Annex D – Wrappers and Metadata

D1 – Platform Neutrality

Platform Neutrality normally refers only to the byte-ordering of multi-byte data items in Files. It also refers to the limitations of file lengths through the storage filing system (a common limit is 4Gbyte). A format is Platform-Neutral if the complexity of implementation is roughly equal on any platform. For the business of the Task Force, Platform Neutrality must have a wider scope: it needs to cover efficiency of encoding and decoding simultaneously on different platforms.

There are several topics discussed here:

- bit ordering
- transmission of multi-byte data
- Endian-ness flag issues
- 10-bit data and C_b Y C_r [Y] sample structure

D1.1 Bit ordering

All computers and many other items of broadcast equipment use a minimum symbol size of 8 bits. Bit ordering within each byte therefore presents no problems for media interchange. However, if symbols are serialized for transmission, the bit order must be defined to ensure that the decoded bytes match the bit order from the transmitter. Both 'MSB first' and 'LSB first' are used for different inter-connects but provided a given order is defined by the transmission protocol, byte reconstruction at the receiver will be consistent with the transmitted bytes. There is thus no requirement for a "bit-Endian-ness flag" nor the accompanying complexity.

D1.2 Multi-byte data

Suppose a 32-bit word, 0x76543210 is contained in a Wrapper. If it was written on a Little-Endian (Intel) machine, it will be stored in the sequence: 10 32 54 76. If it was written on a Big-Endian (Motorola and others) machine, it will be stored in the sequence: 76 54 32 10. Clearly this presents problems for interchange between the two Endian forms whether by media transfer or transmission.

Words which are part of Overhead (i.e., not Essence or Metadata) would be expected to always be one way (Little- or Big-Endian) in order to avoid the definition of decoders which first need to refer to an Endian-ness flag before deciding how to decode.

In the case of words which are part of the Essence or Metadata: if the data is Presented or Transported as a File (i.e. it is going from A to B as a unit), and is transported transparently, the decoder will accurately reconstruct the original File contents. This applies whether the File is being transmitted or transferred by media exchange.

However, if the data is being Presented or Transported as a Stream in which interpretation of the data is required (e.g. for a downstream MPEG decoder), then the Byte order of file transfer becomes very important.

In some cases, we are assisted because the data format defines the answer. For example, MPEG bit stream syntax is one-bit serial, and elementary streams are mapped onto a byte stream, MSB first; the byte stream is Endian-free (a byte string is always stored the same way on both Intel and Motorola processors). Also, any Metadata which is character string based is Endian-free.

It would be desirable to have Essence data which is Endian-free but allow encoded Metadata (for example, position measured as a 16 bit number of pixels) to be either way.

D1.3 Endian-ness flag issues

Byte order is simply handled by having an Endian-ness flag (Little- or Big- Endian). Then all data types: short, long and long-long use the Endian-ness flag to change the byte order if necessary.

It is not possible to enforce a particular byte order in all cases, so a byte order flag is a requirement. If a Wrapper contains data with the wrong byte order value, and the platform cannot operate with that byte order, then the data has to be converted. This is necessary and time-consuming, but unavoidable.

For operational simplicity, it would be desirable to have all the data within a wrapper be of the same byte order to ensure the most efficient playout.

For applications which merge data of different byte order, there are two possibilities:

- convert the data at the time it is copied, or use a reference between two Wrappers, and
- allow a change of byte order from one Wrapper to the other.

The byte order flag within a Wrapper can occur in two places:

- as early as possible in the Wrapper Overhead
- as part of the Composition Metadata referring to another wrapper.

Note that only one of these two is required; and if both are present, they must be the same. From one Wrapper to the next, there would be two opportunities to signal the Endian-

ness: once in the reference pointing out of the parent Wrapper, and once in the Overhead of the Wrapper being referred to.

D1.4 10-bit data and ITU-R BT.601 C_b Y C_r [Y] sample structure

The bit packing of Essence data must be designed with a view to efficient decoding on all platforms since no perfect Endian-neutral scheme exists. Complex data packing such as required to accommodate, for example, a 10 bit per sample RGB sampling structure occupying 30 bits per Word, must be organized to avoid placing the optimal pattern in favor of one value of Endian-ness.

This is discussed further in D4 – Notes.

