"Streaming Metadata, Applications and Challenges"

Robin Rowe, CEO and Chief Technologist
MovieEditor.com
820 West G Street, Ste. 418
San Diego, California 92101
619-232-5880
Robin.Rowe@MovieEditor.com

Abstract

Metadata is data about data. Streaming metadata can be embedded in a television transport protocol such as MPEG to enable new types of viewer experiences such as interactive television. SMPTE Working Group W25, Metadata and Wrapper Technology, is defining standards for digital streaming metadata. New standards are needed because without them new forms of data or even existing analog data like closed-captioning (which is standardized in VBI for analog television) is encoded in proprietary ways in otherwise standard digital video streams (such as MPEG-2).

We will explain why metadata is necessary, how it works, and why it is different from conventional databases. We will describe some of the features that make the metadata approach taken by SMPTE more powerful and flexible. We will discuss some lessons learned from creating BigScreen, a Windows software video player that supports metadata.

Why Metadata?

Metadata is the underlying technology to enable consumers to make purchases, perform searches, find related information, and do other data-related things we would like to do with video. The most prevalent real-time streaming data protocol today is probably SMPTE timecode. Television data more familiar to consumers in the U.S. is closed-captioning or subtitles and teletext in other countries. Analog streaming data is embedded in the television picture vertical blanking interval (VBI). But, digital television data is not constrained to fit within VBI and can be much more flexible.

Digital metadata offers capabilities far beyond that of data in analog television. For instance, consumers might want to buy an article of clothing being worn by an actor in a TV show (using interactive TV). The information of who makes the shirt and where to buy it are data, the organization of that data in the television transport stream is metadata.

As a point of reference, figure 1 shows our notion of a typical future utilization of metadata, including authoring, transmission, and reception. The metadata is authored on a laptop using software that accesses video created in a conventional edit suite. From there the metadata is broadcast to consumers as part of the video signal. Consumers need video software on computers or metadata-enabled set-top boxes to be able to use the authored metadata. They might click on the image of an actor to have a browser take them to a Web site, or they might click on an article of clothing to pop up a home shopping menu on a set-top box.

Assuming we start with a typical video edit suite, the metadata authoring process seems more akin to adding closed-captioning than like writing television news scripts. Content tends to lead metadata in that you generally prefer having something to tag the data to before you tag it.
Like both closed-captions and scripts, metadata seems more of a production process than an editing process. In our illustration edited clips reside on a server that is accessed by production or administrative staff overlaying metadata using a PC. With a data network there no requirement that the metadata actually be authored in the same facility. Like closed-captioning, metadata could be created by operators working at home. The metadata is then embedded in the broadcast stream (by master control) or could be a separate stream operating in conjunction with a streaming video Web site or cable head-end.

At the receiver a viewer clicks on something in the picture to cause some action to happen, to view a Web site, to buy something. How that works will depend on what manufacturers chose to do in designing set-top boxes, receivers, PC software video players, and even mobile phones. The same metadata may create a different experience depending on the device you view it on, but should not disrupt legacy devices that don't understand metadata or don't know what to do with a particular piece of metadata.

What's So Special About Metadata?

Many of us use relational databases, such as Microsoft Access. A relational database organizes data into rows and columns. Unlike a spreadsheet, the field structure is rigidly set. Each field (column) is labeled and accepts only a particular type of data such as text or a number. This data consistency enables users to perform powerful queries that span all the rows in a table. The data layout of the table is called the schema. A schema is a static form of metadata. It describes the data in the table. Figure 2 shows a sample schema and one-row table created in Access.
What isn't obvious in figure 2 is that these fields are not the same type even though they all say "text" in the schema. When creating a text field in an Access table you must specify the maximum width of the field, in the case of the [First Name] field we set it to 20 characters. Once you specify a field of 20 characters that field consumes 20 characters in every row in the table, even where the field is blank.

Database administrators go to considerable effort (and frustration) attempting to keep table sizes reasonable by not making the fields too big. However, at some point someone with a really long name, for instance, won't fit into the name field of your table. You must restructure the table to make the field wider. When this happens the table is taken off-line while it is copied into a new table with a larger structure.

Streaming metadata can't work that way. We couldn't possibly know ahead of time what the length of the data in a field might be. Television is too dynamic for that. Also, the empty space wasted in a conventional database would be expensive to transmit and store. And, with a standardized table structure, that waste would become astronomical.

Consider a table that contains virtually every field anyone could ever think of to want in a video stream. The advantage, of course, would be that if you needed to look something up in that data you could know exactly the correct name of the field to look for. Everyone could be consistent in how they named their fields because the fields are all standardized. All data streams would be compatible. Such a master schema would, however, have hundreds or thousands of fields. Most of those fields would be blank in any particular context. Nobody would be inclined to use all those fields at once. However, even an empty data stream would be huge because of the space taken by blank fields.

