the file contains no defined identifier, then the creator of the manifest file shall generate the identifier.

The value of the identifier shall be encoded according to the following rules, applied in order:

1. Where the native identifier of the file is a SMPTE UMID (SMPTE 330M) starting with the fixed 16-bytes `060a2b34`h `01010105`h `01010f20`h `13000000`h, the last 16-bytes of the UMID shall be used as a UUID encoded as a URN according to IETF RFC 4122.
2. Where the native identifier of the file is a SMPTE UMID (SMPTE 330M), the file identifier shall be encoded as a URN representation of the UMID according to SMPTE 2029-2009.
3. Where the native identifier of the file is a SMPTE universal label, the file identifier shall be encoded as a URN representation of the universal label according to SMPTE 2029-2009.
4. All other files shall have a UUID encoded as a URN according to IETF RFC 4122.

9.2.2 Role element

The role (`Role`) element shall be used to describe how the file is used within the bundle. The value of the element shall be used both for display as guidance for the user and as machine-interpretable information for content processing. The allowed values for the role element shall be as specified in Annex B.

9.2.3 Size element

The size (`Size`) element contains the size of the file in bytes. This size shall be expressed as an integer number of bytes, encoded as type `xs:positiveInteger`.

For folders, the size value shall be present and set to zero.

9.2.4 Path element

The path (`Path`) element contains the relative URI of the file, relative to the root of the bundle.

9.2.5 Media integrity check element (optional)

The media integrity check (`MIC`) element contains the media integrity check value for the file.

9.2.5.1 Element value

The value of the media integrity check element (`MIC`) shall be of type `xs:hexBinary`.

9.2.5.2 Type attribute

The media integrity check type (`type`) attribute shall be present and shall indicate the algorithm used to create the media integrity check value for the AS-02 asset. The possible values of this parameter shall be:

- `hmac-sha1` - secure hash algorithm, as defined in IETF RFC 2104 and IETF RFC 3174;
- `crc32` - 33-bits polynomial length cyclic redundancy check, computed according to CRC-32C (Castagnoli) polynomial;
- `crc16` - 17-bits polynomial length cyclic redundancy check, computed according to the CRC-16-CCITT polynomial;
- `md5` - message digest algorithm 5, as specified in IETF RFC 1321.

Note: For more information on calculating CRCs, the related Wikipedia page is a good place to start. See: http://en.wikipedia.org/wiki/Cyclic_Redundancy_Check.

Note: The choice of CRC types is based on ones in common use and easy to decode. The list is left deliberately short to minimize complexity when implementing a decoder. The intention is to replace this section with a reference to AMWA's application specification for content integrity (AS-06) once work is completed. To ensure full coverage of the range of content integrity types, implementers are advised to check the current status of this document and AS-06.

9.2.5.3 Scope attribute

The media integrity check scope (`scope`) attribute shall be present and shall indicate the scope over which media integrity check values are calculated. The possible values of this parameter shall be:

- `essence_only` - the media integrity check was calculated for the wrapped essence stream of an essence component file;
- `entire_file` - the media integrity check value was calculated for the entire file, including essence and any wrapper.

9.2.6 Annotation text element (optional)

Annotation text (`AnnotationText`) elements may be present and shall be a list of zero or more free-form, human-readable annotations describing the file.

Note: Annotation text elements are intended only for display as guidance to a user.

