On Oct 27, 2014, at 4:17 PM, Walter Arrighetti wrote:

The attached CLUT file describes a "toy-model" *creative* LUT converting from **ARRI Log.C** look (probably generated into an ALEXA camera on-set), into **Rec.2020**., which is the result of several creative and technical steps -- each done with a particular color conversion (read below).
After some generic metadata (who made it and when, where's the Schema/DTD file this XML is based upon, etc...), the color encoding system for which it's intended (`<System>` node set to ACES) and the minimum version of it required to correctly process the LUT (`version` attribute set to 1.0) are described.
Color-Pedigree for both the Source and target color spaces is provided, including a reference documentation / schema (`xmlns` attribute in the `<ColorSpace>` nodes), the digital encoding (IEEE-754, 32-bit floating-point for Log.C, vs 10-bit unsigned integer packed for Rec.2020) as well as the colorimetric information where available (primaries, illuminant, etc.)
Information on the intended viewing environment is provided as well (within the `<CAM>`, or *Color Appearance Model*, node). Finally, each key nodes in the CLUT has its unique identifier (expressed as v4 UUIDs in the `id` attributes), to uniquely re-use them or cross-reference them by third-party items (e.g. *Clipster*, *NUKE* or OpenColorIO project files).
Finally, a **pedigree** of the whole process is included, where there are three purposes identified, but ultimately just one described completely in the file.
The three purposes are given three style names (much like CSS for the web works): "***viewing***" for viewing environment (a TV grading bay?), "***proxy***" for low-res screening (Retina iPad? sRGB monitors?), "***master***" for DCDM / IMF archival renderings. Each pipeline is associated with a purpose by simply flagging its `class` attribute. The order in the pipleine is specified by the `num` attribute.

- Map #0 (common to all three pipelines) is just a creative LUT generated on-set as a CDL (maybe directly into the ALEXA).
- Map #1 (common to all three pipelines) is the IDT (described in CTL) from Log.C @ 800ISO, as provided by ARRI (and referencing the color space specified in the `<Source>` node)
- Map #2 is just a neutral LMT, also desribed as a CDL, acting within ACES RGB itself (an LMT acts within the same color- space);
- Map #3 specific for DCDM/IMF ***master***ing is just an ODT (described in CTL), and going to DCI *X'Y'Z'* color space;
- Map #3 common to ***viewing*** and ***proxy*** pipelines is just the RRT;
- Map #4 specific to ***viewing*** is the ODT that goes to Rec.2020 color space.
- Map #5 specific to ***viewing*** is an LMT acting in the Rec.2020 color space and described as a DigitalVision *Nucoda* CMS file (which is a rather nice non-XML LUT format).
- Map #4 specific to ***proxy*** is just an ODT described as a Nucoda CMS file that converts to a Rec.709 space with gamma 2.4.
- Added *tolerances* (`delta` attribute in several nodes describing output-referred values as physically-measurable quantities, like `nits`).
- Moved all the gamut geometrical metadata as subnodes of unique `<Gamut>` node.
- Added Out-of-Gamut descriptors (`<boundary>` sub-node), describing surface mesh triples of the *gamut boundary*.
- Introduced the **aces** XML namespace (which all the relevant nodes might be ultimately made to belong to).
- Changed *root* node name from generic `<ColorLUT>` to namespace-dependent node `<aces:LookModificationTransform>`.
- Changed subnode name describing the *resulting* 3D LUT from `<LUT>` to `<ColorLUT>`.
-

Even if three pipelines are described in the pedigree, just one of them is actually realized by the LUT (and completely, numerically described, in the **<CLUT>** node): that because the **<CLUT>** node has just the ***viewing*** style embedded in its **class** attribute. Again: this is much inherited from W3C's CSS dialect.
The fact that the pedigree not only includes the "active" pipeline, but also two additonal scenarios (***master*** and ***proxy***) that have steps in common with it, further adds “trust” and expandability to the LUT.
Each node in the pipeline can be externally referenced by the UUIDs, so color pipelines that actually pick up pieces from different CLUTs or CTLs can be actually generated, then re-saved as a single CLUT that either re-includes all of them or references them-
This chain of dependencies can be

- **dangerous** on one side (as one might save a .clut believing it is self-consistent, only to discover that it references an external LUT or CTL),
- flexible on the other, as one might just change the CTL that sits in between (for example during an ACES minor upgrade) and automatically have updated LUTs for a whole set of pipelines
- very strictly controllable, likely by external Color Management Systems, that validate UUIDs and give consistency checks of the whole pipeline (for example warning that a DCI P3 output from one LUT is not fed as input of a ACES.Log during concatenation).