D2 – Wrapper Referencing

D2.1 Wrapper Varieties

Program material will involve at least six different varieties of Wrapper:

A. Unwrapped Content - for example, signals from today's equipment, or from foreign, non-conforming systems.

B. Wrappers whose Content is predominantly Essence, but which may include some Metadata.

C. Wrappers whose Content includes no Essence, and only includes Metadata. If Essence is involved it will be kept elsewhere (wrapped or unwrapped), and these Wrappers will include references to it.

D. Wrappers which include predominantly Composition Metadata, and presumably therefore include many references to Content of type A or B kept elsewhere.

E. Wrappers which include Composition Metadata and Essence.

F. Wrappers which include Composition Metadata and additional Descriptive Metadata, which in turn refers to Content kept elsewhere.

Type C is particularly intriguing. The Metadata forms an Index or Directory of Content. This is one variety of Association Metadata.

A simple example from current practice is to refer to a segment of video tape from within an EDL. This is achieved today using "reel numbers" and two timecodes, the "source in-point" and "source out-point".

(Note: SMPTE 258M calls these the SOURCE_IDENTIFIER, SOURCE_ENTRY and SOURCE_EXIT fields).

A reference is one variety of Composition Metadata. A reference might point to the following:

- an entire Wrapper
- a point within some Content (e.g. a single frame, or instant)
- a specific item of Metadata
- a region within some Content (e.g. a "sub-clip" of a shot)

D2.2 Referencing Content

There are at least five ways that Content may be referenced from within a Wrapper (typically from within Composition Metadata):

1. Content is contained within the same file.

2. Content is referenced in an external unwrapped file (e.g. media in some "raw" or "native" format). Note that in this case there is no way to guarantee that this Content is the correct material, except that data format, length, and perhaps name if available in that external format can be checked against the Metadata description of the Content.

3. Content is referenced in an external Wrapper with the same UID as the original reference. (There may be indirection here, where a referenced Wrapper does not contain Content but references yet another wrapper that does.)

4. Content is referenced in an external wrapper, but has been replaced by new Content from the same original source by an application that retains the original Metadata (perhaps recreated at a different resolution, or perhaps because it was deleted to conserve storage and then recreated when needed again).

   It would be desirable to have two levels of UID – a "handle" and one for the actual Content. All Content references would be to the handle's UID. Then a Content item could be replaced with a new UID, and its handle would have its reference to the Content updated. All external references to the handle would remain valid.

5. Content is recreated in a separate environment from the same original source. In this case there is no way to have UID references. An application could examine the original source information (e.g. tape and timecode range) and determine that this Content is equivalent to the original Content, and update the reference to the new UID.

D2.3 Reference Types

The cases of references are therefore as follows:

- 1a. from a "user" Wrapper to the whole of a "source" Wrapper
- 1b. from a "user" Wrapper to a point within a "source" Wrapper
- 1c. from a "user" Wrapper to an item within a "source" Wrapper
- 1d. from a "user" Wrapper to a segment of a "source" Wrapper

example: the "user" Wrapper is Composition Metadata, the "source" Wrapper is Essence

- 2a. from a Wrapper to another point within the same Wrapper
- 2b. from a Wrapper to another item within the same Wrapper
- 2c. from a Wrapper to a segment elsewhere in the same Wrapper

example: the user Wrapper is a conglomeration of Composition Metadata and Essence

- 3a. from a Wrapper to the whole of an external file, tape or other storage means
- 3b. from a Wrapper to a point within an external file, etc.
- 3c. from a Wrapper to an item within an external file, etc.
- 3d. from a Wrapper to a segment of an external file, etc.

example: the user Wrapper is Composition Metadata, the external tape is a timecoded videotape

Note that in this example, the external file has a "natural" labeling method (in this case, timecode); so, even though it doesn't have a header or anything to say exactly what range of timecodes is on the tape, the Wrapper can point directly to the portion required.

A more general case would employ a directory (or index) between the user and the source, to locate the actual Content.

- 4a. from a Wrapper via a directory to the whole of an external file, etc.
- 4b. from a Wrapper via a directory to a point with in an external file, etc.
- 4c. from a Wrapper via a directory to a segment of an external file, etc.

example: the user Wrapper is Composition Metadata, the directory contains Association Metadata to translate time-code into byte address (e.g. "12:23:34.07 is at offset 0x157A3C in the file"), the external file is a data file on a disk array.

