The Problem with Timecode

Robert Loughlin, Sr. Technology Specialist, Innovation

Frame.io, an Adobe Company

Derek Schweickart, Sr. Workflow Architect, Innovation

Frame.io, an Adobe Company

Written for presentation at the

SMPTE 2021 Annual Technical Conference & Exhibition

Abstract. *Timecode as it exists today is arbitrary and imprecise, and conflates its role of identifying an individual media element and functioning as a time clock. As we move more and more into file based, cloud first workflows, we need a new time labeling standard that empowers and achieves the same level of complexity that advancements in our video and audio pipelines do.*

To address this, we evaluate a number of glaring problems with timecode to understand what problems need solving, and what is required in a new standard.

A new standard needs to be extensible and designed to work inside of a file instead of a video signal. A new standard also needs to address all samples — video frames, audio samples, and data samples — while also avoiding the limitations of a 24-hour clock. Additionally, it should allow for layers in order to track manipulation through a pipeline. A frame or sample should be able to be identified from a delivered file back to its original source.

The creation of a precise time labeling system that allows us not only to provide and accurate sense of when a sample was created, but also by whom and what it is, is required to empower the next generation of digital workflow and creative tools.

Keywords. Timecode, Digital Cinema, Cloud, Post Production, Production, UTC, time label

Why do we have timecode at all?

We take it for granted, but what's the big deal with timecode anyway?

Timecode, as it exists today, is a core element of video and audio signals, video and audio files, and video and audio workflow and is often associated with its use as a synchronization tool. It's simple, effective, and virtually universal — so why change it? Before we answer that, let's take a small step back.

Before timecode, film editors used Edgecode (Kodak, 1919)[1] to identify individual pictures on a film reel. This was effective while the media production world relied exclusively on celluloid. With the increased use of videotape in the 1950s, editorial processes and workflows that relied on Edgecode became more of a challenge because there was no way to visually see what frame you were on when trying to make an edit. Many solutions, like using audio tracks to carry sync pulses, were developed to assist with this.

At this point, audio tracks recorded synchronously were used to drive video editing; there was, as of yet, no accepted standard to identify video frames. Different technologies competed for many years before the first Timecode standard, SMPTE ST-12, was proposed in 1970[2]. It was later approved in 1975.

This was timecode's original purpose: To identify a certain frame within a piece of time-based video media. In effect, timecode is an address we can apply to a specific frame in order to identify that frame in context. This was formatted as an arbitrary time reference (HH:MM:SS:FF), but did not itself represent actual time — it was simply a label.

However, because timecode looks like a clock, its most common use case is as a synchronization tool. If two discrete devices set their timecode values to the same "clock", then their labels for frames or audio samples can run in synchronization — therefore enabling media from these two devices to be reassembled and placed back together in context in post-production. But this is the most fundamental issue with the way we use timecode today: Timecode is not time.

Over the years, Timecode was modified here and there to include some more information like user bits but has virtually remained the same. SMPTE ST-309 was introduced in 2012 to standardize a way to include date and timezone information in user bits[3], however this had little impact on production workflows.

Since ST-12 was designed to be able to be encoded onto video tape, it had to be standardized, consistent, and, importantly, have a small bit footprint. It was also designed with the concept of a "tape" or "reel" in mind. As we bring the use of timecode into the digital, file-based world, we start to see how this becomes limiting.

The time is now

Now that the majority of media creation is completely file based, many of the initial constraints around what timecode could and couldn't do are no longer present. Files can carry large amounts of additional data and metadata. Files, in the sense of acquisition (i.e., takes), are discrete entities with specific bounds and identifiers. Files can also be aware of where and when they were created.

Additionally, media creation today is increasingly distributed, and cloud based, widening the context within which the media is created. As global networking continues to leap forward, traditional means of video and audio transmission will be replaced by entirely file and IP transmission.

Both of these highlight areas where timecode's original context and purpose makes it insufficient for the workflows of today and of the future — specifically, as a means to locate a specific unit of an asset (i.e., a frame) as well as identify that asset in time, timecode is arbitrary and imprecise.

We need to address the need for an updated standard using the benefit of a fully digital, cloud-powered pipeline. While much work has been done to develop a new approach to image science in a fully digital world, little has been done with regard to the fundamental work of managing time-based media.

Wireless data protocols are getting better and faster every day. In the near future, moving video around as a video signal will be rare. Instead, video will be transmitted as packets of data over network transport protocols. These packets may vary in size, based on the limits of a given protocol, but we can assume that they will have enough headroom to ascribe much more data to describe the time event that is being recorded. With this additional data, we can track more information to empower modern workflows and allow media creators to have a clearer understanding of what moment of time their media describes and how it fits in context. This also opens the door to allowing media creators to trace individual frames back through generations of editing, duplication, or other manipulation.

If we rethink the way we record and describe a moment of time, like we've redesigned the way an image is recorded, we begin to understand how this doesn't just give us more information. It can create an entire pipeline. Streaming video can be traced back to its source — a specific moment in time recorded in a specific space. Given the rapid advancement of AI-driven technologies like Deep Fakes, the ability to maintain the veracity of a piece of media may even become a matter of national security. Since we can record time as data instead of a stream, it can even be encrypted.

The problem of size

Let's look at the fundamentals of media creation whether film, video, or audio. At its core, the creation of time-based media is the process of freezing multiple moments of time at a certain rate within certain boundaries (start and end time).

Even though a unique timecode value seems to indicate a point in time, it actually represents a time range. A timecode value of **03:54:18:22** is a single label applied to almost 42 milliseconds of time (at a framerate of 24fps).