9.3 XML schema for manifests

The XML Schema document presented in this section normatively defines the structure of a manifest using a machine-readable language.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:mft=”http://www.amwa.tv/as-02/1.0/manifest”
    targetNamespace="http://www.amwa.tv/as-02/1.0/manifest"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">

  <!-- ManifestType -->
  <xs:complexType name="ManifestType">
    <xs:sequence>
      <xs:element name="BundleName" type="xs:string"/>
      <xs:element name="BundleID" type="mft:UUID"/>
      <xs:element name="Creator" type="xs:string"/>
      <xs:element name="CreationDate" type="xs:dateTime"/>
      <xs:element name="AnnotationText" type="xs:string"
          minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="FileList">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="File" type="mft:FileType" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:any minOccurs="0" maxOccurs="unbounded" namespace="##other"
          processContents="lax"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <!-- FileType -->
  <xs:complexType name="FileType">
    <xs:sequence>
      <xs:element name="FileID" type="mft:IdType"/>
      <xs:element name="Role" type="mft:FileRole"/>
      <xs:element name="Size" type="xs:nonNegativeInteger"/>
      <xs:element name="Path" type="xs:anyURI"/>
      <xs:element name="MIC" minOccurs="0">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:hexBinary">
              <xs:attribute name="type" type="mft:MICTypeType" use="required"/>
              <xs:attribute name="scope" type="mft:MICScopeType" use="required"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
      <xs:element name="AnnotationText" type="xs:string"
          minOccurs="0" maxOccurs="unbounded"/>
      <xs:any minOccurs="0" maxOccurs="unbounded" namespace="##other"
          processContents="lax"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
```

```
  <!-- IdType -->
  <xs:simpleType name="IdType">
    <xs:union memberTypes="mft:UUID mft:UMID mft:UL mft:emptyID"/>
  </xs:simpleType>
  <!-- UUID -->
  <xs:simpleType name="UUID">
    <xs:restriction base="xs:string">
    <xs:pattern value="urn:uuid:[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-
F]{4}-[0-9a-fA-F]{12}"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- UMID -->
  <xs:simpleType name="UMID">
    <xs:restriction base="xs:string">
      <xs:pattern value="urn:smpte:umid:([0-9a-fA-F]{8}\.){7}[0-9a-fA-F]{8}"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- UL -->
  <xs:simpleType name="UL">
    <xs:restriction base="xs:string">
      <xs:pattern value="urn:smpte:ul:([0-9a-fA-F]{8}\.){3}[0-9a-fA-F]{8}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="emptyID">
    <xs:restriction base="xs:string">
      <xs:pattern value=""/>
    </xs:restriction>
  </xs:simpleType>

  <!-- MICTypeType -->
  <xs:simpleType name="MICTypeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="hmac-sha1"/>
      <xs:enumeration value="crc32"/>
      <xs:enumeration value="crc16"/>
      <xs:enumeration value="md5"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- MICScopeType -->
  <xs:simpleType name="MICScopeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="essence_only"/>
      <xs:enumeration value="entire_file"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- FileRole -->
  <xs:simpleType name="FileRole">
    <xs:restriction base="xs:string">
      <xs:enumeration value="file"/>
      <xs:enumeration value="version"/>
      <xs:enumeration value="essencecomponent"/>
      <xs:enumeration value="shim"/>
      <xs:enumeration value="manifest"/>
      <xs:enumeration value="qc"/>
      <xs:enumeration value="graphic"/>
      <xs:enumeration value="caption"/>
      <xs:enumeration value="folder"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:element name="Manifest" type="mft:ManifestType"/>
</xs:schema>
```

Note: The schema is available from the AMWA FTP site as file `AS-02/AS-02-10-2011-11-17_Manifest.xsd`.

9.4 Sample manifest file (informative)

The following manifest sample XML is a valid instance of the manifest schema. It is non-functional and intended for informative purposes only.