We should also consider:

- 5a. from a Wrapper via a directory to the whole of a source Wrapper
- 5b. from a Wrapper via a directory to a point with in a source Wrapper
- 5c. from a Wrapper via a directory to an item with in a source Wrapper
- 5d. from a Wrapper via a directory to a segment of a source Wrapper

D3 – Access Control and Copyright

D3.1 Access Control

Operational security should be implemented to prevent unauthorized operational access to the data. There are a number of access control methods available which address the users requirements from which the most appropriate one can be selected. The access control method may involve randomization or encryption of the Metadata in order to prevent unauthorized access.

It is expected that all users will initially access a system via a log-in process on a local machine. Permissions will be granted to a user according to a log-in status map defined below.

Authorized users (i.e. those who have passed the log-in process) will be granted access to Wrappers at three levels of user identification:

- as a named user,
- as a member of a named group,
- as any user.
- Additionally, users may be restricted to access of Wrappers within the limits of the following locations:
- local machine,
- site (e.g. the company building),
- corporation,
- anywhere.

Users may be further restricted to access of Wrappers by time limitations as follows:

- from a defined time,
- to a defined time,
- between two defined time intervals and
- anytime.

(Note: the method of identifying defined times is not specified but would normally be through a script defining the valid times and allowing repeats such as days in a week, hours in a day, holidays etc.)

Authorized users may be granted access to Wrappers according to a profile of user capabilities including:

- the right to access (i.e. make the Wrapper visible to the user),
- the right to play (read),
- the right to create (new write),
- the right to modify (re-write),
- the right to erase (delete)
- the right to administer (change any of the restri tions above).
- Rights such as browse and other rights may be added but need further consideration.

Note that these rights are granted only for the specific file accessed and may change when accessing another file.

The granting of these rights may also be controlled by an API interacting with Metadata describing Ownership, Ratings control and so on.

Operational security measures as above may be implemented in each Wrapper so that access requests and responses may be coordinated through the Application Programming Interface (API). In order to reduce duplicate security measures, equipment may bypass one or more of the above access levels where the underlying operating system provides equivalent security level replacements.

It would be desirable if the access control could be monitored to provide an audit trail for the tracking of faulty operations, unauthorized access, and commercial transactions.

D3.2 Intellectual Property Rights

The Wrapper should contain ownership parameters for Content as follows:

- Content Originator,
- Content Copyright and
- Content Owner.

The Content originator is the name of the creator of that content. It is a permanent name since the creator can never change. This value will be maintained over any Content operation such as copy, move, modify etc.

The Content Copyright is the name of the owner of any copyright contained in the content material. An example is a picture of a work of art, where the copyright of the artist must be associated with the Content. The Content Copyright is permanent except where the copyright owner has given express permission for a change of owner.

The Content Owner is the current owner of the Content and may change.

Each of these values may be associated with any defined frame or segment of the Content. Furthermore, since the Content can be represented as a sequence of Content Components (e.g. a frame of video), all parts of that Content Component is associated with the values of the whole Content Component. For example, if a small still picture is created from a frame of video, that small picture must copy the Intellectual Property Rights (IPR) values from that frame.

If any parameter has no owner, then the value is null.
A charging transaction mechanism is required for the automatic billing of resource usage, which might be provided through an API.

D4 – Notes

Several discussions which contributed the output of the present work on Wrappers and Metadata should be noted for future work. These included an analysis of applications, ideas for future implementations, consideration of possible solutions to the requirements given, or notes on the constraints of present technology.

These topics are noted here.

D4.1 Applications

In considering the categories of Applications for the Wrapper Profiles and Metadata Sets, as many potential activities as possible were enumerated, and these were then grouped into the major categories used in the main Wrappers and Metadata Annex.

This list also forms the preliminary list of Metadata Sets which will be required.

1 Pre Production
- 1.1 Scripting
- 1.2 Music Composing
- 1.3 News Assignment (Event)
- 1.4 Planning/Design
  - 1.4.1 Storyboarding
  - 1.4.2 Location Research
  - 1.4.3 Budgeting and Contracting
  - 1.4.4 Sets, Props, Costume
  - 1.4.5 Modeling

2 Production /Acquisition
- 2.1 Live News and Sports
- 2.2 Live Production
- 2.3 Video/Audio Recording,
- 2.4 Film Shooting,
- 2.5 Graphics
- 2.6 Animation
- 2.7 Motion Capture