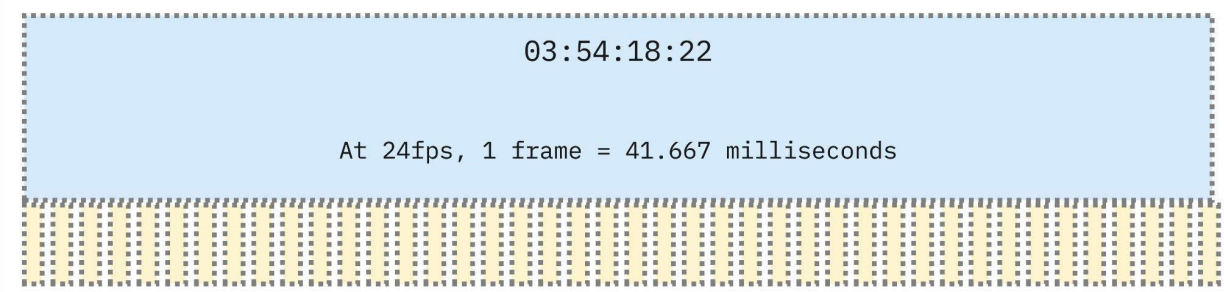

Figure 1. A single timecode value (i.e., a single frame), is about 42 milliseconds long at 24fps.

One of the most common cases of varying range sizes is when syncing picture to audio. The time range of a video frame is much longer than the length of an audio sample. At 24fps with 48KHz audio, there are 2000 audio samples for each image and, therefore, each individual timecode value. So, while timecode is generally controlled on a production by the sound mixer, the timecode values are not adequate for identifying sound samples. In fact, this is why sound recorders generally start their recordings at whole second intervals, which is very useful for synchronizing. However, when working with sync in post, sub-frame adjustments often need to be made. Most often, this is referred to as “perf-slipping,” which is a way of nudging the audio by the time range of a film perforation (typically either 1/3 or 1/4 of the duration of a frame).

While timecode in this context is too coarse or too broad to accurately label points in time, it also has a limitation of 24 hours before “rolling over.” Once the 24-hour mark is hit, timecode values start over at **00:00:00:00**. This results in having a fixed and limited amount of unique timecode values. At 24fps, there are only about two million unique timecode values available in a 24-hour period. For example, if a production is using time of day timecode for video an audio, files recorded on Day 1 will have the same timecode values of files recorded on Day 2, even though they refer to two different points of time. So, on this end, timecode is also too narrow.

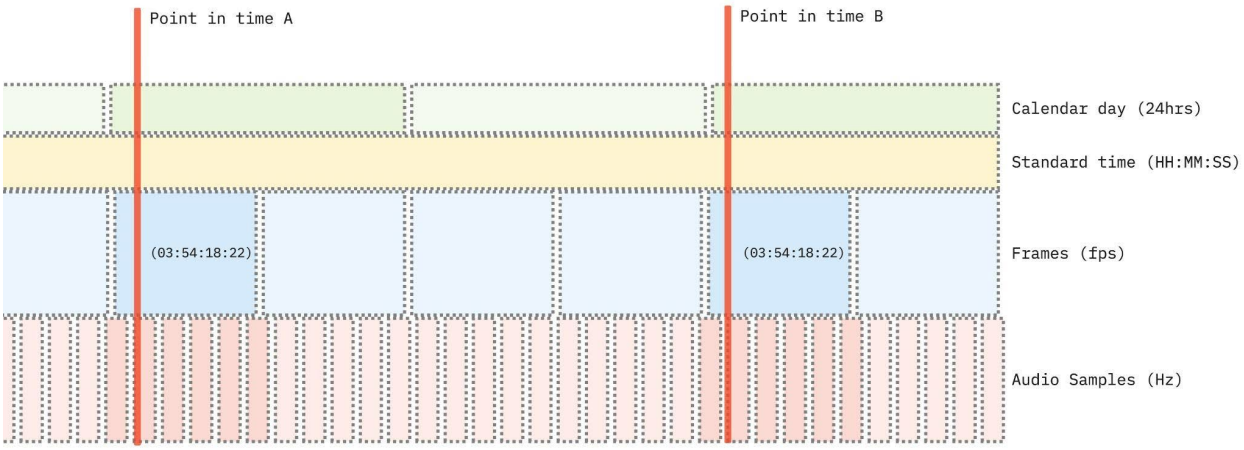

Figure 2. Two discrete points of time across several calendar days may have the same timecode value

In a strange way, timecode is both variable in and of itself but also does not support variable rates. While a particular timecode value has a fixed rate, it has no way to indicate what that rate

is. A frame — the smallest unit of timecode — can be 1/24th of a second, 1/30th of a second, and so on. Since ST-12 does not provide a location to store this information, the rate needs to be stored elsewhere in a file. Additionally, the rate itself cannot be variable.

Variable frame rates allow the duration of each frame to change for either technical/encoding reasons or for creative effect. The development of variable frame rates and supersampling temporal resolution is blocked at the moment because of the limitations of timecode. Amazing workflows could be unlocked by allowing acquisition to happen at one frame rate (say 120fps), creating proxies for editing at a lesser rate (say 60fps), and ultimately finishing in 24fps. The benefits would be that every shot could be time stretched with no need for optical flow interpolation. Motion blur and shutter angle would be post production effects providing sharper frames for VFX tracking and masking. Display and projection frame rates could be unbound by a static frame rate, creating a new dimension of creativity with regards to frame rate. Just like supersampling video and audio resolution, we could do the same for time.