```
<?xml version="1.0" encoding="UTF-8"?>
<mft:Manifest xmlns:mft="http://www.amwa.tv/as-02/1.0/manifest"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <mft:BundleName>AmberFin_AS02_MP2_720i25_4</mft:BundleName>
  <mft:BundleID>urn:uuid:D08C1211-5172-4875-96D5-1B731E1D71CD</mft:BundleID>
  <mft:Creator>AS-02 Reference Manifest File MP2 - Manual Generation - Richard Cartwright -
AMWA</mft:Creator>
  <mft:CreationDate>2011-12-06T10:53:22-00:00</mft:CreationDate>
  <mft:FileList>
    <mft:File>
      <mft:FileID>
urn:smpte:umid:060a2b34.01010105.01010d11.13000000.f7693d57.31540580.905f0022.5f0bc2f2
      </mft:FileID>
      <mft:Role>version</mft:Role>
      <mft:Size>16153</mft:Size>
      <mft:Path>AmberFin_AS02_MP2_720i25_4.mxf</mft:Path>
      <mft:MIC type="md5" scope="entire_file">fc8578b82d4eb729be0212633f81318f</mft:MIC>
      <mft:AnnotationText>Single version file of this AS-02 bundle.</mft:AnnotationText>
    </mft:File>
    <mft:FileID>urn:uuid:D67509D3-5579-4388-8148-9FF1E08BBB59</mft:FileID>
      <mft:Role>shim</mft:Role>
      <mft:Size>411</mft:Size>
      <mft:Path>shim.xml</mft:Path>
      <mft:MIC type="md5" scope="entire_file">b587ea43b80900fcf5112aaafcc31ab8</mft:MIC>
    </mft:File>
    <mft:File>
      <mft:FileID>urn:uuid:ACC8E4E7-5D37-4CE6-889D-E88C3DA4023C</mft:FileID>
      <mft:Role>folder</mft:Role>
      <mft:Size>0</mft:Size>
      <mft:Path>extra</mft:Path>
      <mft:AnnotationText>Size and MIC are not encoded for folders.</mft:AnnotationText>
    </mft:File>
    <mft:File>
      <mft:FileID>urn:uuid:03EF7F92-8339-4A2A-BB7D-562A5A486ADF</mft:FileID>
      <mft:Role>folder</mft:Role>
      <mft:Size>0</mft:Size>
      <mft:Path>media</mft:Path>
    </mft:File>
    <mft:File>
      <mft:FileID>
urn:smpte:umid:060a2b34.01010105.01010511.13000000.bbaffc56.31540580.23480022.5f0bc2f2
      </mft:FileID>
      <mft:Role>essencecomponent</mft:Role>
      <mft:Size>62517511</mft:Size>
      <mft:Path>media/AmberFin_AS02_MP2_720i25_4_v0.mxf</mft:Path>
      <mft:MIC type="md5" scope="entire_file">9805614535f364690491a5ac16e8c620</mft:MIC>
    </mft:File>
    <mft:File>
      <mft:FileID>
urn:smpte:umid:060a2b34.01010105.01010811.13000000.d240fc56.31540580.23480022.5f0bc2f2
      </mft:FileID>
      <mft:Role>essencecomponent</mft:Role>
      <mft:Size>1932121</mft:Size>
      <mft:Path>media/AmberFin_AS02_MP2_720i25_4_a1.mxf</mft:Path>
      <mft:MIC type="md5" scope="entire_file">a2b275b19bfa4bfebcc422f604e25b8d</mft:MIC>
    </mft:File>
    <mft:File>
      <mft:FileID>
urn:smpte:umid:060a2b34.01010105.01010811.13000000.2e41fc56.31540580.23480022.5f0bc2f2
      </mft:FileID>
      <mft:Role>essencecomponent</mft:Role>
      <mft:Size>1932121</mft:Size>
      <mft:Path>media/AmberFin_AS02_MP2_720i25_4_a2.mxf</mft:Path>
      <mft:MIC type="md5" scope="entire_file">1cca0628ed8d1d7aba35858a567a3d2c</mft:MIC>