3 Post Production
- 3.1 Editing
  - 3.1.1 Off-line Editing
  - 3.1.2 Field Editing
  - 3.1.3 A/B roll and On-line Editing
  - 3.1.4 Film Editing
  - 3.1.5 Screening, Workprint and Negative Conforming
- 3.2 Compositing/ Manipulation
  - 3.2.1 DVE, Keying, Paint, Rotoscoping, Colorizing
  - 3.2.2 Real-Time Graphic Workstations,
  - 3.2.3 Multiple M/E Linear On-line Editing
- 3.3 Sound
  - 3.3.1 Dialogue Editing
  - 3.3.2 Foley, ADR
  - 3.3.3 Music and Effects Editing,
  - 3.3.4 Mixing, Audio Sweetening, Re-recording
- 3.4 Multimedia Authoring
  - 3.4.1 Pre-Mastering, Assembling, Linking, Encoding, Bit-budget Allocation
- 3.5 Film Negative Cutting
- 3.6 Foreign Language Dubbing, Titling, Captioning, Sub-Titling, Internationalization

4 Distribution / Storage

4.1 Routing, Client/Server Access, DDRs VTRs and ATRs
4.2 Receiving Feeds, Internet Download, Archive Retrieval, Inter-facility Transfer, Relay, Backhaul
4.3 Standards Conversion
4.4 Lay-back
4.5 Quality Control
4.6 Asset Management
4.7 Uplinking

5 Emission
5.1 Playlist Preparation, Log Creation
5.2 Wholesale Delivery
5.2.1 Uplinking
5.2.2 Cable Headends
5.2.3 Satellite Headends
5.2.4 Shipping to Theaters,
5.2.5 Shipping Master to Duplicator
5.3 Broadcast
5.4 Commercial Insertion
5.5 Motion Picture Projection

6 Archival
6.1 Near-Line Storage
6.2 Long Term Storage
6.3 Deep Archiving
6.4 Asset Management

D4.2 Usage of Essence and Metadata

Various attributes of the Content were evaluated in each category of activity. The attributes were:

- the absolute amounts of Essence and Metadata
- the absolute bit rates of the Essence
- the relative proportions of each category of Metadata
- the extent to which Content is consolidated into a single Wrapper
- the kinds of access to Content

Approximate ranges were assigned to each attribute as noted in the table headings (except for access kinds), and the values were normalized to the range 0-10.

It was clear that with few exceptions, each activity required formats optimized for both streaming and richness, and for both sequential and random access.

D4.3 References and Labels

The internal method of storing references may change according to the adopted file system and may include methods such as sample offset and sub-files.

D4.4 Security

It might be desirable to include a platform-independent decryptor within the Wrapper (such as a Java applet) unless standard encryption methods are used.

D4.5 API

It is desirable to have a standard API available to lower the barrier for reading and writing the Wrapper format. It might be desirable to include platform-independent executable code such as Java within the Wrapper itself for self-unpacking/packing of both the Wrapper Metadata and Content. An advantage of a built-in API is that it hides the internal organization and specific storage methods of the Wrapper data, so the structure can be changed as long as a correct API is included.

Table D4.1: Essence and Metadata Attributes in each Activity

CATEGORY of ACTIVITY	ESSENCE		METADATA						
	AMOUNT	BITRATE	AMOUNT	DESCRIPTION	COMPOSITION	ASSOCIATION	CONSOLIDATION	RANDOM	MIXED
Scale 1 Scale 10	**Little Lots**	**1Mbps 200Mbps**	**Kilobytes Megabytes**				**Many Wrappers Single Wrapper**		
Pre-Production	1	1	3	3	1	0	1		X
Production / Acquisition	10	7–10	2–7	7	1	2	5–8		
Post - Production (Edit, Composite, Sound, Multimedia Authoring)	3	8–10	10	9	10	2	1–5	X	X
Distribution / Storage	2	7–10	1–5	5	2	4	1(in)7–10 (out)	X	
Emission / Mastering	1	5–10	1–3	1	1	2	10		
Archive	2–10	1–10	10	10	10	2	5 (Regular) 7–10 (Deep)	X	

D4.6 Essence Extraction

Means for coding and/or decoding Content might be provided in the API by supporting various methods such as:

- hardware devices accessed by device drivers
- executable code specific to a processor architecture and
- platform independent executable code such as Java.

These methods may be implemented as plug-in modules to provide user extensibility and future upgradability. Software code may be attached to the Content to provide features such as self extraction.

D4.7 Efficiency and Completeness

Different uses of Wrappers place different requirements on the performance of the Wrapper format.

For Content which is to be presented as a stream, there is an emphasis on encoding efficiency and efficiency of information retrieval. For other data, there is an emphasis on richness of data description.