SMPTE has taken a smarter data encoding approach called KLV. That stands for Key-Length-Value. Data elements are triplets of the field name, the length of one field, and the field data. Each KLV triplet stands on its own. There is no separate schema, rather a constant stream of dynamic metadata. If instead the schema was transmitted separately during a broadcast, say at the top of the hour, changing the channel would produce gibberish. Channel surfers would be locked out of interactive data. KLV is an efficient dynamic data stream that solves all these problems.
In figure 3 we see a graphical representation of how KLV is laid out in the video stream. Video transports such as MPEG interweave chunks of video and audio. Another chunk type is metadata. Each metadata chunk contains a GUID, a globally unique identifier that enables you to tell exactly what video stream you are looking at that the metadata belongs to. The GUID is generated statistically, avoiding the logistical problem of having a central key authority to assign unique ids. An authoring program creates a GUID automatically by hashing together the time of day, the network card serial number, and other statistics specific to a machine. After the GUID is a key count to let the reader know how many KLV triplets follow. The length of each KLV triplet will vary.

SMPTE has also done something special within the KLV keys. Recall that we would like to define a master schema so that all users will hold to the same field names thereby providing maximum interoperability of data fields. Using KLV instead of a separate schema makes that practical. However, humans browsing a database containing hundreds or thousands of standard fields would be overwhelmed. If you had a relational database with that many fields you would need a search engine just to navigate to the right column. SMPTE metadata uses a field hierarchy approach to neatly eliminate this problem.

Each key works like a dotted sub-net address. By breaking the field namespace into a tree structure with top categories like origination, editing, legal, and so forth we can quickly drill to the appropriate field even in a huge schema. In our simplified example key 1 is the street address of the person who shot the video. That would be [Origination.Name.Address.Street].

Don't become confused into thinking that the data itself is a hierarchy. The data is kept relational in rows and columns to maximize relational query compatibility. Only the field names are in a hierarchy.
With a conventional database you aren't allowed to have two fields with the same name. A consequence is that fields containing the same type of data lack consistency in naming. For instance, although a table might have many different fields that contain someone's last name, each field would be named slightly differently as [Producer Last Name] and [Editor Last Name] and so forth. Using hierarchical field names is much more elegant than that. You can have more informative field names such as [Origination.Producer.Name.Last] and [Editing.Editor.Name.Last]. Each field is still unique as a whole, but contains the same sub-name for the same type of data.

Using sub-netted hierarchical field names enables powerful multi-field queries. You may query something like <SELECT ANY Name.Last> to search for any last name field that has data in it. This is important since using a standard schema implies hundreds of fields, but in practice those fields are mostly blank. (Recall that blank fields incur no overhead using KLV.) Being able to easily make a spanning query like this empowers users to quickly target the relevant fields that contain data. If you need to quickly find any phone number you have for some metadata video stream you can simply say <SELECT ANY Phone#>. You don't have to browse. You can get the answer directly. In a conventional database there is no concept of spanning like fields this way.

The top-level field name component is the *namespace*. As we've already established, it is highly desirable to have standardized field names so users can access and interchange data easily. But, how do you know if you are looking at a SMPTE standard field name or some other custom (proprietary) field name? The answer is to use a namespace, to call the first field name component SMPTE.

In figure 4 we see a field hierarchy that contains the fields [SMPTE.GUID.Copyright.Primary Holder.Phone#] and [SMPTE.GUID.Origination.Camera Operator.Phone#]. Note the use of namespace, GUID, and duplicate field name components. On the transport any individual metadata chunk of KLV triplets could be implemented to have the same namespace and GUID thereby reducing overhead if the field names are transmitted as text. The implementation could implicitly prepend the namespace and GUID when reading the KLV triplets. Another approach is to dispense with text field names within standard namespaces and instead use an index into the standard data dictionary.

**SMPTE Metadata Standards**

We've been discussing SMPTE metadata as though it is already standardized and lacks controversy. That's not really the case. The standardization process is ongoing. We've taken liberties to make it easier to describe the metadata concepts involved, and we've taken little care to avoid introducing our own thoughts about how to implement those concepts. For instance, using dot notation as an implementation for hierarchical field names is an idea we adapted from how TCP/IP sub-nets subdivide a network topology. And, spanning sub-field queries is our idea.
Those who want a deeper understanding or to participate in the SMPTE metadata standardization process should join the W25 Working Group as Observers or Members. That can be done from the SMPTE Web site at www.smpte.org. Doing so will put you on the email reflector that provides discussion of the standardization draft documents.

<table>
<thead>
<tr>
<th>SMPTE 335M-2000</th>
<th>Metadata Dictionary Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMPTE RP210.2-2000</td>
<td>SMPTE Metadata Dictionary Contents</td>
</tr>
<tr>
<td>SMPTE Egxxx-2000</td>
<td>Node Structure for the SMPTE Metadata Dictionary</td>
</tr>
<tr>
<td>SMPTE 336M-2000</td>
<td>Data Encoding Protocol using Key-Length-Value</td>
</tr>
<tr>
<td>SMPTE RP213-2000</td>
<td>Format for Non-PCM Audio and Data in AES3 – KLV Data Type</td>
</tr>
</tbody>
</table>

**Metadata Implementation**

BigScreen is a Windows software video player we are implementing with support from the BBC and other organizations. It is not presently a general purpose video player like the Microsoft Windows Media Player or the Real Networks RealPlayer. It is optimized for specific types of motion imagery content, such as high definition fly-overs of satellite imagery or maps. Its purpose is to enable users to view motion imagery at high definition over the Internet on PCs without picture artifacts.