This is much like what happens for Digital Cinema validation tools.
Finally, as a last comment, let's take into account security. There are a couple of Python scripts embedded into the XML (e.g. for realizing the ARRI Log.C curve translation). This can be avoided by using references to Python scripts rather than embedding application/* MIME types into it.
No one, of course, wants ColorLUTs to be potential *vector for malicious code injection.*

6.7 KB

Peter Postma Wed, 29 Oct 2014 at 12:13am

Walter -
As Jim said, lots of ideas to consider, but ultimately I think this is way more than what's called for to get ACES 1.0 shipped so most of the discussion should be tabled until then. I think a comprehensive and portable color pipeline descriptor like this is something to strive for in the future, and something to keep in mind as we finish designing the minimum bits for ACES 1.0, but trying to do it all at once would hamper adoption.
The broad strokes of the XML- the use of UUIDs and Xpath compatibility are all good. Creating dependencies on Python or even on CTL should be avoided if possible. Not just because of security issues but because it makes getting broad support across applications more difficult. It would be better to refer to ACES standard transforms (the reference implementations all being in CTL of course) and other formulas can be specified with a small set of operators that could easily be translated to OpenCL, OpenGL, C, Python, or whatever by a host application.
We also have to be careful about mixing informative metadata with instructional metadata. Presumably the <Source> and <Target> blocks are all just informative, or would you expect an

application to act on the information in some way? I'd expect the <Pipeline> block to all be instructional, but if your example is suggesting the in-camera CDL should be applied before the IDT that would ruin everything.
Also, in the future it's worth running your example through an XML validator, just opening it in Chrome or Firefox will point out a bunch of errors.
Thanks,
Peter Postma

Walter Arrighetti Wed, 29 Oct 2014 at 12:28am

Thanks for the insights Peter.
As I told Jim yesterday, I wouldn't expect all of this be implemented for ACES 1.0.
In fact only a "minimal" set of tags/nodes should be mandated at the start. Those would work for the simplest possible LUT scenario and be interpreted by most parsers and vendors.
Additional levels of metadata can become optional (yet strongly suggested in more "professional" LUTs -- like those used in a DI, possibly multi-site, facility).
Pedigrees and UUID-references are an example of optional Metadata: they can be used in more complex color pipelines so that changing just one piece of the overall LMT (much like you do with *Nuke* or *Resolve* nodes) doesn't destroy the whole process and lends in for self-refresh. I think OCIO is designed with this flexibility as well -- at least as at a concept level.
Simpler workflows (where you really don't need anything more complicated than using one LUT per passage) can still use and consider the simplest metadata only: **`<ColorLUT>`**. This tag and a few others can be considered for ACES 0.1.
I agree with you that scriptable parts should be ultimately written in CTL !!I designed and have been using this LUT format with my own engineering scripts, which are mostly Python-based (that's where it's from), but an AcademyLUT should use more trivially use CTL (which is also more secure).
As for the **`<Source>`** and **`<Target>`** nodes instead, I don't agree with you. *Yes*: some info in them is purely decriptive and doesn't really play any role in CV computing.
But some other --like color-space names, legal range CV definitions or bit encodings (float vs int10 vs int16, ecc.)-- should be always used.
Any parser should be aware of them in order to "raise a warning" when things like the following happen:

- concatenate a LUT with legal-range output CVs after a LUT expecting full-range input CVs;
- A LUT generated with specific interpolators is used by a software that actually has a different interpolation algorithm;
- A LUT that outputs to full-range Rec.709 (gamma 2.4) CVs, on a viewing device set to Rec.2020 or to legal-range Rec.709 (gamma 2.2) output;
- Tonecurves, gamma corrections, shapers or simply bit-depths are incorrectly assumed by the user/operator or by the parser itself, whereas the LUT carries different interpretations of them burnt inside;
-