It is recognized that there may be a conflict between efficiency and completeness in some applications. To help resolve this conflict, mechanisms to automatically prune or ignore optional Metadata may be required within Wrapper format converters, importers or application programmer interfaces (APIs). In the interests of efficiency, it is also desired to avoid the copying of data (particularly Essence) when converting between Presentations.

Real-time live transfer by streams may require repeating of Metadata and interleaving of structures. Certain synchronous Metadata falls into a category closely bound to the Essence, for example Timecode. There is little value in separating such Metadata from the Essence when it does not need to be accessed independently of the Content.

D4.8 Extensibility

Any new Wrapper format to be developed is required to be standardized and to have reasonable longevity, of decades or more. It is certain that new Metadata types and Essence formats will be required within the life of any standards document. Therefore, every Wrapper format is required to be extensible in the following ways:

- By addition of new Essence and Metadata types,
- By extension or alteration of data syntax and semantics.

To achieve maximum backwards compatibility, the addition of new Essence and Metadata types must be achieved without change to the underlying Wrapper data syntax with an efficient but complete documentation process, to ensure that any extensions are equally accessible to all implementations. This will depend upon maintenance of a proper Registry of data identifiers.

When unknown identifiers are encountered in the processing of a Wrapper, they (and any attendant data) should be ignored gracefully.

The extension or alteration of data syntax poses a greater problem in providing for backward compatibility. To facilitate future extensions, every Wrapper format is required to carry a Version Number, to be managed as follows:

- The Wrapper Version Number will be carried in every Wrapper, within the Overhead.
- The Version Number will be assigned by the stan dards body which documents the Wrapper format, and will be published in a Registry.
- Every device or application which accepts the Wrapper format must decode and check the Version Number.
- Every device or application which accepts the Wrapper format must decode every earlier published version of the format.
- The Version Number will be changed every time that that the Wrapper format is changed in a way which is not absolutely transparent to every imple mentation of an earlier version.
- The Wrapper format must never be changed in a way that makes decoding of the Version Number impossible or unreliable.

D4.9 Immutability and Generation Numbering

In the general case, the Content in a Wrapper should not be changed because of the possibility of unknown references to the Content in this Wrapper. Changes can be achieved by altering a copy of the Content with a different UID (this does not apply where it is known that there are no references to this Content – but reference counting is possible only within closed systems).

A possible solution is to include a Generation Number as a part of the Reference, so that the Composition Metadata in a downstream Wrapper can indicate the generation which was current at the time of creation. When content is changed, it is not altered in place, but a new copy is made with a new generation number. Then, if the previous generation is deleted for whatever reason, the downstream user can be offered the choice of trying to use the later generation or sending off a request to retrieve the older one.

D4.10 Endian-ness of 10-bit Sample Structures

When 10 bit data is carried in an 8 bit channel, issues of word sync exist. For example, 10 bit Y Cb Cr components might be packed into 32 bit words at 3 components per word, so that we have a packing of 6 samples into 4 words (16 bytes):

C_b	Y	C_r	[Y]	C_b	Y	C_r	[Y]	C_b	Y	C_r	[Y]
w1			w2			w3			w4		

There are many choices for the actual bit-by-bit packing of the components into the words. Four are considered here – "straightforward", "LSBs separate", "sparse", and "tight". The choice of packing method may strongly affect the effi-

ciency of decoding the components on a different platform from the originator.

D4.10.1 "Straightforward" Packing

"Straightforward" packing is as follows

AAAA AAAA aaBB BBBB BBbb CCCC CCCC ccxx

where aa bb and cc are the LSBs of the 10bit words, and A B C are the MSBs.

After being stored big-Endian, read little-Endian (or vice versa), this bit pattern would become:

CCCC ccxx BBbb CCCC aaBB BBBB AAAA AAAA

All similar alternatives create similar or even more complex permutations. For example:

xxAA AAAA AAaa BBBB BBBB bbCC CCCC CCcc

becomes:

CCCC CCcc BBBB bbCC AAaa BBBB xxAA AAAA

Software algorithms to untwist these are inefficient. Hardware implementations may require an additional stage of multiplexers or shifters plus an Endian-ness flag.