![BigScreen Player with Metadata Regions Visible](image-url)
In implementing hierarchical KLV metadata our goal was not to be compliant with the SMPTE metadata standard (which is still a moving target) but to embody the concepts necessary to support that paradigm. We want to be able to support whatever standards evolve.

This video software plays motion imagery full screen on a PC, usually at 1024x768 pixels and 50fps although it is resolution and frame rate independent. The video is motion imagery constructed from a scene-graph. Unlike the scene-graph version of MPEG-4, we are not attempting to deconstruct raster video into individual objects. Rather, the user must author using the software to construct the scene-graph from individual components such as images (textures), metadata, and actors (manipulators). Actors work similarly to key frames in a DVE (television digital video effects box). They move (fly) the images based on time. Unlike a DVE, our source images are not typically motion video, but stills.

In figure 5 you can see a satellite fly-over of Washington D.C. that contains two metadata objects, one being the Washington Monument. Because the video implementation is a scene graph, the metadata hangs off of individual objects rather than simply being embedded into the video stream. The object in the case of the Washington Monument is not the monument as you might expect, but the entire image (texture) that is sliding by in the full screen playback window. However, it doesn't work that way in all cases. If we had a separate sprite helicopter in this scene flying over the monument we would be able to attach metadata to just that sprite. The helicopter sprite would in turn hang off of the background image in the scene graph. The metadata attaches to individual objects (typically moving still images) but can be specified, as shown here, to apply to a particular bounding box region.

Users author metadata by pausing video playback and selecting an object and region to attach the metadata to. Because the implementation is a scene graph, not raster video, the metadata implicitly stays with the region tagged as it moves. We don't face the challenge of creating a metadata region that must follow an object embedded in a moving raster. The metadata can be anything. It may be the latitude, longitude, and physical dimension of an image. Or, it could be a hyperlink that would launch the user's browser to a related Web site.

At this time we are not trying to make users conform to a standardized field namespace, but permitting them to add any fields they wish. And, within reason they may attach any length of data to a field. Internally, our KLV implementation doesn't actually encode data length as a parameter, but instead uses null-terminated text strings and calculates the length. We have also taken the liberty of introducing our dot notation for field hierarchy so that any field name can be implemented as a simple text string as well. These variations on SMPTE KLV made implementation easier in C++ software.

**Conclusion**

Metadata is data about data, what type of data a structure contains and how it is laid out. Unlike a fixed field schema in a conventional database, streaming metadata is in constant flux. Streaming metadata embedded in a television transport protocol enables new types of viewer experiences such as interactive television.

Streaming metadata receivers, such as television sets, can't know the field structure (schema or metadata) in advance. As you flip through the channels the data layout will change. Each channel, and even each program, will want to offer different data fields depending on content. Since a TV receiver won't know what data to expect, metadata tells it what fields are there. For interoperability, we want a standardized data field namespace. With hundreds of fields in such a standard, a field hierarchy (a tree layout) helps avoid user confusion. And, a statistically assigned Globally Unique ID makes it simple to tell apart similar video streams.
Digital TV sets will need new capabilities in order to use metadata, just as changes were required to enable closed-captioning in analog receivers. Key to receiver manufacture is having a standard. It is important that with metadata, unlike say closed-captioning, the data stream is extensible and customizable without breaking the standard. Using metadata, rather than a rigid field layout, accommodates changes. We can minimize future TV receiver obsolescence.

Conventional database engines don't take kindly to variable length fields, changing field layouts, or hierarchical field schemas laid out as trees. SMPTE KLV requires a new type of data engine designed for streaming metadata.

Robin Rowe is a partner in MovieEditor.com, a technology company that creates Internet and broadcast video applications. He writes a monthly column about multimedia for Linux Journal. He has written for Dr. Dobb's Journal, the C++ Report, the C/C++ Users Journal, Data Based Advisor, and has many published conference papers.

His software designs include a client-server video editing system in use at Manhattan 24-hour broadcast television news station Time Warner New York One and associated Web site www.ny1.com, and an automated television news monitoring system developed for DARPA and the Pentagon. He has taught C++ at two universities and designed video software in Fortune 500, defense, and academic environments. He was a technical director of broadcast news at NBC WICD-TV and helped build the robotic studios at NBC WMAQ-TV Chicago.

You can reach him at Robin.Rowe@MovieEditor.com.

BigScreen is a trademark of MovieEditor.com. Windows and Windows Media Player are trademarks of Microsoft Corporation. RealPlayer is a trademark of Real Networks. Star Trek (7-of-9) is a trademark of Paramount Pictures.