    </mft:File>
  </mft:FileList>
</mft:Manifest>
```

10 Shim file format

Each AS-02 bundle shall have a single shim file, located in the root level folder of the bundle. The shim file contains an identifier for the constraints of the shim in the form of a URI namespace reference to the shim that is being used. The shim file shall be encoded as an XML document [XML 1.0]. The shim file shall be named `shim.xml`.

Note: Implementers need to be aware that the shim schema will be actively further developed within AMWA to be a machine-readable list of shim constraints.

10.1 Shim structure

The top-level element in the shim file shall be designated `Shim`, as shown in figure 9. See the XML schema declaration in Section 10.4 of this document for explicit type information.

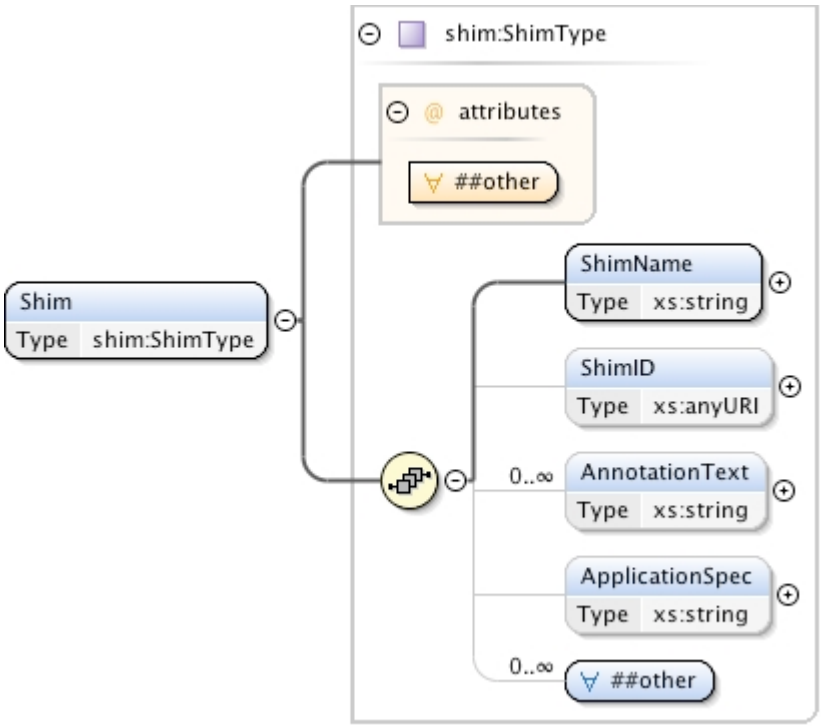

Figure 9: Top-level shim structure

Note: In figure 9, elements with grey outline are optional.

10.1.1 Shim name element

A name for the AS-02 shim according to a controlled vocabullary, e.g. Network HD Contribution. The value of the element shall be of type `xs:string`.

10.1.2 Shim identifier element

The shim identifier (`ShimID`) element contains the string identifier of this shim specification (section 8). The value of the element shall be of type `xs:anyURI`.

Note: This document does not specify the process by which a shim identified is generated.

10.1.3 Annotation text element (optional)

Annotation text (`AnnotationText`) elements may be present and shall be a list of zero or more free-form, human-readable annotations describing the shim.

Note: Annotation text elements are intended only for display as guidance to a user.

10.1.4 Application specification element (optional)

Application specification element (`ApplicationSpec`) is reserved for future use. The value of the element shall be of type `xs:string`.

10.2 XML schema for shims

The XML schema document presented in this section normatively defines the structure of a shim file using a machine-readable language.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:shim="http://www.amwa.tv/as-02/1.0/shim"
    targetNamespace="http://www.amwa.tv/as-02/1.0/shim"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">

<!-- ShimType -->
  <xs:complexType name="ShimType">
    <xs:sequence>
      <xs:element name="ShimName" type="xs:string"/>
      <xs:element name="ShimID" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="AnnotationText" type="xs:string"
          minOccurs="0"  maxOccurs="unbounded"/>
      <xs:element name="ApplicationSpec" type="xs:string" minOccurs="0"/>
      <xs:any minOccurs="0" maxOccurs="unbounded" namespace="##other"
          processContents="lax"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <xs:element name="Shim" type="shim:ShimType"/>
</xs:schema>
```