D4.10.2 "LSBs separate" Packing

"LSBs separate" packing is as follows

AAAA AAAA BBBB BBBB CCCC CCCC xxaa bbcc

After being stored big-Endian, read little-Endian (or vice versa), this bit pattern would become:

xxaa bbcc CCCC CCCC BBBB BBBB AAAA AAAA

This still requires untwisting, but the permutation is the same for reading and writing in all cases, whether the Endian-ness of the destination is the same as the source or is the opposite.

Hardware implementations of this scheme still require an Endian-ness flag, but no additional multiplexing or shifting is required.

D4.10.3 "Sparse" Packing

"Sparse" packing expands 10 bit samples to 16 bits each, and is as follows:

AAAA AAAA aaxx xxxx BBBB BBBB bbxx xxxx

After being stored big-Endian, read little-Endian (or vice versa), this bit pattern would become:

bbxx xxxx BBBB BBBB aaxx xxxx AAAA AAAA

This is quite simple for either software or hardware to process, but at the expense of considerable storage overhead.

D4.10.4 "Tight" Packing

"Tight" packing leaves no bits unused, and is as follows:

AAAA AAAA aaBB BBBB BBbb CCCC CCCC ccAA AAAA AAaa BBBB BBBB bbCC CCCC etc.

After being stored big-Endian, read little-Endian (or vice versa), this bit pattern would become:
CCCC ccAA BBbb CCCC aaBB BBBB AAAA AAAA Cccc *AAAA* bbCC CCCC BBBB BBBB etc.

(where AA is from the second group and *AA* is from the third sample group). As can be seen, this method causes the most involved permutations of all.

D4.10.5 Permutation during the transfer

Certain transport and interconnect technologies (for example, Fibre Channel) provide some facilities for automatic conversion of byte order during the transfer.
These facilities must also be taken into account when defining data packing. In some cases, automatic conversion can improve the efficiency of importing data from a foreign platform. However, without proper identification of the original byte order and adjustment of the flag after conversion, the automatic process might actually worsen compatibility.

D4.10.6 Conclusion

The conclusion to be reached from this discussion is that 10 bit samples add considerable complexity to the choice of a platform-neutral scheme, probably implying that Endian-ness flags must be carried as part of the Format Metadata for each Essence Component.

D4.11 Ideas for Wrapper Referencing - SMPTE 258M and HTML

One approach to References might be based upon extensions of HTML to reflect current practice within EDLs, as described in this section.

Within HTML, the concepts of labeling points within a file and referring to them from within another file are provided (by "<a name=nnnn>" and "<a href=rrrr>" respectively).

In addition, the concept of a directory or index is provided (in HTML, this is achieved by the "<img ismap src=dddd>" construct).

However, some issues are not dealt with cleanly by employing the exact scheme used in HTML. In particular, the need to address a segment of a source file, and the very common use of timecode in television do not presently map to HTML.
It is possible that a new variety of URL could be devised to address these needs, for example:

tfhs://server/mount/path/filename#hh:mm:ss.ff--hh:mm:ss.ff

In this example, the fields are as follows:

- "tfhs:" identifies the service, just like "http:" or "ftp:"
- "server" identifies the location of the material. It may be a physical address, or a local name, or an absolute name; following HTML precedent, it might include user names

and passwords for access control.

- "mount" and "path" are the route through the filesystem.
- "filename" is the name of the Wrapper; in the case of a videotape, it might be the tape reel name.
- "#hh:mm:ss.ff--hh:mm:ss.ff" identifies a segment of material. There might also be "#hh:mm:ss.ff" to identify a point, and "#anchor" to identify a specific item of Metadata, and so on.

Further work is required to understand how this notation would be applied in all the cases above. But it seems that there are some intriguing possibilities:

The source reference in today's EDL:

0001 tape66 VA1A2 C 12:12:12.00 12:12:17.00 10:00:00.00 10:00:05.00

Could become:

tfhs:tape66#12:12:12.00--12:12:17.00

(following normal HTML "defaulting" rules, if server, mount, path are omitted, they would refer to obvious defaults).

D5 – DAVIC and DAVIC Terminology

The Digital Audio-Visual Council (DAVIC) is a non-profit Association registered in Geneva. Its purpose is to advance the success of emerging digital audio-visual applications and services, initially of the broadcast and interactive type, by the timely availability of internationally-agreed specifications of open interfaces and protocols that maximize interoperability across countries and applications or services. The DAVIC concept of Digital Audio-Visual Applications and Services includes all applications and services in which there is a significant digital audio video component. The goals of DAVIC are to identify, select, augment, develop and obtain the endorsement by formal standards bodies of specifications of interfaces, protocols and architectures of digital audio-visual applications and services.

