Technical Report
Preprint of June 6, 2005

Methods to Store Metadata within Motion JPEG 2000 Files

Glenn Pearson

Communications Engineering Branch
US National Library of Medicine
NIH/HHS

Summary

The specifications of Motion JPEG 2000 (MJ2) and related documents in the JPEG 2000 family are analyzed, to identify and assess the multiple mechanisms available for storing metadata within a MJ2 file and the types of metadata that can be so accommodated. Adequate mechanisms seem to exist for associating metadata with the production as a whole, individual tracks, and individual frames. In those cases, attention is turned to alternative forms such metadata might take, both those suggested within the JPEG 2000 family specifications and those defined externally.

Conversely, existing mechanisms seem inadequate, or at the least underspecified, for embedding and synchronizing scene-based metadata. Various workarounds and possible extensions are discussed, as well as alternative wrapper formats and methods of metadata synchronization from without.

Of the externally-defined metadata systems considered, those of MPEG4 (whose file format is a cousin of MJ2) and MXF (which is paired with JPEG 2000 for Digital Cinema) get the most attention herein, but many others, well known to the archival community, are touched upon.

This document strives to identify various approaches and, ideally, their pluses and minuses, to provide a basis from which recommended practices or refinement of standards could be developed for video archiving.

Contents

Section I. Overview

The opinions expressed herein are the author's own, and do not necessarily represent any position of the National Library of Medicine.

Motion JPEG 2000 (MJ2) is a relatively new format for encoding video. It is defined as part of the JPEG 2000 family of standards [ISO/IEC 15444], and has been described thusly [Lee]:

"JPEG 2000 Part 3 defines Motion JPEG 2000… and is primarily based on the technology in Part 1 with the addition of a file format. It results in an encoder that is significantly less complex than the MPEG family of standards (because motion estimation is not used) and provides full random access to the individually coded frames. It is intended for applications such as digital sill cameras with burst capture mode, video editing in postproduction environments, and digital cinema archive and distribution."

The initial motivating application for MJ2 was to take short video clips with digital still cameras. The thought was that a single embedded hardware chip would JPEG-2000-compress both stills and frames within clips. Although such chips have been produced by a number of vendors, they have not yet been incorporated in marketed cameras, in spite of the recognized superiority of lossy JPEG 2000 images over lossy JPEG of corresponding size. One can speculate that this is due to high cost-sensitivity at the low end of the market (e.g., cell phone cameras) and real-time performance problems at the high-megapixel end of the market.

Where JPEG 2000 has been reasonably successful is in post-capture editing and presentation of stills, including, apropos of an archiving context, annotation with metadata. It is possible, but by no means certain, that MJ2 could likewise develop in the future as an important video archiving format, particularly given its lossless compression mode. Lossless compression reduces original filesize by roughly a factor of three. On current consumer PC hardware, the resultant file will not playback at full frame rate and size, unless frame dropping or a drop of quality to some lossy level is done. However, equipment a decade hence will be more powerful, and in any event, other provisions can be made for delivery, as opposed to preservation. For instance, on-the-fly extraction and delivery of a reduced-bandwidth MJ2 stream from a higher-quality MJ2 file by a specialized server has been demonstrated.

Like most video formats, MJ2 is composed of both metadata and media-data (e.g., video frames, sound samples). For our purposes here, the metadata can be further subdivided into "structural metadata", primarily required to allow file decoding, and "descriptive metadata" (in the very broadest sense) of a mainly optional nature. While the structural metadata is described in the defining (rather convoluted) standard in extensive detail, the descriptive metadata, beyond a minimal framework, is left largely to the imagination. Here we explore that framework and suggest how it might be utilized and populated.

That such an analysis is necessary is to some extent dictated by the positioning of MJ2 as an "amateur digital video" format. MJ2 is less complex internally than the professional MPEG2 / MPEG4 formats and their archival wrappers such as MXF. Specific MJ2 provisioning for metadata tends to be an afterthought of still-image metadata. The information about the metadata framework is also diffused across multiple standards documents, and employment of the framework, particularly in a video context, is underspecified.

MJ2 was at one time considered a candidate for digital cinema. The Digital Cinema Initiative has since chosen lossy JP2 frame compression within an MXF wrapper. If MJ2 is “too simple”, then DCI can be said to be “too complex”. (DCI is further encumbered with extensive encryption and time-and-place security features, which limits its appeal as an archiving profile.)

Nevertheless, one can also conjecture a “video archiving profile” for simplified MXF+lossless JP2, which is one conceptual competitor to what is presented here. Such a profile might be more able to benefit from broadcast and studio tools developed for DCI or other production purposes. (The MXF standards committee reportedly has a JP2 embedment within MXF as a low-priority work item.)

On the other hand, if the video archiving community embraced MJ2, and further shaped and invigorated the format with its “own profile”, there might be less pushback from studios than with MXF, and tool vendors might well emerge to support that profile.

(It is also possible to conceive MJ2 embedded in MXF, but probably no one’s working on that. Other competing codecs to JP2 include VC-1 (aka WM9) and MPEG4. In particular MPEG4/AVC has a lossless mode.)

This document chiefly explores mechanisms for storing descriptive metadata within the video file. Not everyone will agree that is desirable, and certainly there is some information that makes more sense to store (exclusively or redundantly) in external files or in a database. Such external storage presents another alternative for dealing with “difficult metadata” described towards the end of this report.

Section II. Provisions for “Easy Metadata” – Per Movie, Per Track, Per Frame

II.1. Introduction via Still Images and their Metadata

Since a MJ2 movie is made of frames represented by JPEG 2000 (JP2) stills, it is helpful to first consider facilities provided for metadata with JP2 images, particularly since incorporation of such facilities is one path to metadata in MJ2.

The Basic JP2 File Format

Core JP2 is defined in Part 1 (and its amendments) of the standard [ISO/IEC 15444]. A basic JP2 file can be considered to have two aspects:

1) The overall JP2 file structure. This file structure is made up of “boxes”: labeled, nested regions, each started with a small header stating the box type and its size. The top level of the file structure is summarized in Table II.1-1. (The file structures of JP2 and MJ2, while similar, are of course distinct. But they can be merged in a defined way, about which more will be said later).

In a JP2 file, metadata can be sequestered in the Intellectual Property, XML, and UUID boxes:

- The Intellectual Property box is undefined in Part 1; Part 2 fleshes it out.
- An XML box can hold any arbitrary XML-form metadata. The average JP2 reader can at best handle unrecognized XML in an uninterpreted manner (e.g., displayed as a generic tree).
- The content of UUID data can be of various forms, such as simple strings. (Indeed, UUID and the UserData/Copyright box are the only forms currently for “simple textual” metadata, as mentioned in Part 12 2nd ed.).

Example content for these boxes will be discussed in a later section.

2) The codestream. This is the encoded image (or in video terms, a frame of media-data), found within a Codestream box. Metadata can be embedded within the encoded image as one or more optional “comment marker segments” (COM). This is most plausible in the main header of the image, but could also appear in any tile-part header. (A tile represents a rectangular subdivision of an image; typically, a video-sized image is encoded with just one tile. A tile-part further subdivides by components, normally 3 for color movies). The flag within a COM indicates whether the data is binary or Latin-1 characters.

Table II.1-1. Top-level Structure of a JP2 File. A JP2 file is made of a series of "boxes", where each box (contiguous regions of usually-binary data) has a length, a 4-ASCII character type identifier [shown here in square brackets], and contents of variable size. Boxes can be nested; a parent box is a "superbox" of its children. The top-level boxes are indicated

Name [& box type]	**How many**	**Content**
JPEG 2000 Signature ['jP '].	1	This holds a fixed string.
File Type ['ftyp']	1	Contents are a "brand" of "jp2 " and integer minor version number. (Also listed, if it's a Motion JPEG 2000 file, is 'mjp2' as a "compatibility brand". There may be other compatibility brands as well.)
JP2 Header ['jp2h']	1	This superbox is made up of an image header and at least one color specification, as well as optional items not further considered here. The image header, pertaining to the first codestream below, includes image height and width, number and bitdepth of components, and a Boolean flag if an Intellectual Property box is provided.
Codestream ['jp2c']	1 [see note at right]	Codestream format is defined Part 1 Appendix A. Briefly, there is a main header, then (for an image subdivided into one or more rectangular "tiles", which also subdivides each component into "tile parts") a series of tile-part header + data pairs. **Note: There may be additional contiguous codestreams**; software readers are expected to ignore all but the first.
Intellectual Property ['jp2i']	0,1	Not defined in Part 1. Defined as XML in Part 2 for JPX.
The remainder are vendor-specific and may be ignored by conforming JP2 readers…		
XML ['xml ']	0+	Any XML. May be "anywhere" in the file after File type box.
UUID ['uuid']	0+	Any data whose format is identified by a 16-byte UUID (Universal Unique Identifier, discussed below). May be "anywhere" in the file after File type box.
UUID Info ['uinf']	0+	This superbox contains two boxes, one with a list of UUIDs, and the other with a URL. It allows lookup of external help on particular UUIDs found in the file. Unlike 'uuid', must be at top level of file.
(Unknown)	0+	A type other than those defined in the standard is tolerated; conforming readers ignore it.