The dramatic difference of 60fps versus 24fps is shocking for many viewers, but what about 33fps? What happens when we can subtly shift between 21fps and 29fps, or when we can replicate two or three blade film projector judder looks? There is precedent for variable frame rates in production in the video game industry and it is worth exploring those techniques in video production, but we need a standard that can support it.

In a strange way, timecode is both variable in and of itself but also does not support variable rates. While a particular timecode value has a fixed rate, it has no way to indicate what that rate is. A frame — the smallest unit of timecode — can be 1/24th of a second, 1/30th of a second, and so on. Since ST-12 does not provide a location to store this information, the rate needs to be stored elsewhere in a file. Additionally, the rate itself cannot be variable.

Variable frame rates allow the duration of each frame to change for either technical/encoding reasons or for creative effect. The development of variable frame rates and supersampling temporal resolution is blocked at the moment because of the limitations of timecode. Amazing workflows could be unlocked by allowing acquisition to happen at one frame rate (say 120fps), creating proxies for editing at a lesser rate (say 60fps), and ultimately finishing in 24fps. The benefits would be that every shot could be time stretched with no need for optical flow interpolation. Motion blur and shutter angle would be post production effects providing sharper frames for VFX tracking and masking. Display and projection frame rates could be unbound by a static frame rate, creating a new dimension of creativity with regards to frame rate. Just like supersampling video and audio resolution, we could do the same for time.

The dramatic difference of 60fps versus 24fps is shocking for many viewers, but what about 33fps? What happens when we can subtly shift between 21fps and 29fps, or when we can replicate two or three blade film projector judder looks? There is precedent for variable frame rates in production in the video game industry and it is worth exploring those techniques in video production, but we need a standard that can support it.

The problem of location

Location adds an interesting challenge when dealing with time, which most of us frequently experience as differences in time zones. With the proliferation of cloud technologies and

distributed teams, it is more and more important to localize timestamps to local time zones so that creation dates make sense to users in different locations.

In the context of productions using the cloud to instantly share dailies, as in a Frame.io C2C workflow, the uploaded media file doesn't exist in a practical way in any particular time zone, since the media can be accessed and interacted with from anywhere on the globe with very little latency.

It's possible (and common, in fact) for a comment to be placed on a clip from a different time zone than the one where the media originated — or even where it lives currently. When a reviewer leaves a comment on a piece of media that was recorded with time-of-day timecode, but from a different time zone, we now have a time mismatch. So which time stamp is correct?

This fluidity in time zones is also indirectly used when changing the clock on a set to **00:00:00:00** at the start of the shoot, which might be 7:00 am, in order to give the production 24 hours of unique timecode and to mitigate the problem of the midnight rollover. If you're concerned about midnight rollover, just move midnight to another timezone. This is a clever way to avoid a tricky workflow issue, but it underscores both how arbitrary timecode is (again, timecode is not time) and how fragile it can be.

The problem of perspective

Next, the same moment in time can be perceived very differently from different points of view. Capture technologies are limited to a very narrow point of view, which is why there are often multiple devices used simultaneously to capture as much relevant data as possible about a particular moment in time.

In a typical acquisition, we have video, and we have audio. These are two unique perspectives. They describe the same moment, but with wholly different data types. In this instance, timecode works well to bring them together. However, it's common to have several different cameras shooting at the same time. We can synchronize all of these cameras and the audio recorder to the same timecode value, but we've also added another layer of complexity: We now have multiple perspectives using the same data type (i.e., video).

During the media management and editing processes, this means that we will have yet another source of duplicate timecodes. The timecode itself doesn't have enough information to let us know if we are talking about a frame from A-Camera or B-Camera. There is no inherent connection here, and so we end up using another layer of technology to create a unique identity of a frame, which is generally a Roll, Tape, or Clip ID. An example of this is the EDL[4], which requires not just source timecode but also a reel in order to know what frame to choose.

While multiple perspectives can most obviously mean different camera angles and audio, motion, telemetry, lighting, script notes, and more all also constitute distinct perspectives. The number of data types being recorded continues to grow every year. As more and more perspectives require time labels, we begin to see how timecode is unsuited for this level of complexity. We now have much more data associated with a particular moment in time (or timecode value) than we did before. This intersection of creation and time label illustrates a need for a time label that is also media identity. Knowing when something is created is not enough to identify what it is.

Today, timecode is most readily associated with synchronization, and we can use timecode to put our different perspectives together in time. However, timecode really only supports video frames. Timecode for audio is derived from sample rate and a capture of the timecode value at the beginning of the file. Audio itself does not inherently have timecode. Beyond that, many other data types we are beginning to record (motion, telemetry, etc) are likewise unsuited for timecode. Again, these are file-first media, and therefore receive a timestamp from the device that created them. But since timecode is arbitrary, and may not use time of day, there isn't an effective way to bring these into synchronization.

The problem of editing

Of course, the lifespan of a piece of acquired media goes beyond the capture phase. The whole point of identifying media in time is so that we can assemble it with media from other perspectives to create our program. Each timeline or sequence that's used to create a program has its own timecode as well — this is so that each frame of the edit also has a time label. Once a piece of media is used in an edit, an individual frame is now associated with two different timecodes simultaneously:

- Source Timecode
- Record Timecode

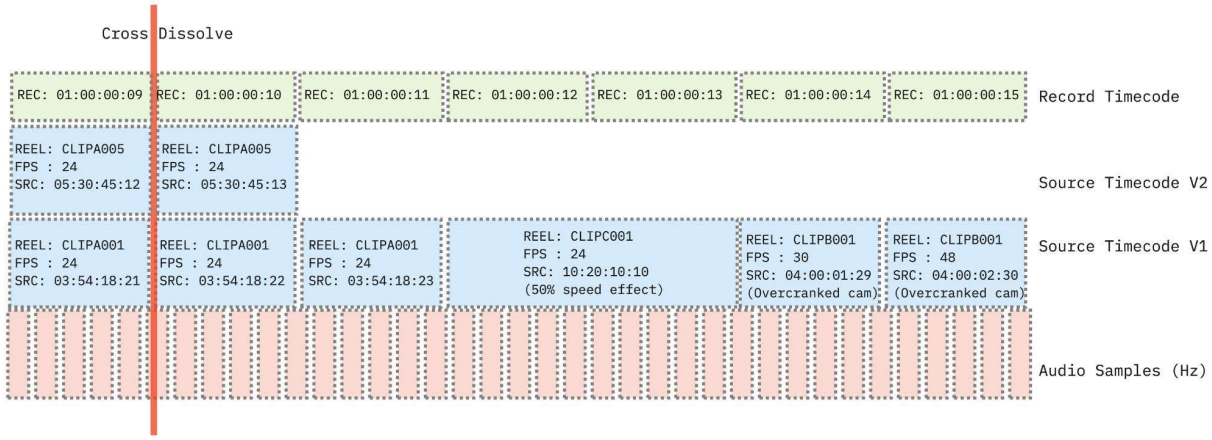

Figure 3. Once used in a timeline, a frame, sample, moment, or asset gains an additional time label

This problem compounds as you add layers of concurrent media, such as synced audio, stereoscopic video, multi-cam, composite video elements, and so on. This gets especially messy when an editor manipulates time with speed effects. And once speed effects are used, the original time label (source timecode) becomes irrelevant as the link between the frame and the timecode value is broken.

The timecode of the timeline (usually called the "record" timecode, again referring back to its origin in tape workflows) can start at any value, but often starts with **00:00:00:00** or **01:00:00:00**

by default in most tools. So now we need to track both where an asset is in a specific timeline (or several timelines) as well as which part (or parts) of the asset is being used. Again, the EDL is designed to track this — but the EDL exists outside both the timeline and the asset.

Source timecode and record timecode together can give us context about where an asset is being used and what part of it is being used. Separately, however, neither can give us any identifying information. Source timecode by itself tells us what part of an asset is being used, but not which asset. Record timecode by itself tells us where in the sequence the asset is being used, but not which asset nor which part of it. But ST-12 does not support recording these two values together. We need to store this context elsewhere, in a file like an EDL, with other information like reel, tape, or clip name.

The problem of context

In all of these problems, the common thread is that there is necessary context to carry along with the timecode to truly be able to identify the individual frame. A simple timecode value can tell us little about the frame it labels. We don't know what created it, what it is, or how it is being used. User bits are a part of the timecode specification for exactly this reason, however there is little to no standardization in the industry for how to use these bits.

This brings us into one of the problems we talked about initially: Timecode is not time. But, again, because it looks like a clock, we tend to think of it that way. As such, we try to use it to not just label a frame in time, but also identify that frame as unique. This is why we take care to avoid things like midnight rollovers. There are, however, only so many frames inside a 24-hour cycle and repeating timecode values is unavoidable.

Therefore, there may be additional relevant data that is not strictly time-based that we may need to associate with time-based media (slate, color correction, script notes, LIDAR, and so on). Contextual data to help us identify *what* the media is, which timecode by itself cannot do. We try to use timecode to tell us both *what* and *when* a piece of media (or unit of media) is, but there's not enough information in ST-12 to effectively achieve that.

And so, even if much of the data we may want to include is not time information, it is necessary for using a time label as a way to clearly and uniquely identify the smallest unit of a piece of media in the context of its original asset as well as among other assets. This is not a radical idea, and there are a number of methods for organizing production media. Most professional grade cameras embed their own sets of metadata into their camera files, for example.

There are new possibilities that open up when we create room for contextual data in the time label specification itself. For one, it can help standardize a way to both locate a frame or sample in time and identify what it is across camera manufacturers, video hardware manufacturers, and software developers. What else can we carry at the frame level?

Timecode as Identification

Returning to the initial purpose of timecode, the core functionality is not as a clock, but as a label. Timecode first and foremost is identification. The brilliance of timecode is that it actually can serve two purposes of identification at the same time: It can create a unique media identity

on a piece of tape, or inside of a digital file, and it can also identify a specific moment in time. Timecode creates a bridge between real time and media.

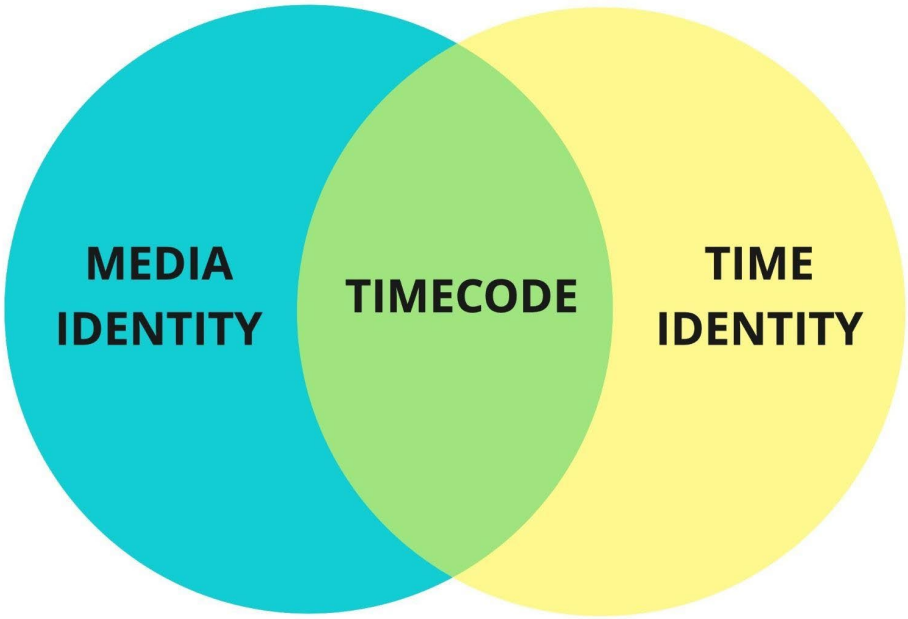

Figure 4. Timecode is used as both media identity and time identity

Making a distinction between these two different identities is important because they have different attributes and different needs to enhance their functionality for modern workflows.