Note: The schema is available from AMWA as FTP site as file `AS-02/AS-02-10-2011-11-17_Shim.xsd`.

10.3 Sample XML shim file (informative)

The following sample shim XML is a valid instance of the shim schema. It is non-functional and intended for informative purposes only.

```
<?xml version="1.0" encoding="UTF-8"?>
<Shim             xmlns="http://www.amwa.tv/as-02/1.0/shim"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <ShimName>Network HD Contribution</ShimName>
    <ShimID>http://amwa.tv/AS-02-shim/2010/hd-j2k.txt</ShimID>
    <AnnotationText>An example shim for HD J2k created by AMWA</AnnotationText>
</Shim>
```

Annex A. AS-02 sample shim document (informative)

A.1 Essence component files

The essence component files contain the video, audio and ancillary data essences that make up the asset. The table below specifies the constraints on the essence and its wrapping as MXF.

<table>
<tr><th>#</th><th>Parameter</th><th>Type</th><th>Value</th></tr>
<tr><td>1</td><td>mono_essence_op1a</td><td>boolean</td><td>True. All essence components are stored as mono-essence OPa1a files</td></tr>
<tr><td>2</td><td>VBI_custom and ANC_custom</td><td>boolean</td><td>False. All ancillary data, both VBI and ANC in this case, is stored as separate SMPTE 436M streams in essence component files. No other proprietary ANC formats are maintained.</td></tr>
<tr><td>3</td><td>VBI_render and ANC_render</td><td>boolean</td><td>False. No ANC or VBI information is rendered into the active picture</td></tr>
<tr><td>4</td><td>picture_family</td><td>list</td><td>The following list of video essence types is permitted. Stored in media*_v0.mxf
<table>
<tr><th>MXF</th><th>Codec</th><th>Constraints</th><th>Resolution</th></tr>
<tr><td>SMPTE 422M
Mapping JPEG 2000 codestreams into the MXF generic container</td><td>JPEG 2000</td><td>YUV 422 10bit, 100Mbps
CDCI picture essence descriptor fields:
Essence container UL - 06.0E.2B.34.04.01.01.01.0D.01.03.01.02.0C.01.00
JPEG 2000 picture sub-descriptor field values:
Rsiz – 0
Xsiz – 1920
Ysiz – 1080
XOsiz – 0
YOsiz – 0
XTsiz – 1920
YTsiz – 1080
XTOsiz – 0
YTOsiz – 0
Csiz – 3
Picture component sizing – {Number of component = 3, 3, Ssiz0 = 9, XRsiz0 = 1, YRsiz0 = 1, Ssiz1 = 9, XRsiz1 = 2, YRsiz1 = 1, Ssiz2 = 9, XRsiz2 = 2, YRsiz2 = 1}</td><td>1920x1080 23.98p</td></tr>
</table></td></tr>
<tr><td>5</td><td>audio_family</td><td></td><td>The following list of audio essence types is permitted. Stored in media*_a??.mxf.
<table>
<tr><th>MXF</th><th>Codec</th><th>Constraints</th><th>Sampling / structure</th></tr>
<tr><td>SMPTE 382M
Mapping AES3 and Broadcast Wave Audio into the MXF Generic Container</td><td>Uncompressed BWF</td><td>Stored as Stereo pairs.</td><td>24-bit, 48kHz or 96kHz</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>#</th><th>Parameter</th><th>Type</th><th>Value</th></tr>
<tr><td>6</td><td>CC_data_essence</td><td></td><td>The following list of data essence types is permitted. Stored in media*_vanc??.mxf
<table>
<tr><th>MXF</th><th>Contents</th><th>Coding</th><th>Processing</th></tr>
<tr><td>SMPTE 436M
MXF mappings for VBI lines and ancillary data packets</td><td>SMPTE 291M
VANC data</td><td></td><td>Packets used to create .scc files in "extra\captions" folder.</td></tr>
</table></td></tr>
<tr><td>7</td><td>partition_spacing</td><td>time</td><td>10s between partitions.</td></tr>
<tr><td>8</td><td>index_strategy_frame and index_strategy_clip</td><td>enum</td><td>Lagging. Index tables are in the partition after the essence.</td></tr>
<tr><td>9</td><td>track_tag_policy</td><td>specific-ation</td><td>All MP tracks are tagged with their nominal track position. Video tracks are labeled “1”, audio channels are labeled from “1”. In a stereo pair, the track number applies to the left channel, the right channel shall take the value of the left channel plus one.</td></tr>
<tr><td>10</td><td>ingest_TC</td><td>enum</td><td>00:57:30;00 is the expected start timecode of the first frame of any complete asset. This depends on the asset being ingested and is not fixed by this specification.</td></tr>
<tr><td>11</td><td>lead_TC</td><td>enum</td><td>MP::TC. The lead timecode shall be taken from the timecode component of the material package of version file. All other MXF timecodes shall be considered to be annotations.</td></tr>
<tr><td>12</td><td>shim_id</td><td>UL list</td><td>http://www.amwa.tv/as-02/hd/v0/2010</td></tr>
</table>