Note that the definitions of XML and UUID boxes (in Annex I.7.1, I.7.2) specifically allow these to be found throughout file (and presumably not just at top level), even though the discussion and diagram in I.2.2 might suggest otherwise. The definitions of JP2I (in Parts 1 and 2) are silent on this matter.

More about UUID

The UUID box has, as part of its header, a Universally Unique Identifier (UUID), or what Microsoft calls a Globally Unique Identifier (GUID). A UUID is essentially a 16-byte number and in its canonical form may look like this:

550E8400-E29B-11D4-A716-446655440000

Synopsizing here from the Wikipedia description, a UUID is an identifier widely used in software construction, standardized by the Open Software Foundation as part of the Distributed Computing Environment. UUIDs let distributed systems uniquely identify information without significant central coordination. Thus, any developer can create a UUID to identify something, with reasonable confidence that it won't unintentionally be used by anyone for anything else. Information labeled with UUIDs can therefore be combined into a single database without need to resolve name conflicts. The "version 1" generation scheme for UUIDs (simplifying) concatenates the version, the generating computer's MAC address, and a datetime value. Other schemes rely on random numbers (version 4), or hash a URL (versions 3, 5). Popular UUID generators include those by Microsoft (calling them Globally Unique Identifiers or GUIDs), and within Java release 5.0. UUIDs can form part of an ASN.1 ObjectID (OID) or Unique Resource Name (IETF RFC 2142).

UUIDs are documented as part of [UUID ISO], and by further efforts underway through ISO [ISO FDIS 9834-8] and IETF (Internet Draft RFC, Dec. 2004). Centralized generators and registration authorities are possible, for example, http://www.itu.int/ITU-T/asn1/uuid.html.

JP2's Use of UUID and Interpretation as KLV

JP2's UUID box, in a movie content, is easily made compatible with defined metadata elements expressed in SMPTE's Key-Length-Value (KLV) format [SMPTE 336M]. The "key" is the UUID stored in the box header. The "length" is already provided by the box length field. The "value" can be anything, binary or text, but is defined by the key.

The KLV format is widely used to pass information that, because of its functional use in video equipment, must have a representation that is both compact and tightly controlled. (It is the basis for the MXF format, and also used for user data in AAF; Appendix 3 provides more insight.)

Thus, while UUID could in theory be the identifier of an application that can read/write complex, heterogeneous "value" data, it often is an element identifier for the data itself, or of a set of data as a fixed structure or simple tabular form. For instance, in the case of registered metadata elements with known UUIDs found in the public SMPTE Metadata Dictionary and Registry [RP210] (designed for MXF and AAF), "value" is typically a string or a simple binary value, or a defined set thereof. MXF does not nest registered KLVs more deeply than that, but uses a different technique for "logical nesting". (JP2 wouldn't necessarily have to abide by this restriction for non-SMPTE metadata.) Predefined SMPTE hierarchical frameworks for logical nesting, such as DMS-1, are discussed later.

Since neither keys nor binary values are self-documenting, auxiliary UUID Info boxes provide URLs to external interpretive help. More precisely, a given "UUID Info" box has a list of UUIDs and a Data Entry URL box ['url '] holding the URL string in UTF-8 form.

KLV labels that are not found in the RP210 database are usually treated as "dark metadata" in MXF, and might be so treated in MJ2. Such items can be analyzed to see if their contents are in XML form for display purposes.

An example of user metadata in UUID/KLV form, by the International Press Telecommunication Council, will be discussed later.

JPX Extensions

Part 2 of the 15444 standard allows extensions to this format, resulting in “JPX” files (which have either .jpx or .jp2 extensions; the latter requires the brand to be ‘jpx’ and compatible brands include ‘jp2’). Since the extensions are a hodgepodge of different things, another top-level box, “Reader Requirements”, is introduced, to indicate what extensions will be found in the file. At the minimum, this normally includes the tag, “codestream is compressed as defined in Part 1”. (Also, if Part 1’s UUIDInfo box was used, there are additional “vendor feature” and “vendor mask” aspects to Reader Requirements.)

The extensions about metadata and related video structuring concerns are concentrated in Annexes M and N of Part 2.

Looking at the Intellectual Property box introduced but not defined in Part 1, there is now a fully defined metadata structure given in Annex N, with a publicly-available DTD and Schema [jpx xml]. This structure (and related structures for XML boxes) is derived from the DIG35 metadata v 1.1 effort [dig35] of what had been the “Digital Image Group” and is now the “International Imaging Industry Association” (I3A). This will be discussed further in a later section, as will other types of JPX extensions.

In addition, binary MPEG7 information can be contained, as also detailed later.

Table II.1-2. Additional Top-Level and Child JPX Boxes Related to Metadata

Name [& box type]	**How many**	**Content**
Reader Requirements [‘rreq’].	1	Contains a list of integer codes of Part 2 extensions found in the file.
MPEG-7 Binary [‘mp7b’]	0,1	Contains metadata in MPEG-7 binary format (BiM) as defined by ISO/IEC 15938. See: Part 2 Section M.11.19
Others	opt	See further discussion elsewhere about JPX’s use of [‘xml ’] and of Intellectual Properity Rights [‘jp2i’] boxes.

Part 2 Section M.13 describes ways in which “multi-mode” JPX content can be combined with other content, such as MJ2 and/or MPEG7, into a single file. In each case, the standard reader for each mode sees only its own data, but a “smart” reader would see all modes (and presumably a similar “smart” editor would be needed to create the multi-mode file).

II.2. Introduction to Motion JPEG 2000 and Metadata

A simple MJ2 presentation may be made up of a single video and single mono or stereo audio track. A more complex one might have multiple video tracks overlapping in time, such as in layers (perhaps blended by alpha-channel transparency) or geometrically arranged on the display. Positioning, including rotation, is controlled by transformation matrices. An optional edit decision list (EDL) controls what portions of media are displayed and when.

Stored on disk, a presentation may be in two forms:

- Self-contained in a single file, with both metadata and media data.
- In multiple files. In that case, a parent “presentation file” conforms to the MJ2 format, and houses all the conforming metadata for the presentation. The parent then points by reference into places in the other secondary files where media samples exists. The other files aren’t required to be MJ2 conforming; it is hoped to permit incorporation of existing media files, in whole or part, without reformatting or header wrapping. (Practical implementations will no doubt limit secondary media files to audio files, MJ2, or still-image JP2.) The important point is that, officially, any metadata, including the curatorial, in the secondary files is skipped and cannot be included by reference, only by explicit copying.

The detailed MJ2 format is rather flexible. We’ll sketch the overall possibilities, but emphasize the MJ2 “Simple Profile”, limited to a single self-contained file with additional constraints (described later), such as that video frames are in temporal order.

As with JP2/JPX, a MJ2 file is made up of a series of boxes. The design philosophy of the format strives to separate, rather than interleave, the raw media content from metadata about it (such as time and data indices).

Specific Provisions for Metadata
MJ2 is specified as Part 3 of the JPEG 2000 standard. The top level structure, particularly parts related to metadata, are detailed in our Appendix 1. Briefly, provisions for metadata are sketchy; defined locations for metadata (in addition to the per-frame COM segments of JP2) are:

Per movie or per A/V track

- Optional “User Data Box”, each with one or more “Copyright Boxes” (holding a string and language code). This ‘udat’ box appears designed to be a general container, but nothing else is defined beyond copyrights.

Per Video Track

- A deeply-buried “JP2 Header Box” (see Table II-1), that could (by one interpretation) contain child XML and/or UUID boxes;
- An optional “JP2 Prefix Box”, containing a shared leading part of all the codestreams (frames), that could contain shared COM segments.

Is it compliant with core (non-hybridized) MJ2 to include XML or UUID boxes? The argument for this is that a video track contains a sample table (See our Appendix 1 Table E) that has a JP2 Header, and that Part 1 allows optional XML and UUID boxes anywhere within a JP2 header (and elsewhere).

A counter argument is that if they were allowed, they would be mentioned in Part 3. Furthermore, in the case of UUID, this is usually paired with UUID Info, which in Part 1 must be at the top file level; no such optional top-level box is listed in Part 3.

But no such doubt pertains to a file to be both JP2 and MJ2 compliant, by including both “brands” in the header and including a still image for the JP2 viewer (which presumably a smart encoder would allow to be one of the existing frames). In that case, the optional JP2 boxes like XML or UUID would be allowed. (Nevertheless, today’s average MJ2 player will no doubt ignore XML and UUID boxes, and even if it didn’t, would it be expected to know about, say, Annex N structures? Not yet.)