Media Identity Requirements

- Can locate a single unit of time-based media, e.g. video frame, audio sample
- Can carry a history of identity when media is used to create new media, e.g. in an edit.
- Can carry contextual data, allowing media to be related to other media, i.e. filename, production information, camera position, scene, take etc.
- Can provide ability to distinguish generations/proxies of media from their parents.

Time Identity Requirements

- Can give precision to at least a microsecond level
- Can sync to a world clock
- Contains date and time zone information
- Contains frame rate and frame duration information
- Can be human readable

Prior Solutions to Timecode Limitations

The revelation that the limitations of timecode need to be addressed is not a new one. There have been several proposals to expand timecode (like ST-309 or RDD 46:2019, for example) or even replace it (like the TLX Project). Solving the limitations of timecode is a complex problem and requires a large amount of forward thinking. A solution to these limitations will need to not only enable the workflows we are trying to execute today, but also create a foundation to support workflows well into the future. Let's walk through a few solutions that have been proposed in the past.

Genlock + Timecode

As we illustrated earlier, timecode has a "too broad" problem. The smallest unit of timecode, the frame, constitutes a large amount of time. As such, the process of synchronizing multiple devices together leaves much room for error. This problem becomes worse when the sample rates of the devices are vastly different (like audio and video).

This results in offsets between media and devices that are at the "sub-frame" level. In these instances, the frame or sample boundaries of two different pieces of media may not be in exactly the same place. To get around this, video cameras can use Genlock to more precisely synchronize themselves to each other. This is a requirement for stereoscopic photography, for example, because sub-frame synchronization offsets cause the stereo effect to fall apart.

Genlock is also used to synchronize hardware devices receiving input from multiple different devices. Video switchers are a common use case for this (although with frame-sync tech this is less essential). Another example is when trying to film a monitor which has its own specific frequency. Genlock allows the camera to synchronize its individual exposure time to the cycles of the monitor. This is a sub-frame synchronization.

While this solves for the "too broad" problem, and gives us more precisely synchronized media, it neither solves for the "too narrow" problem nor the identification problem. While Genlock solves many other issues, it only addresses one small part of ST-12's limitations.

Timecode + UTC

Given that we understand we may need different levels of precision at different phases of the media creation process, it's reasonable to take this one step in the other direction: Where timecode can be enhanced with a wider standard of time, such as UTC or Coordinated Universal Time.

UTC is designed to offer a standard clock to synchronize global efforts. It carries information about the date and also considers the variable of time zones. It is broader than timecode in that it treats seconds as a constant and doesn't consider intervals smaller than that. Typically, devices that have time events faster than one second (very common) will calculate the smaller intervals themselves but adhere to the global UTC timebase for the larger intervals (seconds and above).

Like most technology, there are nuances and problems to solve with UTC, but issues of global time related to Earth's rotation (leap seconds, etc.) are outside the scope of this document. The

takeaway point should be that a global time base can help to solve the “too narrow” issue with timecode.

Having date information in the time string (for example using ISO 8601) would eliminate the problem of duplicated timecodes each calendar day and would help to solve the midnight rollover issue. It also helps with the problem of identification, as it does actually represent real time as well as real place (through time zones). But timecode and UTC by themselves still do not address all the limitations of timecode.

It’s also worth noting that there are alternatives to UTC including **GPS (**Global Positioning System time), **Loran-C (**Long Range Navigation time), and **TAI (**Temps Atomique International).

TLX Project

Over the last few years, SMPTE members have been working on a new standard to update ST-12, which they have dubbed “TLX.” Derived from “Extensible Time Label” the purpose of TLX is to create a time labeling model which has high precision, persistent media identifier and is “extensible” allowing for the addition of arbitrary metadata information. Without explicitly making the distinction, TLX succeeds at separating out the concept of time identity and media identity by leaning into two key notions:

- Digital Birth Certificate
- Precision Time Stamp (IEEE 1588 Precision Time Protocol)

The desire to add more extensible information enhances the media identity piece of the puzzle. The complexity of what this could mean is evidenced immediately by the need to define TLX Profiles which could define a certain set of required key-value pairs depending on the application. This is a wonderful approach to the problem of context.

The TLX committee has also acknowledged the need for multiple time labels on a single instance, which addresses the problem of editing, although the true complexity of this is worthy of much investigation.

One facet of the limitations of timecode that we were unable to identify as addressed by TLX was the problem of size and the subsequent problem of variable frame rates. Knowing the duration of a single instance (frame or sample) is vital for being able to open up the door to a new generation of time manipulation and handling of multiple frame rates — whether linear in a single timeline, or in parallel by supersampling time.

Finally, it’s not clear how TLX would be backwards compatible with ST-12.