The Content structure and terminology used by the Wrappers Sub-Group is intended to be compatible with that employed in DAVIC documents, although there are some slight differences. It is thus possible for DAVIC to be a subset or an application profile of our more general definition. The following discussion reviews the DAVIC concepts contained in DAVIC baseline document #22, to show the similarity of approach.

All programming content is represented in the DAVIC system as Multimedia Components. Multimedia Components comprise one or more Monomedia Components coupled with the logical relationships between the Monomedia Components. The Multimedia Components will be created by content providers.

D5.1 Monomedia Component Types:

The basic Monomedia Components in DAVIC include the following elements:

- Characters
- Text
- Language Information
- Service Information
- Compressed Audio
- Linear Audio
- Compressed Video
- Still Picture
- Graphics

The term Essence as used by the Wrappers Sub-Group is equivalent to the actual data of a DAVIC Monomedia Component excluding any Metadata.
Multimedia Components comprise one or more Monomedia Components.
DAVIC specifies a number of different profiles. In a specific profile there may be support of a subset of the Monomedia Components.

D5.2 Content Structure Definitions

A **Content Item Element** (CIE) is indivisible, and consists of Essence of a single monomedia type, plus any Metadata directly related only to that Essence.

Wrappers need to be concerned with even smaller units of Essence, reflecting the division of Content Item Elements for interleaving and multiplexing. The Wrappers Sub-Group calls these smaller units Content Components (or Essence Components when specifically referring to Essence).

The Wrappers Sub-Group uses the term Content Element instead of Content Item Element.

A **Content Item** (CI) consists of a collection of one or more Content Item Elements (CIE), plus any Metadata directly related to the CI itself or required to associate the component parts (CIEs) together.

A **Content Package** (CP) consists of a collection of one or more Content Items or Content Item Elements, plus any Metadata directly related to the CP itself or required to associate the component parts (CIs and CIEs) together.

For each Monomedia and Multimedia Component the coding format is specified, as well as applicable constraints for coding of the Components. The coded representation of each Monomedia Component is defined to be packetized in specific ways that permit to include time stamps to support mutual synchronization of multiple Monomedia Components.

Among the important objectives for such a structure are (with no priority in the listing):

- Random access to individual Content Item

Elements, e.g. for transfer, updating, deletion and adding.
- Extensibility of elements, i.e. adding new attributes must be possible with backward compatibility.
- Global referencing possibilities in and out of pack ages.

D5.3 Metadata

DAVIC divides Metadata into two categories: Content Management and Navigational. Content Management Metadata refers to Metadata that allows the Content to be