Considerations Using COM Segments

It is desirable for encoders to maximize use of the JP2 Prefix Box, in order to reduce filesize and decode time. This can be done by arranging that the beginning of each codestream have as much that is fixed, that is, common throughout the movie, as possible. In each codestream, the main header comes first (with its COMs at the end), then the various tile-part headers (each potentially with their own COMs at the end). If the encoder always sticks comments specific to an individual frame at the end of the last tile-part header for that frame, it would maximize the JP2 Prefix Box potential. Conversely, COM-based comments that apply uniformly to the whole movie (or at least whole track) should go in the main header.

SMPTE Timecodes and COM Segments

MJ2 does not use video timecodes in any functional way, relying on timing tables instead. Nor (regrettably) does it have a designated place to store timecodes. Since these are associated with individual frames, the use of COM seems plausible. Since they will be unique for each frame, the suggestion in the previous paragraph about the last tile-part header seems apt. To be interoperable and allow software to recognize and parse them, a convention would need to be developed for their tagging and syntax.

As for how other related file formats treat timecodes, MPEG-4 allows but does not require a timecode track, called a Clock Reference Stream (‘crsm’). Optional ‘sync’ references from one track to another (not necessarily crsm) can over-ride the default synchronization.

QuickTime evidently has a similar system; its equivalent of ‘sync’ is ‘tmcc’ [mp4ra].

II.3. Using MPEG4-Style MPEG7 Metadata via the ISO Base File Format

Part 12 of the standard presents an ISO Base Format, from which both MPEG4 and MJ2 are said to "derive", although this is a retrofit. The MPEG4 activity in particular is resulting in new features being added to this base format. It would seem that it should be possible to therefore create a hybrid MJ2/Part 12 file that would take advantage of this. This is an alternative to including MPEG7 via a hybrid MJ2/JPX approach discussed above.

In particular, the second edition of Part 12 (April, 2005; available as free download from ISO) introduces a new box type, 'meta', for descriptive or annotative metadata (Table II.3-1). Reader support for this is optional. At most one 'meta' box can appear in every case:

- At top file level, for a file that is primarily metadata;
- Within the 'moov' box, to annotate the movie as a whole;
- Within a 'trak' box, to annotate an entire stream.

Although the contents of 'meta' can be interleaved, it appears to be considered static, untimed information, even in its 'trak' embedment. Only text-form or binary MPEG7 is offered so far. If a file-level meta box with MPEG7 is present, the 'mp71' brand should be included in the 'ftyp' box.

Unlike basic MJ2, which does not allow incorporation by reference of metadata in a second file (only of media data), the Part 12 2nd ed. does permit such referencing. Thus, MPEG7 data in an external file can be incorporated if desired.

More Direct Incorporation of Base File Format MPEG4-like Features

The ISO workgroups may be amendable to bringing more of the Part 12 items explicitly into future MJ2 amendments. Edwards and Stoeffer [Edwards] mention, under directions for new work in Digital Cinema, that regarding metadata in MJ2, the "MJ2 file format based on the MP4 syntax is expandable to integrate additional data, like the SMPTE metadata directory or unique extensions."

Table II.3-1. Contents of Meta Box.

Name [& box type]	**How many**	**Content**
Handler ['hdlr']	1	Indicates format. Such handler_type values aren't yet defined *except* for MPEG7. In that case, choices are "mp7t" (text XML found in an 'xml' or 'pitm') or "mp7b" (binary XML found in a 'bxml' or 'pitm'). See also our Appx 1 for hdlr.
Text XML ['xml ']	1 of these 3 "Primary Items"	Directly-stored metadata as UTF-8 XML (or UTF-16 if data starts with a byte-order mark).
Binary XML ['bxml']		Directly-stored binary XML, for which the handler must indicate the well-defined binary method.
Pointer to Item ['pitm']		The meta is stored elsewhere, or divided into extents. The box just contains an item ID (found in the Item Location box), which the handler interprets.
Data Information ['dinf']	0,1	"Lists other files in which metadata values (e.g., pictures) are placed." [what's that mean?] (See also dinf in our Appendix 1).
Item Location ['iloc']	0,1	Says where in those files each item is located (e.g., in the common case of multiple pictures stored in the same file.) This may be thought of a set of binary tables. Each table refers to a particular file by number (either zero if local, or the index of files in the data information box). Each table entry has an item ID, the offset from the filestart, and the length of the metadata (or zero if entire file). A metadata item can be further broken up into multiple "extents", to allow interleaving with other content. (There are additional complications.)
Item Protection ['ipro']	0,1	Provides an array of (undefined) item protection information, for use by the Item Information box.
Item Information ['iinf']	0,1	Has extra information about an item, including a symbolic file name. Also gives MIME type, and, optionally, file encoding (e.g., "gzip", "compress", "deflate").
(others)		

II.4. Examples of Metadata to Consider for MJ2 Embedment

These are generally metadata sets that are defined outside of the JPEG 2000 documents, unless indicated otherwise.

Introduction – More about Hybridization of MJ2 with JP2/JPX Still Image Formats

One approach to storing descriptive metadata is to create a hybrid file with JP2 or JPX. In theory, the metadata stored relates to the "thumbnail" image, not to the movie. But it's not much of a stretch to consider this as metadata for the movie as a whole. [Can one assume that the first frame of the movie can serve as the thumbnail image if you don't explicitly store a thumbnail?]

However, much potential JP2/JPX metadata makes sense only if applied to a particular image (e.g., f-stop), not to an entire movie. Careful selection of a reasonable subset would seem needed. Furthermore, roles peculiar to video production, and temporal information, are not covered.

UUID Metadata if Hybridizing with Part 1 (JP2)

These, as well as XML below, might also be legal with unhybridized MJ2 as discussed earlier.

Wire Service Metadata. The International Press Telecommunications Council (IPTC) has developed a IPTC-NAA Information Interchange Model (IIM) for electronically transferring a typically-compound data object, such as one or more photographic images, text, or any arbitrary data (including presumably video, although the thrust of this is more towards print). An "envelope" around the object provides information about data types, file formats, caption, news category, and dateline. Alternatively, existing media file formats that can incorporate IIM metadata can be used. Jung et al [Jung] have registered a UUID to allow IIM metadata to be embedded in JP2/JPX files in KLV form. Nevertheless, such embedment seems not a good idea when referring to a compound digital object. (Also, be aware that IPTC has also developed a separate XML-form envelope, "NewsML"; see www.newsml.org)

A Choice of UUID or Textual XML Metadata

Video Production, Distribution, or Digital Archiving Metadata. Among the descriptive methods for media files, some are available in both KLV and XML forms. Two examples:

- DMS-1. Part of MXF (Material exchange Format) is "Descriptive Metadata Scheme 1"DMS-1 [SMPTE S380M], also known as the "Geneva Scheme". It provides three descriptive "frameworks":

    - Production (as a whole)
    - Clip (scenes from a given source – like "track")
    - Scene

- P/Meta – Business-to-business metadata for producers, studios, and archives.

Table II.4-1. Some Examples of Popular Library/Museum Archiving Metadata Systems with a Standardized XML Representation. Adapted from [RLG]. The third column is an un-nuanced guess at how reasonable it is to embed the given metadata within a video file. Basically, individual records seem more plausible than descriptors of collections of records, or complex digital objects.

Standard	**More Info**	**Embed?**	**Scope**
Art Museum Image Consortium (AMICO)	www.amico.org/distribute/ docs/xml/amicorecord.dtd, amico.xsl	N	Individual records. CDWA-tagged text with a conversion to XML.
Dublin Core (DC)	dublincore.org	Y	Individual records. A minimalist approach. There are Qualified and Extended versions.
Encoded Archival Description (EAD)	www.loc.gov/ead/		An organized collection of records
Making of America II (MOAII)	http://sunsite.berkeley.edu/ MOA2/	N	Digital objects, particularly digitized photos, diaries, books. Informed creation of METS.
MARC 21 (particularly MARCXML)	www.loc.gov/standards/ marcxml/	Y	Individual records. A complex, detailed encoding traditionally used by US libraries. Covers books, journals, maps, sound recordings, movies, etc.
UNIMARC	www.ifla.org/VI/3/p1996-1/unimarc.htm, (XML:) www.bookmarc.pt/unimarc/	Y	Individual records. A complex, detailed encoding, with mappings to individual country MARCs.
Metadata Encoding and Transmission Standard (METS)	www.loc.gov/standards/ mets/	N	Digital objects. Describes in a single record a set of object locations and associated metadata (in, say, a MARC, MODS, DC, etc. format)
Metadata Object Description Schema (MODS)	www.loc.gov/standards/ mods/	Y	Individual records. A simpler, more human-readable subset of MARCXML
SPECTRUM	www.mda.org.uk/spectrum. htm	N	Collections of individual records. Required in UK.
Text Encoding Initiative (TEI)	www.tei-c.org, sourceforge.net/projects/tei/	?	Individual records/digital objects. Used for transcriptions of textual materials (with optional links to page images)
Visual Resources Association (VRA) Core Categories	www.vraweb.org/vracore3. htm	Y	Individual records. Of moderate complexity; XML schema under development.