Finding the Right Successor to Timecode

Before a new standard can be developed and accepted, we need to think about what it needs to do and what it needs to provide. Taking all of the outlined limitations of timecode and prior proposed solutions, we determined that a new time standard should:

1. be able to identify the smallest unit of an asset (i.e., frame or sample) as unique from other units and other assets;

2. be able to identify how assets relate to each other in time;
3. be data rich;
4. be able to be easily implemented into existing and future file containers;
5. and be able to have provisions for backwards compatibility with ST 12

While thinking through this, we also decided that we need to set up guardrails to help guide us on a trajectory into the future. And so, we started with a few assumptions. In the future, we assume that:

1. video transmission will disappear, to be replaced by file transmission;
2. all hardware will be connected to the internet;
3. all original assets will transmit to the cloud on creation automatically;
4. and video, audio, and time resolutions will increase for the purpose of super-sampling.

Working with these assumptions and future automated workflows in mind, we determined that the solution should be network based, metadata driven, and extensible.

Timecode + Frame Rate

For the sake of illustration, we're going to visualize a model using UTC as a world clock. However, while UTC may offer a way to synchronize at the level of the "second," and help to address the "too narrow" issue, we need to also look at the "too broad" issue of timecode.

We can consider that timecode in the context of media creation has one unique constant, which is that media creation involves a "range of ranges of time." We're not so interested in the single theoretical point in time, but rather a certain amount of time that allows us to slice up observable reality in a meaningful and often artistic manner.

This means that the rate of time (for example, the 1/24th of a second value at 24fps) is of paramount importance to media creation. Carrying rate with time also means that certain calculations can be performed to arrive back at what we expect from timecode. Loosely:

$$UTC\ second\ +\ framerate\ =\ timecode$$

If a camera and sound recorder were synced to the second using UTC, and then each ran at their own sub-frame rate, they could be synchronized using the combination of UTC time and framerate.

Carrying frame rate with each frame's timecode would at least solve the "too broad" issue because we would know how long each frame is meant to be.

It's also worth pointing out that framerate alone is slightly limited in that the framerate is usually a display rate linked to the refresh rate of the display device. The individual frame is affected by how much data is sampled during that time. In cameras, we talk about the shutter angle or exposure time. Including these values with the identifying timecode will give more meaning to the individual frame value.

Carrying frame duration could be a very valuable piece of data for cinematographers to always be able to determine the shutter angle of the picture, but it could also be used to vary the frame duration on the display device. A projector or monitor would be able to display a frame with the time it was actually captured. Including this information addresses multiple variable frame rate scenarios as mentioned here and earlier in this document.

Here is an example of data to carry in a single frame. *To be clear, this is not a proposition of a standard, but an exploration of requirements.*

```
timecode = {
"utctime" : "2021-09-16T16:48:02Z",
"rate" : 24000/1001,
"frame_duration" : 1000/48048, #180 deg shutter at 23.98
"utc_frame" : 3, #how many frames after the second boundary
}
```

Note: it will be possible to calculate shutter angle by separating out frame duration from frame rate. It will then become a choice by the hardware to hold the frames in a display device for the duration of the frame, or the frequency of the rate. Essentially, all the time data could be provided to either recreate the exact acquisition rate or interpret it differently.

Backwards compatibility

It will be essential that the next generation of timecode be backwards compatible with the current generation. This means that if the new standard were to adopt a world clock, then in order to convert back to standard timecode, or to localize to local time of day, we would need offsets.

```
timecode = {
"utc_time" : "2021-09-16T16:48:02Z",
"rate" : 24000/1001,
"frame_duration" : 1000/48048,
"utc_frame" : 3,
"local_zone" : -5, #how to normalize the time to a local time
"utc_offset" : "2021-09-16T00:00:00Z", #offset back to 24hr clock
"tc_string" : "11:48:02:03" #time of day TC clock localized
}
```

In the case of editorial files, where UTC would not make as much sense, it would make sense that the utc_time would be the creation time of the file or frame, and then in order to get back to a desired standard timecode, an offset could be used to bridge creation time and timecode.

```
timecode = {
"utc_time" : "2021-09-16T20:48:02Z",
"rate" : 24000/1001,
"frame_duration" : 1000/24024, #This can be different from acquisition duration
"utc_frame" : 0,
"local_zone" : 0,
"utc_offset" : "2021-09-16T19:48:02Z",#note one hour offset
"tc_string" : "01:00:00:00"
}
```

The benefit of this would be a standard that can connect creation time with arbitrary timecode. It would also allow for a method to jam device clocks using existing technology, perhaps using user bits to carry the date from which to create a valid **utc_time**.

Timecode itself does not provide or require accuracy; the actual accuracy of the timecode is only as good as the practices and tools that generate it. In other words, the UTC time does not have to be accurate, but if the tool can provide accurate UTC time, then the standard should have a place to store it.

Additionally, since a new standard would be designed to be file first, it can be added to a file container in a different location than where containers are already storing ST-12 data. In this way, a new standard and ST-12 can exist simultaneously in the same file.

Timecode layers

It's common for media to carry multiple timecode tracks, but there is not much standardization here. Being able to carry multiple timecode tracks with additional data would eliminate the problem of losing reference to the source timecode when the media is rendered into an edit. In a digital asset world, there are not significant barriers to carrying an arbitrary number of timecode layers, including varying layers per frame.

Imagine that the source timecode for a given frame was able to be appended, not overwritten, when being rendered into a transcode or an edit. This would then create another place to track the generations of media, empowering users to be able to trace their media back to its source.