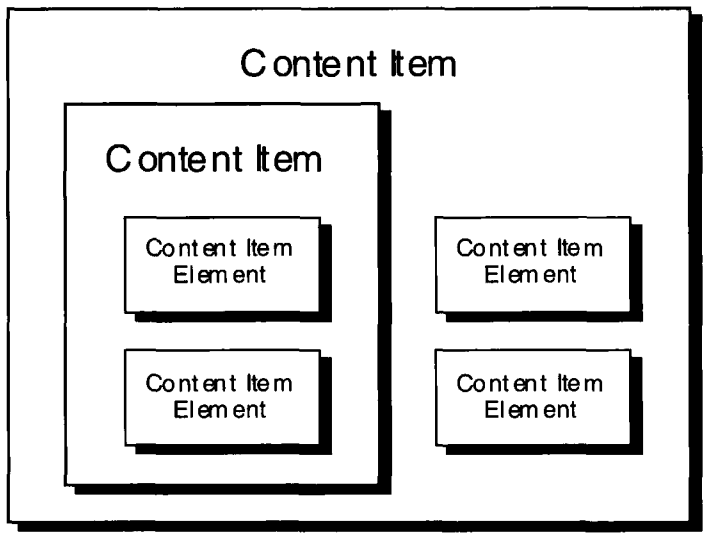

Figure W.5.1 Illustration of Content Items and Content Item Elements

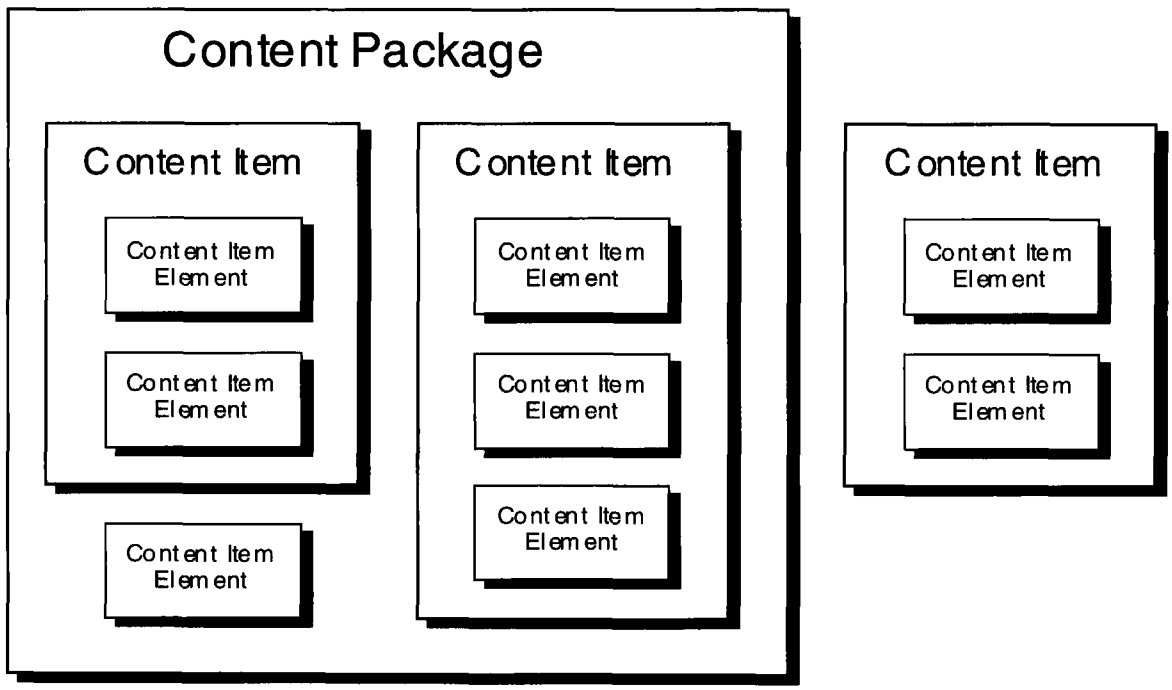

Figure W.5.2 The relationships between Content Packages, Content Items and Content Item Elements

produced, processed and transferred. Navigational Metadata is used to enable use of the server content, as controlled by the client and server application executables. This may include navigation, embargo and deletion control, copyright management, and user access control.

A core set of Metadata items are required to be provided with each Content Item. This small core set provides for the basic management of server items.

Metadata can be related to Monomedia Components (one channel of audio, video, stills, graphics, text) or to Multimedia Components (Content Item Element, Content Item, Content Package).

Metadata which is related to a Content Item Element, (a subsection of the Content, e.g. a frame or scene), is referred to as Time Variant Metadata. A subset of those is formed by Synchronous Metadata that are needed for real time live transfer.

Metadata Types define broad classes of Metadata. Metadata Attributes define specific parameters within a Metadata Type and take either an explicit value or a reference to a data source.

D5.4 Overall Structure

The structure used should be hierarchical. It is recommended that certain sets of types and attributes are allocated to application dependent Metadata Sets. These would be registered by an independent registration authority like SMPTE. The following example shows one possible arrangement of Content Packages, Content Items and Content Item Elements, Metadata Types and Attributes within a Wrapper.

D5.4.1 Metadata Data Structure Example

```
[Wrapper]
    [content package]
        description.title = "Days of our Lives"
        description subtitle = "Episode 24"
        description.owner = "XYZ-TV"
            [content item]
                description.title = "first program segment"
                temporal.duration = 00:00:15:00
                    [content item element]
                        video.format = "525 4:2:2"
                        video.location = "server1\days24_1.vid"
                    [end content item element]
                    [content item element]
                        audio.format = "48kHz AES/EBU"
                        audio.location = "server2\days24_1.aud"
                    [end content item element]
            [end content item]
            [content item]
                description.title = "second program segment"
                temporal.duration = 00:00:15:00
                    [content item element]
                        video.format = "525 4:2:2"
                        video.location = "server1\days24_2.vid"
                    [end content item element]
                    [content item element]
                        audio.format = "48kHz AES/EBU"
                        audio.location = "server2\days24_2.aud"
                    [end content item element]
            [end content item]
    [end content package]
[end Wrapper]
```