More Textual XML Metadata if Hybridizing with Part 1

Library or Museum Metadata. Examples of cataloging and curatorial metadata, some of which one might plausible want to embed, are given in Table II.4-1. Not shown in the table, Project MIC incorporates aspects of both Dublin Core and MPEG7.

Video Production, Distribution, or Digital Archiving Metadata. Besides the methods already mentioned under UUID above, MPEG-7 can be stored in textual XML form. However, both JPX and ISO Base File Format have separate provisions for storing MPEG-7 in a compressed binary form. More so than some of the systems in the Table II.4-1, MPEG-7 provides support for describing the temporal aspects of audio and videos in detail.

Domain Specific XML. There are additional domain-specific XML metadata schemas that have been proposed for JP2 files, reflecting the broader purpose for which a given image is made. For instance:

- A paleontologist might use OpenGIS's Geography Markup Language (GML) [Jung].
- An ecologist might use a biodiversity schema such as the Taxonomic Working Group's Structure of Descriptive Data (SDD) [BDEI]. (The latter reference also proposes to use JPIP, yet another part of the JPEG 2000 standard, to embed active elements in still images, to let them circulate freely, acquiring annotations, and when connectivity allows, "phone home" to a master copy to share annotations across the community.)

DIG35/DSC Metadata if Hybridizing with Part 2 JPX Still Image Format

Unlike the case with JP2 or MJ2, JPX provides one well-defined set of descriptive data. Annex N gives a series of XML Schemas and DTD for this data, which would be presented as XML documents in a set of five boxes. (These are basically subtrees of the DIG35 specification and its ANSI DSC incarnation.) Specific xml documents are shown in Table II.4-2. When using these features, they need to be tagged in the JPX-specific Reader Requirements box (i.e., "contains Content metadata", "contains History metadata", "contains Creation metadata", "contains IPR metadata"). If the JP2 'jp2i' box is present, an IPR bit must also be turned on in the ImageHeader.

Relationship of DIG35/DSC to NISO z39.87

As further background, note that early work on still-photo metadata resulted in 2002 in the original NISO "z39.87 Draft for Trial Use" [NISO] data dictionary and model. A mapping of this form to XML was done by the MIX Project. Separately (but no doubt cognizant of this work), the DIG35 group (subsequently renamed) developed their specification. This formed the basis for Annex N; the structure was further formalized and slightly expanded as ANSI JPEG 2000 DSC Profile in 2004. In turn, in March 2005, the much-revised "NISO Data Dictionary Draft 1.2" was based on DIG35, TIFF 6, TIFF/EP, and EXIF vocabularies. The motivation is to capture modern digital camera configurations, settings, readings, and optional user data, without being tied closely to a particular compression method or file format. A draft of the updated mapping to XML is available [MIX].

Limits of Applicability of DIG35/DSC/z39.87

Only a limited subset of these metadata elements is apt for our needs. It was mentioned earlier that many such elements are more applicable to a given frame than to the video as whole. Furthermore, for historic films and videos, most technical information (e.g., camera model, lens stop, lighting, etc.) isn't known or of interest to the archivist. Technical information that is known may be for the capture and digitalization process, not the original analog or film media.

Table II.4-2. JPX's Standard Metadata

Name [& box type]	**XML root name**	
Image Creation [‘xml ’]	IMAGE_ CREATION	Typically camera and lens settings, and other capture conditions.
Content Description [‘xml ’]	CONTENT_ DESCRIPTION	This is the main repository of curatorial metadata about the image.
History [‘xml ’]	HISTORY	A log of image capture and manipulation events.
Intellectual Property Rights [‘jp2i’*]	IPR	Moral rights, copyrights, and exploitation information.
Image Identifier [‘xml ’]	IMAGE_ID	Information to uniquely identify the image.

* Referred to, evidently incorrectly, as [‘ipr’] in some locations in Annex N.

There are pages of specifications for the subfields of these, which are broadly applicable to MJ2, except that the still-image orientation has little sense of “scene”, “storyline”, and the environment involved in video production. It seems to be the idea that “Image Creation” and “History” are filled in by equipment and software. Mapping between JPX and other schemas such as Dublin Core have been proposed [Colyer].

MPEG7 Metadata if Hybridizing with JPX or ISO Base File Format

As indicated earlier, JPX allows an MPEG 7 box with content in BiM format, a binary version of XML. Part 12 does likewise (as well as allowing text form). There have been several efforts at making a binary version of XML, so that metadata can be sent in a considerably more compact form. The version specified for MPEG7 (probably encumbered by patents) was developed by a working committee representative of Expway, a French company offering a “BiM compliant” product, “BinXML”.

Tools for manipulation of MPEG7 metadata with video files (though not MJ2) include Ricoh's MovieTool, for MPEG1 videos, and IBM's Multimedia Annotation Tool.

Other Video Production Aspects

Other potential metadata formats that have not been investigated here, but that might be embeddable include MPEG21 (the digital rights companion of MPEG7) and the metadata within GXF.

Also, MJ2 supports simple edit decision lists (EDLs). But it is not a suitable format for retaining details of complex video-editing sessions; it is not a serious competitor to formats like Advanced Authoring Format (AAF) or Avid's OMF.

Section III.
MJ2 Embedment of More Difficult Metadata

These include possible implementations of:

- Scene Level Descriptions
- Regions within Frames
- Moving On-screen Objects
- Timed Test – Captions and Subtitles
- Content-based Indexing and Retrieval

III.1. Implementing Scene Level Descriptions

Working with Core MJ2

Two techniques suggest themselves:

1) Using a COM box, one could easily add a comment to any given JP2 frame. (Recall the suggestion earlier to use a trailing tile-part COM box to avoid limiting effectiveness of shared prefix feature.) However, a scene description should apply to multiple frames. A convention would need to be invented for describing frame ranges.

2) Make each scene a separate track, with appropriately offset start times for sequential playback. Then put the metadata in the track headers. This is straightforward, although the track fragmentation may strike one has unappealing, and the result is no longer a “Simple Profile” MJ2 file.

Hybridizing with JPX GIF-style Animation

Here we discuss additional extensions defined as part of JPX, within 15444 Part 2’s Annex M, that allow more than one image to be handled, either composed in layers and/or animated over time. Two possible techniques are:

1) Use an “Association” box to link one or more ‘xml’ annotation boxes with a “number list” box of image numbers (an “image set”). See Table III.1-1, and also Annex M’s Fig. M.29.

Table III.1-1. The JPX Association-Related Boxes

Name [& box type]	**How many**	**Content**
Association [‘asoc’]	0+	This superbox allows several other boxes (e.g., boxes containing metadata) to be grouped together and referenced as a single entity.
Number List [‘nlst’]	0,1	A child box of Association, specifying what codestreams or compositing layers are associated.

2) More ambitiously (and of unknown achievability in practice), arrange that the movie’s codestreams (frames) also be configured as an animation, one with a single composition layer that fills the frames opaquely and animates at a uniform rate. Additional boxes to support this are shown

in Table III.1-2. This is chiefly attractive only if the labeling of static or moving regions within frames are of interest (discussed later).

As Annex M discusses, the multiple images within a JPX file don't have to be contiguous (unlike with JP2), an important consideration for pairing with video and audio interleaving. Thus, in theory at least, it would seem a JPX+MJ2 file could be cleverly arranged internally so that all frames could enjoy annotations. Appropriate tagging is needed in the Reader Requirements box (e.g., "animated, but first layer covers entire area and is opaque", "codestream is fragmented such that fragments are all in file and in order").

Table III.1-2. Additional JPX Boxes Related to Multiple Images. Note that a "baseline JPX" reader (with brand "jpxb") need render only the first layer of the first image. Thus, a reader for animation goes beyond the baseline.

Name [& box type]	How many	Content
Composition ['comp']	1 [see note]	This top-level superbox must be present when animating or otherwise compositing multiple images. It contains a child box with options for the whole composition (height, width, and number of times to loop it), then a series of "instruction set" boxes, one for each composition layer. Each "instruction set" is a series of instructions in time order (plus an overall repetition count and time-increment definition). Individual instructions can set the layer's origin, size, cropping, and duration.
Codestream Header ['rreq'].	1 per code-stream	Similar in content to the JP2 header (which is no longer mandatory but if present defines defaults for optional aspects of codestream headers). Can contain optional Label, ROI Description, and IPR boxes.

Utilize MPEG-7 Scene-Level Descriptors through Part 12 Hybridization?

The MPEG4 file format supports A/V synchronization with MPEG7 scene descriptions. In MPEG4 (unlike MJ2), MPEG-7 is not just a header; it is also one or more streams. Could MJ2 be hybridized with MP4 through the common ISO Base Media File Format (Part 12) to get a similar feature? More about this next.