In some ways, this could echo the way ACES defines a color pipeline—we could define a time-event pipeline. By having layers and framerate, it's conceivable that the timecode stream could carry multiple frame rates. The source timecode value might get duplicated or dropped in the case of a speed effect, but the output timecode would be consistent. This would allow for a way to track speed changes only using the timecode track.

```
timecode = {
  [
    {
      "layer" : 0,
      "layer_name" : "source",
      "utc_time" : "2021-09-16T16:48:02Z",
      "rate" : 24000/1001,
      "frame_duration" : 1000/48048,
      "utc_frame" : 3, #how many frames after the second boundary
      "local_zone" : -5, #how to normalize the time to a local time
      "utc_offset" : "2021-09-16T00:00:00Z", #offset back to 24hr clock
      "tc_string" : "11:48:02:03" #time of day TC clock localized
     },
     {
      "layer" : 1,
      "layer_name" : "edit",
      "utc_time" : "2021-09-16T20:48:02Z",
      "rate" : 24000/1001,
      "frame_duration" : 1000/24024,
      "utc_frame" : 0,
      "local_zone" : 0,
      "utc_offset" : "2021-09-16T19:48:02Z",
      "tc_string" : "01:00:00:00"
     }
  ]
}
```

Timecode context

The final point to look at here is the problem of perspective. While we may be able to avoid duplicate values by incorporating UTC time, we would need additional data to recognize frames that are recorded synchronously as unique points of view.

Because the media being created could be video or audio (or other), the main context key to include would be the concept of “creator.” Creator is a complicated concept with media, so this field should exist, but the contents should be flexible. Something as simple as a serial number could be added to create fields that would result in a unique hash.

While creator answers “who,” we also want to answer “what.” This is not really possible to standardize given the amazing variety of what could be captured, so it may be as simple as updating the concept of “user bits” to user_data allowing this to also be a variable field. This would be a place to store whatever else might be helpful to create context, but it can’t be a requirement.

This data is not time data, but it enables one of the primary uses of timecode in practice, which is to serve as a sync point between various pieces of media.

```
timecode = {
"time_identity" : {
    "utc_time" : "2021-09-16T16:48:02Z",
    "rate" : 24000/1001,
    "frame_duration" : 1000/48048,
    "utc_frame" : 3, #how many frames after the second boundary
    "local_zone" : -5, #how to normalize the time to a local time
    "utc_offset" : "2021-09-16T00:00:00Z", #offset back to 24hr clock
    "tc_string" : "11:48:02:03" #time of day TC clock localized
                    },
"media_identity" : {
    "uuid" : "4ee3e548-3374-11ec-9f4f-acde48001122",
    "creator" : {
        "application" : "ARRI_ALEXA",
        "serial_number" : "818599130"
        },
    "user_data" : {
        "scene" : "101A",
        "take" : "01",
        "clipname" : "A001C001"
      }
}
```

Implementation

Once a standard is reached, it will need to be implemented to the video and audio file containers used by the industry. The thrust of the specification would likely be around the data structure defining certain required fields and how to handle concepts of layers, nested dictionaries, and unique instance identifiers.

We think the majority of fields in the above examples should be requirements, with the exception of **user_data** examples, and, of course, the nuances of each field needs further investigation before a standard can be reached. Again, this is not a proposition of a standard, just suggestions of requirements.

Beyond this, it would make sense to require that the data model can be easily converted into XML and JSON (as proposed by TLX).

Based on our understanding of the trajectory of video technology, the requirements for carrying this data in a file format may be very different than what is required for a digital video signal, and we believe that carrying this data *inside* of existing file formats is an undeniable entry point.

There are many points of cross over between file-level metadata and per-frame metadata. It's not clear where to draw the lines between the two, or what sort of metadata should be excluded from an extensible timecode format. Should color data not be included? What if this is useful for understanding the color pipeline of rendered media, on a per clip or per frame basis? Should it include time information about layers which are not visible?

Where do we draw the line between timeline formats like AAF and FCPXML which give much more media information that is usually available in a single flat file?

Isn't it highly inefficient to carry deep and redundant metadata for each frame?

Time Codec

Given the profound advancements in video codecs to compress difficult-to-compress data such as video color information, it doesn't seem like a stretch to think that an efficient codec for per-frame time data would be out of reach. Even if we consider the amount of data that would be generated by stamping each individual sample in an audio file, and how this would be carried in a file that has both audio and video, the reality is that the amount of duplicate data that would be present in this stream makes it a highly compressible data set.

Ultimately, if we look at the fact that digital cameras are capable of creating the exact same file formats as the non-linear editing tools, there's no reason that a codec for extensible time information could not be implemented as a supported codec all the way through the media production life cycle.

Generally speaking, NLE render options include a choice for video codec and audio codec, and the implementation of a time codec would fit well into that model. There would be options to select a specific profile, for example (much in the design of TLX), which would define what bits of metadata get added into the file on a per-frame basis. Things like timecode history could be included or discarded. There could be options for encryption, if this media identification information were sensitive and needed an additional layer of security.

To the question above about AAF, we could have options to include specific NLE effect data or not. In truth, this timecode codec could be extended in such a way to be a viable alternative timeline representation. Could such an approach help with interoperability between NLE systems? The standard itself would be agnostic, but through partner developments, we could establish specific profiles which relate to required fields for Avid, Premiere, Resolve, etc.

A successful implementation of this codec would be open-source, and handlers could be built to extract timecode codec data into human readable forms like XML and JSON, as well as directly into existing interchange formats like EDL, AAF, FCPXML, OTIO and more. The beauty of carrying this data inside of files is that there would be an entirely new option for interchanging this data without the need of additional sidecar files.