A.2 Simple version files

A simple version file (SPV) links together the essence components to create a complete asset. The simple version files do not describe any edits to the components.

#	Parameter	Type	Value
1	***generic_streams_in_version***	Boolean	**False**. No generic streams are permitted in a simple version file.
2	***min_sv_dur***	Time	**1 sec**. The minimum duration of an AS-02 file compliant with this shim is 1 second.
3	***mic_algorithm***	Enum	**N/A** Media integrity check is not used.
4	***mic_scope***	Enum	**N/A** Media integrity check is not used.
5	***track_numbers***	Boolean	**True**. The track numbers will be written into the material package of the version file.

A.3 Version files

Some facilities only use simple version files. This should be indicated here.

#	Parameter	Type	Value
1	***sub_clip_limit***	Time	**1 sec.** The minimum size of a clip built into an edit list is 1 second.
2	***sync_cut***	boolean	**True.** All of the tracks in present in the version file must be cut synchronously, constraining the version files to be butt-edits of pre-conformed clips.

A.4 Metadata in MXF-AS-02

Here we detail any special metadata to be included in the file or any constraints on the metadata used in the workflow. This section intentionally blank in this case.

#	Parameter	Type	Value

A.5 Bundle

This section intentionally blank in this case.

#	Parameter	Type	Value

Annex B. Registered manifest file roles (normative)

This annex contains a registered list of different roles that may be assigned to a file in the manifest. It shall be the responsibility of an application that creates a new file in the bundle to assign the correct role. It shall be the responsibility of applications that modify or transcode bundles to persist those roles in the new manifest.

Note: The purpose of roles is for an AS-02 aware application to be able to, for example: quickly locate the correct QC file in a workflow; to clear out all the graphics files prior to distribution.

Role	Contents	Description
file	Just a file	Default file role.
essencecomponent	Essence component	All .mxf files in the media sub folder unless overridden by some other role.
version	Version file	All .mxf file in the root folder of the bundle that represent a version of the asset.
manifest	Manifest file	Manifest file in the root folder. Note: As specified in section 9,2, the manifest file shall not be listed as a file within a manifest. This role type of manifest is included for use in future or external applications.
shim	Shim	Shim file in the root folder.
graphic	Graphics files	Graphics files for bugs, logos, overlay etc. Note that .mxf files in the media subfolder may be labeled as graphic if the intention is that it shall be overlayed by some process onto the main video.
qc	QC reports	A quality control report. Dependent data files should be labeled as file.
caption	Caption source files	.cap, .stl, .dfxp, smill, sami and other files used to source captions should be labeled as caption.
folder	Folders	All subfolders (e.g. media, extra extra/amwa.tv) shall have an entry in the manifest. This helps implementers persist intentionally empty folders.