Add New Scene Description Track Type?

Another similar approach would be to add a new descriptive metadata track type to the MJ2 specification, analogous to the ones found in MPEG4 and QuickTime.

For example, Foessel [Foessel] suggested, in considering MJ2 for Digital Cinema, a new type of MJ2 'trak' box (called 'cdsc' for 'Content Description'), with an internal pointer to the appropriate video 'trak' and an external pointer to a separate MPEG-7 file.

Turning to sibling formats, in MPEG-4, MPEG7 is not just a header object, but one or more streams. Linking from a MPEG7 stream (or other kinds of metadata stream) to the content described is done with named links of 'cdsc' track reference type ("content describes"). To better accommodate possible edits to the content, this linking is indirect, via a table reference in the MPEG7 stream header, following a recommendation of Apple [Singer].

There is a faint whiff of this in Part 12 2nd edition. The description of the Track Reference Box 'tref', by which one track can reference another, is primarily described so that a 'hint' track (to support streaming-over-network protocols, e.g., RTP for the internet, MPEG-2 transport for broadcast) can reference the audio and video tracks it refers to. However, the tref definition mentions that an alternative enclosing track type to 'hint' is 'cdsc'. The latter is not otherwise defined. Also, within the Media Information Box, there is a new, fourth Media Information Header box defined (beyond those for video, audio, and hint), called Null Media Header Box, vaguely described as "streams other than visual and audio may use a null Media Header Box".

Another section of Part 12 2nd edition describes "AVC Extensions … which are more generally applicable" beyond just technical support for MPEG4/AVC. This describes a new way to group audio and visual samples, using one (or more) pairs of tabular Sample To Group and Sample Group Description boxes. The latter is underspecified, but is designed to hold, not boxes, but an array of fixed-length items, or an array of lengths and variable-length items. Perhaps Sample Group Description could hold metadata strings.

More insight about metadata streams can be found by turning to the MPEG-4 specifications, as further explored in Appendix 2.

As for QuickTime, note that, while its API defines a constant 'cdsc' (as a "sequence grabber atom" called "sgChannelDescription"), this may not be pertinent. See the "QTMetaData…" functions for more [QT API]; there are also a few words about such QT streams in Appendix 2.

Utilize MXF's Descriptive Metadata Scheme 1 (DMS-1)?

It is beyond the scope of this document to describe MXF in any detail. Suffice it to say that it spans a range of possibilities, from single self-contained files to multi-file, multi-presentation ensembles, and with various degrees of editing and assembling. A simple MXF file is typically composed a header (including metadata), an index table, then interlaced streams (like MJ2 chunks). A file might also be split up into coarser "partitions", each with its own metadata and index. Viewed logically, the latter supports a Descriptive Metadata stream. Furthermore, there are three types of "packages", each with its own timeline and metadata: "source package", with historical annotation, "file package", holding the actual raw essence and its input timeline for the file, and "material package", with the output timeline of the presentation.

Within this, DMS-1 [SMPTE S380M], also known as the "Geneva Scheme", provides three "frameworks", as mentioned earlier:

- Production (as a whole)
- Clip (scenes from a given source – somewhat like MJ2's "track")
- Scene (with frame synchronization)

Each framework is an organized hierarchy, build from the more than 1000 elements and sets in the SMPTE Metadata Dictionary. It is possible that DMS-1 could be embedded in MJ2, possibly by using the 'uuid' box with KLV representation of DMS-1 structures. But, for the third case, it is unclear how synchronization with scenes might occur.

(From that point of view, it might be easier to embed MJ2 in MXF than MXF in MJ2.)

III.2. Regions within a Frame and Moving On-Screen Objects

The JPX Approach to Regions within a Frame

The JPX format allows adding text annotations to a still image overall, or to multiple regions-of-interest within the image. This is accomplished through "Label" boxes for rectangular or elliptical regions, using the boxes in Table III.2-1. There is no support for arbitrarily-shaped regions. Use of these features need to tagged in the Reader Requirements box (e.g., "contains ROI metadata", "codestreams have labels").

Presumably a dual-mode hybridized JPX+MJ2 file, that also support the animation structuring discussed previously, could offer labeling for individual frames by embedding such labels in the Codestream Headers.

Table III.2-1. JPX Boxes Related to Labels and Regions of Interest

Name [& box type]	**How many**	**Content**
Label ['lbl ']	opt	A child box of Code Stream Header, Compositing Layer Header, or Association, holding textual data (Unicode in UTF-8).
ROI Description ['roid']	0+	Identifies the location and shape of one or more rectangular or elliptical "Regions of Interest" within the image. (These Regions of Interest are independent of those marked by RGN or ARN markers within the codestream.) Can appear within: • File top-level: defines ROIs for the composed, rendered image; • JP2 Header: default ROIs for all codestreams; • JPX Codestream Header: ROIs for that codestream; • JPX Association: group with metadata.

On-Screen Objects

To annotate, say, the location of a particular character, the foregoing approach can be extended as a temporal sequence of Labels with same keyword or object ID, and moving ROI boundary. A labeling convention would need to be established for interoperability. To make such labeling work tractable for the annotator, tool vendors would need "smart annotator" (like keyframe-based colorizers, that perform dynamic segmentation).

Other Approaches

Both MPEG7 and DMS-1 streams could address this area, if the MJ2 could be extended to incorporate them with appropriate synchronization, as discussed earlier.

III.3. Timed Text – Captions & Subtitles

Here we are considering text that is not burned into the original video stream, and which we don't want to burn into the MJ2 video track. The text for captioning and subtitles may originate with the video source ("line21" EIA 608; 708B), or from a separate file. The latter is typically generated manually (but for automated approaches, see III.4).

Internal MJ2 Approaches

Core MJ2 has no designated method for caption storage, synchronization, and display (e.g., fonts). Some possibilities:

- For text storage, propose a standardized syntax and adopt one of the scene-based-metadata approaches discussed earlier. Players would have to be aware of this format and perform character rendering.

- Render the text as graphics (similar to the approach used in DVD) and composite text rectangles as a front-most layer, evidently with transparent backgrounds if desired. This could be done either within the context of core MJ2, or of JPX-style animation (but in any case beyond "Simple Profile"). This takes up more file space than the text-based approach, but arguably requires less of the player. (Any potential conflict with the JPX-animation method suggested earlier for labeling would need further investigation.)

User selection of language alternatives would require players extending beyond the current standard.

External Approaches

Synchronization with captioning text can be done from the outside the video file, and is probably the practical approach in the short term. Methods to do this include:

- Use an AVI wrapper around MJ2, with a separate AVI-level text stream (discussed later).
- Keep the text separate. For delivery (and now we're usually talking a lossy stream, perhaps from a derivative file or, more ambitiously, specialized MJ2 server), use a technique for web-based streaming playback with associated synchronized text, such as:

  - HTML
  - Microsoft's Synchronized Accessible Media Interchange (SAMI)
  - Synchronized Multimedia Integration Language (SMIL); e.g., for Windows Media Player, QuickTime
  - RealText (*.rt)
  - QuickTime Text (*.txt)

A typical third-party tool like the captioning tool in FMV Pro [SlipStream] creates a SAMI file plus .asx and .html files to put an embedded Windows Media player and caption box into a web page. Another example: MAGpie [MAGpie] can create a choice of SAMI file (for Windows Media Player), SMIL file + RealText (for RealPlayer), or an intermediate file suitable for import into QuickTime Pro (to make a QuickTime Text movie). MAGpie 1 uses Windows Media Player for previews. The more complex MAGpie 2 uses Java/QuickTime previewing. Besides MAGpie 1 features, 2 also runs on the Mac, and can do Flash captioning and audio descriptive narration (as SMIL + WAV).

There is a cornucopia of other formats for representing captioning material as text or graphics files. Often the text to be synchronized and the timing information are in the same file. Be aware that W3C has a Timed Text Working Group to develop a standard form.

It is possible to imagine that a copy of, say, the SMIL content (from the .smi file) is stored in the video header for safekeeping, and possible on-the-fly extraction by a custom server.

III.4. Content-based Indexing and Retrieval (CBIR)

CBIR is concerned with helping the user of a video-on-demand system find particular videos of interest, and then scenes within those videos. Interest is generally in fully-automated pre-indexing of content. These indices are often centralized for quick search, so it is less likely that this information need be stored as metadata within the MJ2 file. Particular aspects include:

- Speech to text or phoneme. Storage of this within MJ2 could in theory be handled like timed text.
- Speaker recognition (by voice profile or facial recognition)
- Music/genre/sound recognition
- Scene boundary recognition by audio and visual cues.
- Image characterization for search. This generally requires in-advance shape extraction and categorization
- Extraction of on-screen words

Commercial systems generally have a proprietary method of encoding this information. Among vendor-neutral representations, note that MPEG-7 has a particularly rich set of shape and sound descriptors. It was recently chosen as basis for in-depth content description/CBIR by the PrestoSpace project (a European video archive consortia, but not specifically engaged with MJ2). MPEG7 is also reported to be favored by the committee designing “JPSearch” CBIR for JPEG 2000 stills and video.