When combining dailies footage into an edit, we can imagine that an NLE that supports this timecode codec would need to have the ability to render layers of timecode information into a video output — say H264 wrapped as an MP4, with a Video, Audio and Timecode stream, not all that different than how it works today. When this MP4 file is loaded into a new NLE or DAW system, if this system supports the codec, then all the information about the source media used to create this H264 render would be available to be recreated as individual frames and clips for conform or mix.

Now the intention here is not to create an alternative to timeline interchange formats, but if you start with the purpose of identifying individual time instances in an extensible manner, then it's a logical conclusion to be able to programmatically identify every single frame and layer of a rendered file, which by default becomes a highly effective interchange format.

In other words, if a new standard of timecode can achieve the level of identification that its *own purpose* demands, then it will empower the next generations of highly effective digital workflows to the same degree that ST-12 did for linear and non-linear editing starting back in 1975.

Time Pipeline

Finally, over the last few decades there have been incredible advances in camera technologies creating digital images that have incredibly sharp resolution, wide gamut color profiles, and efficient encoding profiles. Similarly, display technology has also had major advancements in spatial resolution, contrast ratios, screen thickness, and energy efficiency.

To support the advances in camera and display technologies, post-production pipelines and standards have evolved to handle more data. NLEs now support multiple resolutions on a single timeline. Sophisticated color workflows, such as ACES color from the Academy of Motion Picture Arts and Sciences, create pipelines for managing the entire lifecycle of the manipulation of color in media creation. Dolby Labs invented a dynamic metadata-driven color encoding process with Dolby Vision, which encodes various levels of color correction in a single deliverable to adjust to the variety of possible display devices. Object-based audio, in technologies like Dolby Atmos and DTS-X, aims to create truly immersive audio experiences, surpassing the limitations of surround sound.

The idea of a pipeline that we can use to track the journey of a single sample from delivery to acquisition, including original time and location data, raises two very big questions.

One, can this be used to ensure the veracity and authenticity of a frame? Can we use this to track any possible manipulation applied to that frame? In this context, with the rise of AI driven manipulation (like with deep fakes), a standard like the one suggested here could also be a fingerprint.

But the first question raises the second question: If we want to use this to help verify the authenticity of what we're seeing, how do we protect this data from being manipulated or spoofed itself? Similarly, if we can use this to identify exactly when and, importantly, where a frame was created, does this pose a threat to privacy? This ultimately leads us to providing a provision for encrypting this data. This is complex, but necessary to explore in a final standard.

Conclusion

Despite all the advancements in capture technology for image, sound, and data, there have been few significant advancements with regard to capturing time. This isn't to say that there have not been impressive improvements in cameras and encoding technologies in regard to how they operate over time. Most mainstream cameras can record higher frame rates in smaller camera bodies. There have also been quite a few advancements in variable framerate technologies for the purposes of compression and video streaming, as well as much discussion about VFR for display technology.

But how can we connect the dots on the handling of time all the way through the process, much in the way that the Academy tackled the multi-phase journey of color through a media creation pipeline?

Time serves not only as the backbone of our workflows, but also as a core definition of what we are creating:

$$media = (image + sound) * time$$

As such, the way we record, transmit, translate, and interpret time data needs to catch up to our modern workflows. Moreover, it needs to evolve to ultimately allow more powerful and efficient workflows while also unlocking creative potential that was previously impossible.

It's time to talk about time.

References

1. Hess, John, "The History and Science of Timecode", YouTube, uploaded by Filmmaker IQ, 3 Apr, 2019, https://www.youtube.com/watch?v=PgX_R-JgpJE
2. "ST 12-1:2014 - SMPTE Standard - Time and Control Code," in ST 12-1:2014 , vol., no., pp.1-41, 20 Feb. 2014, doi: 10.5594/SMPTE.ST12-1.2014.
3. "ST 309:2012 - SMPTE Standard - Transmission of Date and Time Zone Information in Binary Groups of Time and Control Code," in ST 309:2012 , vol., no., pp.1-11, 5 Oct. 2012, doi: 10.5594/SMPTE.ST309.2012.
4. CMX, CMX 3600, "Description of the CMX EDL Diskette," Available at: https://xmil.biz/EDL-X/CMX3600.pdf

Bibliography

Adee, Sally, "What are Deep Fakes and How Are They Created?", IEEE Spectrum, Apr 2020, https://spectrum.ieee.org/what-is-deepfake.

IEEE, IEEE 1588-2008, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems,", Available at: https://standards.ieee.org/standard/1588-2008.html

International Organization for Standardization (ISO), ISO 8601-1:2019, "Date and time — Representations for information interchange — Part 1: Basic rules," Available at: https://www.iso.org/standard/70907.html.

International Organization for Standardization (ISO), ISO 8601-2:2019, "Date and time — Representations for information interchange — Part 2: Extensions," Available at: https://www.iso.org/standard/70908.html.

Roe, John H, "The Genlock for Improved TV Programming", RCA Broadcast News, 58:10-11, 1950.

SMPTE, "RDD 46:2019 - SMPTE Registered Disclosure Doc - SMPTE Timecode Extensions — Relationships to Higher Rates and Date-Time," in RDD 46:2019 , vol., no., pp.1-78, 5 Sept. 2019, doi: 10.5594/SMPTE.RDD46.2019.

SMPTE, ST 2065-1:2012, "Academy Color Encoding System (ACES)," Available at: https://ieeexplore.ieee.org/document/7289895.

Symes, Peter, "Beyond SMPTE Timecode: The TLX Project," presented at the SMPTE Standards Series Webcast, White Plains, NY, Feb. 2019.

Anon, "The Evolution of Media Creation: A 10 Year Vision for the Future of Media Production, Post and Creative Technologies", MovieLabs, 2019.