III.5 MJ2 Embedments in Other File Formats

MJ2 content can be embedded within other files, and such a wrapper can provide additional places for metadata and media content. The wrapper can be a generic one, like zip, tar, and other compression+aggregation forms. Of more interest are ones that are more video-centric, such as GXF, MXF, and AAF. Of particular note on the Windows platform, is the Microsoft DirectShow/DirectX technology supportive of the Audio-Video Interleaved format (AVI)

MJ2 in AVI, and Close Captioning Metadata
Potential ways to map closed-captioning data to the native MJ2 (core or hybridized) format were enumerated earlier, but there's no evidence any have been tried to date. Perhaps wrapping MJ2 in AVI and using an AVI-level captioning stream would work.

Microsoft's Audio Video Interleave (AVI) is a container format, that can hold a number of video and audio codecs, including MJ2. AVI contains a "fourcc" 4-character code to indicate which specific codec is used. The original wrapper format (AVI Type 1) is now mostly superceded by AVI Type 2 (sometimes called AVI2, but be aware that the Canopus editor has it's own variant AVI2.) Type 1 has the audio interleaved; Type 2 has audio both interleaved and separately appended.

Applications that use the AVI can have difficulties with filesizes greater than 2 GB or 4 GB Older FAT-based file systems also impose these limits. Since lossless encoding does create large files, this can be a concern. Modern computers with the NTFS file system (plus additional techniques as needed; see http://neuron2.net/LVG/filesize.html) should cope with larger files.

With the AVI format, one of the standard filters provided with DirectX is a "Line21 Decoder", to capture captioning data encoded in video line 21 and save it as a separate stream within an AVI file [IAMLine21]. The documentation indicates that Microsoft's AVI handling exposes a programming interface to then hide or show captioning.

In theory, this can be used in conjunction with Morgan Multimedia's MJ2 codec, that handles MJ2-in-AVI. (Will playback visualization be as a separate pane in Media Player?) There could be synchronization issues, but the requirements for closed-caption synchronization are less demanding than sound.

Other Formats: MOV, MXF
Quicktime's MOV is another yet container option.

A proposal for embedment of MJ2 in MXF (as opposed to individual JP2 frames, a la the Digital Cinema Initiative) has not yet been broached.

Full exploration of these would take a separate report.

III.6 Alternative Sound Tracks

Short video clips taken with a still camera can be annotated with a narrative description, which is more like metadata than a soundtrack. For now, we are neglecting audio metadata, as well as issues associated with switching sound tracks, e.g., for different languages, or to support descriptive scene narration for the visually impaired (a la MoPix or DVS).

Appendix 1. MJ2 File Structure

Table A1-1. Top Level MJ2 File Structure, in Typical Order.

Name [box type] count	**Content**
JP2 Family signature [‘jP ’] 1	Always first. This is really a 12-byte fixed string.
File Type [‘ftyp’] 1	Contents are a “brand” of ‘mjp2’ and minor version number. ‘mjp2’ also is included in the “compatibility brand list”, plus ‘mj2s’ if the file conforms to the Simple Profile. There may be other compatibility brands, e.g., ‘jp2’ if stills are included.
Movie Box [‘moov’] 1	Contains the presentation’s metadata. See Table A1-2. ‘moov’ normally here or at file end.
Movie Fragment Boxes [‘moof’] 0+	Provide data similar to that in the Movie Box, defining additional tracks that can extend the presentation. (The tracks themselves are in a Media Data Container.) A typical application is to make a recording session more robust against a crash, by breaking it up into multiple fragments each with its own metadata in a moof box. (These evidently are not found in the Simple Profile, so we will skip detailing this box type, other than to note that it contains “Movie Fragment Header” [‘mfhd’] and “Track Fragment Box” [‘traf’]; the latter has “Track Fragment Header” [‘tfhd’] and “Track Fragment Run” [‘trun’].)
Media Data Containers [‘mdat’] 0+	Contains video tracks, as JPEG2000 video frames, and audio tracks (and in theory hint track for streaming). These tracks have minimal or skippable headers; they are managed by Movie Box data. The Simple Profile has exactly one video track, at 30 fps or less, and at most one audio track (8 or 16-bit raw audio, 48kHz sampling or less). These are in temporal order and interleaved at least once a second. (Transformation of the result is limited to uniform scaling and/or 90-degree rotation.)
Free or Skip Boxes [‘free’, ‘skip’] 0+	Deletable or ignorable content.

Table A1-2. Contents of a Movie Box

Name [box type] count	**Content**
Movie Header [‘mvhd’] 1	Contains declarations for the presentation as a whole. Items are creation and modification times, timescale (units per second), duration, playback rate, playback audio volume, 3 x 3 transformation matrix, and a counter for the next track ID to be assigned.
Track or Stream Container [‘trak’] 1+	Each contains metadata for a single track; detailed in Table A1-3.
Movie Extends Box [‘mvex’] 0,1	This box, if present, warns the software readers that there may be Movie Fragment Boxes in this file, which should be located and their data combined with that of the Movie Box. Furthermore, it contains, for each track in the Movie Box, a “Track Extends Box” [‘trex’], that sets default values used by movie fragments, to save space.
User Data Box [‘udat’] 0,1	Contains more specific boxes, of which only a “Copyright Box” [‘cprt’] has been defined so far. There may be multiple simple-text copyright notices with different language codes.

Table A1-3. Contents of a Track or Stream Container

Name [box type] count	**Content**
Track Header [‘tkhd’] 1	Has overall information about the track. A track may be either a traditional media track, or (considered an extension) a “hint” track with packetized information for streaming protocols. Items are flags (disabled, in movie, in preview), creation and modification times, unique-to-presentation track ID, duration, front-to-back video layer ordering, playback audio volume, 3 x 3 transformation matrix (applied to the video track before that in the Movie Header), and width and height (which can apply scaling to the video track before the transformation matrix).
Track Reference Container [‘tref’] 0,1	Present if this track needs to list other track IDs it depends on. So far, used just in a hint track to reference the media hinted.
Edit List Container [‘edts’] 0,1	Contains an “Edit List Box” [‘elst’]. This provides a mapping between the track’s time line and the presentation time line. It consists of a table, where each entry has: • media-time. Start time for the edit segment, within the media; • segment-duration, in media time scale units; • media-rate (1=normal speed, 0 = dwell for the duration indicated). Starting offset for a track (i.e, delay in presentation) is specified by an initial entry that is “empty” (media-time = -1) with segment-duration giving the delay.
Media Box [‘mdia’] 1	Must contain these three items: “Media Header Box” [‘mdhd’] Overall information about the media: creation & modification times, timescale, duration, and language. “(Media) Handler Type” [‘hdlr’]. Contains codes indicating the media type (video, sound, or hint). “Media Information Container” [‘minf’]. Detailed below in Table A1-4.
User Data Box [‘udat’] 0,1	Allows one or more per-track **“Copyright Boxes”** [‘cprt’].

Table A1-4. Contents of a Media Information Container

Name [box type] count	Content
Media Header [see 3 types at right] 1+	This provides overall information, particularized by the type of track: “Video” [‘vmhd’]. Items are graphics mode for composition with other video (e.g., copy, blue-screen, alpha blend), and opcolor (RGB value to use as blue-screen). “Sound” [‘smhd’]. Has balance (of mono audio tracks in a stereo space). “Hint” [‘hmhd’]. Items are maximum and average PDU size in bytes in this hint stream, maximum and average bitrate in any 1-second window, and average bitrate in any sliding 1-minute window (similar to MPEG4 descriptor).
Data Information Box [‘dinf’] 1	Locates media info in a track. Contains one “Data Reference Box” [‘dref’], which declares source(s) of media data in track, in the form of a table, where each entry is either: - a flag indicating that the media data is in the same file as this box; - a URL to a service returning a file; - a URN, consisting of well-known name and optional URL. Indexing into this table is done by the Sample Table Box next.
Sample Table Box [‘stbl’] 1	Container for the time/space map. A MJ2 “sample” is the smallest unit with which a time is associated; specifically, it’s a single video frame, a compressed audio frame (which when uncompressed may have a number of samples in the audio sense), or a set of streaming packets. This box, a packed directory for the timing and physical layout of the samples in a track, contains exactly one each of: “Sample Description Box” [‘stsd’]. A table of codec types and their initializations, detailed below in Table A1-5. “(Decoding) Time-to-Sample” [‘stts’]. This compact table allows indexing from time to sample number. Time here is relative to track start, ignoring any edit list. Each table entry has: • the number of consecutive samples with the same time delta • that delta “Sample Size Box” [‘stsz’]. Provides framing, so the media data itself is unframed. Either a single sample size is provided, that applies throughout, or a sample count and table of sample sizes (indexed by its sample number). “Sample-to-Chunk” [‘stsc’]. In order to interleave tracks, each is broken into sequential groups of samples called “chunks”, which are numbered starting from 1. This compact table, in conjunction with the one that follows it, allow finding the chunk that contains a sample. Each table entry has the: • first chunk number (of a run of chunks) • samples per chunk • index into the Sample Description table Each entry describes a run of chunks that share the samples-per-chunk and sample description index values. “Chunk Offset” [‘stco’] This table completes the data-offset information for chunk management. It simply provides the byte-offset of each chunk from the start of the containing file. This absolute offset thus ignores any box structure, so can be used to refer to media data in files without such structure. However, self-contained MJP2 files with a Movie Box at the beginning will need to recalculate this table if the Movie Box size changes.

Table A1-5. Contents of a Sample Description Box. This is a table (with entry count) of codec types and their initializations. For the Simple Profile, a track's table is limited to a single entry. Table entries are specialized for type of track.

Name [& box type]	**Content**
Video [‘mjp2’]	Items in an entry are height, width, highest horizontal and vertical resolutions (typically of the luminance channel), compressor name (“Motion JPEG2000”), and depth code (gray, color, alpha), plus … • a **JP2 Header Box**, with color space as given in JP2 Part 1 (or sRGB YCC from Part 2) • an optional **JP2 Prefix Box**, with the shared first part of the code stream. Typically has the JPEG2000 main header. • an optional **MJP2 Profile Box**, with compatible brands. • an optional **MJP2 Field Coding Box**, with field count and field order (both temporal and storage). Field count is either 2 if the sample is interlaced, or 1 (the default if this box is absent). • an optional **MJP2 SubSampling Box**. Has two code fields each to describe horizontal and vertical subsampling of chroma with respect to luminance.
Sound [‘raw’] *or* [‘twos’]	‘raw’ format has samples as unsigned integers, ‘twos’ as signed integer. Items are channel count (1 or 2), sample size (8 or 16 bits), and sample rate. For stereo, left and right channel samples interleave.
Hint	Description is streaming-protocol-specific. Not a normative part of MJ2.

As discussed in the main text, MJ2 also draws upon:

- Part 1 (JP2 Still Image) and amendments. Individual video frames are encoded with JP2.
- Part 12 2nd ed. (Base Media File Format). The base media file format derives from QuickTime, and is the base for both MP4 and MJ2, which inherit from it (except where they don't).

Appendix 2. MPEG-4 Streams

The content here is an initial superficial scan of a very complex topic, mostly drawn from [14496-1.6] and should be considered suggestive rather than in any way definitive.

Both MPEG4 and MJ2 are based on a common base file format and set of boxes, but then diverge. A typical file in both formats has two main boxes, a "moov" box representing the main header and holding headers for individual tracks, and a "mdat" box holding the interleaved media streams for those tracks. Media can also in external files. MPEG4 defines more track types than MJ2, and provides more mechanisms for the externalization of these resources.

In particular, MPEG 4 has an "object descriptor" (OD) model. Within the moov box, as a starting point for the presentation, there's an Initial Object Descriptor box ('iods'), a header that references:

- An OD Track. This is the header for an OD stream, which evidently manages the track IDs of this file and any of its external references, as well as interactive events.
- A "Scene Description" Track. This is the header for a Scene Description stream, which defines, using the "BIFS" system described next, the composition of tracks within the file and in other files and their audio/visual objects. (In theory, the BIFS system is available within MJ2, but is not the normal default compositional system).

Overview of MPEG 4's Scene Description Framework
This is from [Guillemot], with additions in italics.

"An MPEG-4 scene is composed of media objects. The MPEG-4 dynamic-scene description framework, which defines the spatio-temporal relation of the media objects as well as their contents, is inspired by VRML. The compressed binary representation of the scene description is called BIFS (BInary Format for Scenes).

Defined in the original [ISO/IEC 14496] "Part 1: Systems", section 9, BIFS can describe 3D scenes in a compact binary format representing a pre-defined set of audio-visual objects, and each one's:

- *hierarchical position in a scene graph,*
- *temporal and spatial relationships with respect to other objects,*
- *attributes, such as 3D geometry, and*
- *behaviors.*

BIFS also deals with synchronization, updating of scene-graph nodes and their attributes, and streaming of BIFS packets.

The compressed scene description is conveyed through one or more Elementary Streams (ES).... Elementary Stream Descriptors provide information relative to the stream, such as the compression scheme used. Elementary stream data is partitioned into Access Units *during compression; for video these are individual frames....* An Access Unit *(AU), also referred to a sample,* is the smallest data entity to which timing information can be attributed. Two Access Units *in a given stream* never refer to the same point in time.

Natural and animated synthetic objects may refer to an Object Descriptor (OD), which points to one or more Elementary Streams that carry the coded representation of the object or its animation data. An OD serves as a grouping of one or more Elementary Stream Descriptors that refer to a single media object. The OD also defines the hierarchical relations and properties of the

Elementary Streams Descriptors. *The ESDescriptors themselves are a constituent of entries in the Sample Table ('stbl', also found in MJ2).*

A complete set of ODs can be seen as an MPEG-4 resource or session description. The Object Descriptors are conveyed through one or more Elementary Streams. By conveying the session (or resource) description as well as the scene description through their own Elementary Streams, it becomes possible to change portions of scenes and/or properties of media streams separately and dynamically at well-known instants of time.

The MPEG-4 Systems specification also defines a packetization of ES data into... SL packets, or SL-PDUs.... Access Units are the only semantic entities at this layer and their content is opaque. Packetization information has to be exchanged between the entity that generates an elementary stream and the sync layer...."

We note these stream/track types that are additional to MPEG4, beyond those of ISO Base:

Table A2-1. Additional MPEG4 Track Types. These are additional values for the "handler-type" in a Handler Reference Box ('hdlr'), beyond those given in ISO Base Media File Format common to MPEG4 and MJ2.

Name [& handler-type]	**Content**
Object Descriptor Stream ['odsm']	Described above.
Clock Reference Stream ['crsm']	Timestamps within an Object Clock Reference (OCR) model.
Scene Description Stream ['sdsm']	BIFS would reside here. Another example would be a font data stream.
MPEG 7 Stream ['sdsm']	Presumably this is where the MPEG 7 "scene framework" can reside.
Object Content Info Stream ['ocsm']	Tagged keywords, for search engines, whose semantics (either static or streamed) cover - • Technical description of content • Language • Content rating information • Creation dates and name of content/authors This probably will be superseded by MPEG-7.
IPMP Stream ['ipsm']	Contains "International Property Management and Protection", ISO/MPEG's international standard, implementing Digital Rights Management (DRM) for audio/video software. "IPMP" was the original version of this, "IPMP-X" a later extension. Basically a defined set of "hooks" to third-party modules, e.g., for linking to MPEG-21 REL.
MPEG-J Stream ['mjsm']	MPEG-J provides an alternative to BIFS for the creation of interactive content, allowing programmatic control with Java.

Each MPEG4 stream is associated with an MPEG4 Media Header Box (within its 'trak' header), but for the foregoing streams in the table, there is no additional information to be stored, so a null box 'nmhd' is used (but specific '*xx*hd' box codes for each stream type are defined for future use if needed).

MPEG4 streams can synchronize to an external clock with an “object clock reference” (OCR)

Cross References between Tracks [mp4ra]
Tracks that are related are linked by “track references”. The link is typed with a code, a “track-reference type”. The comparable QuickTime links are also shown.

Table A2-2. ISO Family Track Reference Types. All are defined in MPEG-4 v1 specification except as noted.

Code	**Meaning**
`avcp`	AVC parameter set stream link. Defined in AVC spec.
`cdsc`	This track describes the referenced track. Used by scene description tracks.
`dpnd`	This track has an MPEG-4 dependency on the referenced track
`hint`	Links a hint track to original media track. Defined in ISO Base spec.
`ipir`	This track contains IPI (Intellectual Property Info) declarations for the referenced track
`mpod`	This track is an OD track, which uses the referenced track as an included elementary stream track
`sync`	This track uses the referenced track as its synchronization source.

Table A2-3. QuickTime Track Reference Types. All are defined in the QT specification.

Code	**Meaning**
`chap`	Chapter or scene list. Usually references a text track.
`scpt`	Transcript. Usually references a text track.
`ssrc`	Non-primary source. Indicates that the referenced track should send its data to this track, rather than presenting it.
`tmcd`	Time code. Usually references a time code track.

In conclude, we note that, for better or worse, MPEG4 has elaborated many more possibilities than MJ2 with respect to streaming or transport mechanisms, application profiles, and types of codecs.

Appendix 3.
Universal Labels (UUIDs and UMIDs) and KLV in MXF

SMPTE has defined “Universal Labels” (298M), of variable size, in increments of 4 bytes; the second byte encodes the size. Of particular interest are those for which SMPTE has created a registered collection (in the SMPTE Metadata Dictionary – RP220), which are specific UUIDs.

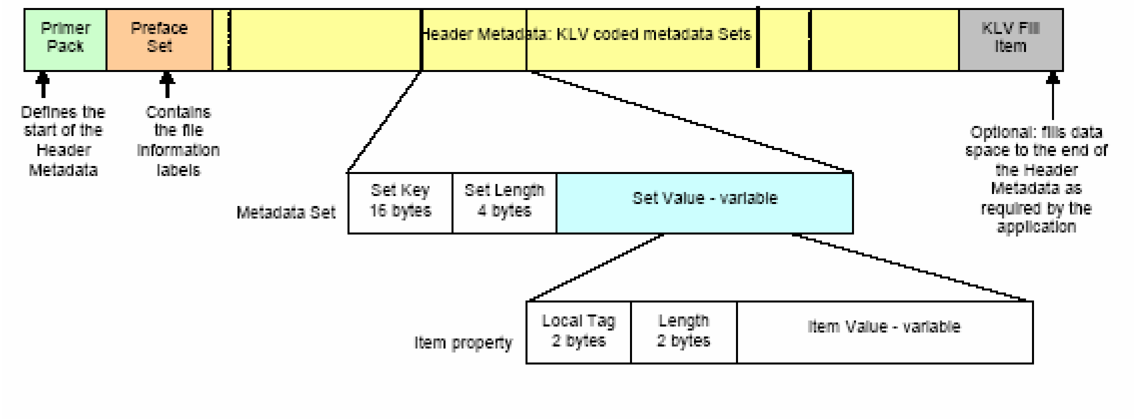
These UUIDs are used in the MXF archiving file format. They are also used for user data in the AAF authoring format, where they are called AUIDs. (For structural essence metadata, such as identifying program packages and clips, there are also longer unique identifiers called Universal Material IDs (UMIDs). Defined in [330M] [RP205], these come in 32-byte Basic and 64-byte Extended forms.)

The overall structure of a MXF file is a sequence of Key-Length-Value (KLV) elements [SMPTE 336M]. For defined metadata elements (as well as undefined “dark” ones), the Key is the appropriate UUID. The size of the Length field can be varied to accommodate content.

Within the file, every defined set or “pack” of metadata has a UUID, as does every item and property within the set. A typical property is a file-specific Instance ID for the set (which could be a UUID or UMID).

For compactness, the items and properties are referred to within a “Local Set” by a 2-byte tag, which can be mapped to a full UUID by a “Primer Pack”, a table in the given file partition. Some Local Tags are statically defined for MXF; others are dynamic. The local lengths are usually 2-byte as well. MXF sets are not physically nested beyond inclusion of their items; deeper logical nesting can be done by cross-referencing of instance IDs.

Figure A3-1. Typical MXF Header Metadata Structure, from [377M], figure 9.


Primer Pack
Preface Set
Header Metadata: KLV coded metadata Sets
KLV Fill Item
Defines the start of the Header Metadata
Contains the file information labels
Optional: fills data space to the end of the Header Metadata as required by the application
Metadata Set
Set Key 16 bytes
Set Length 4 bytes
Set Value - variable
Item property
Local Tag 2 bytes
Length 2 bytes
Item Value - variable

Import/export of KLV metadata to a database is usually done through an XML representation, using this logical nesting. (A commercial mapper is available from MOG.) The default MXF descriptive metadata set, DMS-1, is discussed elsewhere.

References

[BDEI] www.cs.umb.edu/cs639/BDEI/BDEI_Landscape.ppt; by Bob Morris? See Morris's paper about metadata at http://wiki.cs.umb.edu/twiki/bin/view/BDEI/DraftsForChania

[Colyer, Greg, Kats Ishii, Jane Hunter, "Mapping between Dublin Core and JPX metadata", Oct. 15, 2002, ISO/IEC JTC 1/SC 29/WG 1 N2736, www.mpeg.org/public/wg1n2736.pdf]

[dig35] www.i3a.org/i_dig35.htm.

[Edwards] Eric Edwards, Siegfried Stoeffer, "JPEG 2000 for Digital Cinema Applications", April 25, 2001, www.jpeg.org/public/DCINEMA-v2.pdf.

[Foessel] Siegfried Foessel, "Motion JPEG 2000 and Digital Cinema", presented at the SPIE Conf. Visual Communication and Image Processing, Lugano, Switzerland, 2003. www.jpeg.org/public/DCINEMA-JPEG2000.ppt

[Guillemot] Guillemot, C., P. Christ, S. Wesner, A. Klemets, "Internet-Draft - Payload Format for MPEG-4 Streams", March 2000. draft-ietf-avt-mpeg4streams-00.txt

[IAMLine21] "IAMLine21Decoder Interface", Microsoft DirectX 9.0 SDK Update (Oct. 2004), http://msdn.microsoft.com/library/en-us/directshow/htm/iamline21decoderinterface.asp?frame=true.

[ISO/IEC 14496] "Information Technology – Coding of audio-visual objects", Multiple parts. Version 1, January 1999. Now at Version 2.

[ISO/IEC 14496-1.6] "Text of ISO/IEC 144496-1:2001/FPDAM.6", Singer, David, William Belknap, Guido Fanceschini (eds), ISO/IEC JTC1/SC29/WG11/MPEG01/N4856, May 16, 2002.

[ISO/IEC 15444] "Information technology – JPEG 2000 image coding system". Multiple parts; most are available for purchase from International Standards Organization, but some parts are free at http://isotc.iso.org/livelink/livelink/fetch/2000/2489/Ittf_Home/PubliclyAvailableStandards.htm

[ISO FDIS 9834-8] "Information technology – Open Systems Interconnection – Procedures for the operation of OSI registration authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 object identifier components", joint ISO/IEC FDIS 9834-8, ITU-T Rec. X.667, draft, Sept. 2004.
[jpx xml] http://www.jpeg.org/public/15444-2.dtd and .xsd

[Jung] Jung, Klaus, "Including GML data based on the OpenGIS standard in JP2 files", March 11, 2003, ISO/IEC JTC 1/SC 29/WG 1 N2887, www.jpeg.org/public/wg1n2887]

[Jung] K. Jung, S. McPartlin, Algo Vision LuraTech GmbH, "Request for UUID Registration of IPTC Information (Revised)", ISO/IEC JTC 1/SC 29/WG1 N2600, www.jpeg.org/public/wg1n2600.doc

[Lee] Lee, Daniel T., "JPEG 2000: Retrospective and New Developments", Proc. IEEE, 93 (1), Jan. 2005, pp. 32-41.

[MAGpie] Media Access Generator, WGBH National Center for Accessible Media. Free from http://ncam.wgbh.org/webaccess/magpie/ See also: http://cita.rehab.uiuc.edu/magpie/; www.webaim.org/techniques/captions/magpie/

[MIX] Annotated MIX schema, US Library of Congress, www.loc.gov/standards/mix/mix.xsd

[mp4ra] See track descriptions for associated file formats at the MPEG4 Registration Authority, www.mp4ra.com/trackref.html

[NISO] 2005 draft, successor to "NISO Z39.87 Technical Metadata for Digital Still Images", National Information Standard Organization.

[MP7] For a useful synopsis site, see http://xml.coverpages.org/mpeg7.html.

[QT API] http://developer.apple.com/documentation/QuickTime/Conceptual/.... In particular, for QTMetaData functions, see QT7UpdateGuide/Chapter02/chapter_2_section_9.html

[RLG] ”Descriptive Metadata Guidelines for RLG Cultural Materials., www.rlg.org/en/page.php?Page_ID=214, www.rlg.org/en/pdfs/RLG_desc_metadata.pdf

[RP205] “SMPTE Recommended Practice RP 205 – 2000 – Application of Unique Material Identifiers in Production and Broadcast Environments”, 2000.

[RP210] “SMPTE Recommended Practice RP210 – 2004 – SMPTE Metadata Dictionary”, version 8, Aug. 2004, www.smpte-ra.org/mdd/. See also “SMPTE 335M-2001 - SMPTE Standard for Television - Metadata Dictionary Structure”.

[Singer] Singer, David, “Proposed linking of MPEG-7 streams to content streams“, ISO/IEC JTC1/SC29/WG11/MPEG2001/N4857 May 2002.

[Slipstream] Slipstream Systems, Ltd, FMV Pro ($20). www.f-m-v.com (Captioning wizard was previously separate, as “Caption-It” from easycaption).

[UUID ISO] UUIDs can be generated by the algorithm in “Information technology -- Open Systems Interconnection -- Remote Procedure Call (RPC)”, ISO/IEC 11578:1996.

[330M] “SMPTE Standard 330M-2000 for Television – Unique Material Identifier (UMID)”, January 31, 2000.

[377M] “SMPTE 377M Proposed Standard for Television – Materials Exchange Format (MXF) File Format Specification”, 2003. www.smpte.org/standards/pdf/s377m.pdf