Peter Postma Wed, 29 Oct 2014 at 7:55pm

I think there are two separate problems we need to solve which are getting mixed together here. One is to document the transforms used to create a set of ACES files so that when they're handed from facility to facility or device to device you know the version of ACES used and which LMT, viewing and display transforms were used. The second is to provide a container for LMTs that can hopefully be adopted by a majority of software and hardware tools supporting ACES. There would be advantages to combining these things into one file that both describes the pipeline and provides the transforms to implement it, but at the huge cost of hampering wide adoption.
ACES does not require that transforms be implemented in any specific way, only that manufacturers match the reference CTL implementation. This allows vendors to optimize for processing speed and flexibility while remaining compatible with the larger ACES ecosystem and doesn't require them to be dependent on any particular code set. For a number of reasons, few applications or devices are implementing CTL directly. In Baselight for instance we use OpenGL for IDTs and ODTs and a combination of OpenGL and 1D and 3D LUTs for the RRT. Many implementations are just LUTs, which is fine if the LUTs are crafted with enough care to closely match the reference implementation. The specifics are left to vendors, Vendor A may use tetrahedral interpolation and a smaller LUTs than vendor B using trilinear interpolation due to speed or hardware limitations. They also both may have different approaches to handling ranges outside 0-1.
Therefore the color metadata document should reference the standard ACES transforms. This could be with reference to the specific CTL files with UUIDs which a vendor would use to know which of their matching internal transforms to use. When it comes to LMTs beyond the ones distributed with ACES there is a need to provide the actual transform. The recommendation for v1.0 would be to implement them as either CDL or in the Common LUT Format which hopefully many vendors will adopt. Again I think there is value in providing formula options in the future but consideration needs to be given to what is likely to be widely adopted, and neither CTL nor Python are good candidates in my opinion.
The color metadata for v1 should also be limited to the standard pipeline of LMT->RRT(->TCT)->ODT so there is no need to check for color space mismatches between transforms. An application could check that the final resulting color space does match the application's display setting to avoid problems with legal vs full and the like.

Walter Arrighetti Wed, 29 Oct 2014 at 10:59pm

They are all interesting points Peter.
As I said before, the LUT/LMT format can be brought down as easy as a wrapper for a 3D LUT codevalues' list, with some really basic metadata.
That way vendors just have to code a parser for the AcademyLUT around their existing CMS
Deeper metadata (including colorimetric infos for nerds) will be implemented later, as products will learn how to parse them later.
In the real world LUT interpolators are the "secret sauces" behind color-related products, so it's clear that **interpolators cannot be enforced** effectively.
Ideally instead, interpolation algorithms should not be left fully open to interpretations -- the same LUT would produce different effects if baked by different apps.
But, again, I don't see any "onboarding problems" as long as nodes like **`<interpolation>`** are *optional* (not mandated for LUT readers/writers).
The apps which can write/understand those tags, will honor their content and write it. Those that don't or can't, won't.
*Example (with **`<interpolation>`**, but holds for many others):*

If product *A* generates a LUT with interpolator #1 then the LUT carries product *A*'s name and "`1`" in the `<interpolation>` node because that is what *A* uses.
When product *B* reads that LUT, it parses the `<interpolation>` tag as well: if *B* is capable of using interpolator #1 with similar performance then it just uses it even if it's not its default method.
If *B* is not capable, it will simply use its own method.
It's advisable that, in this case, the professional product issues some kind of visual warning to the color manager itself (that's a matter for the UI WG).
On the "*ACES certification*" perspective instead (which is also another topic of discussion), what is the stand-point?
Does the product need to match `CTLrender`'s behaviour on every interpolated CV, or it's sufficient for it to match reference implementation on the LUT explicit CVs, in order to get "ACES certification"?
The same goes with tags that interpret things like legal/full ranges, int/float encodings, etc.
That's the main reason I'd suggest Academy LUT includes those metadata as well even if no product is expected to interpret it as of ACES 1.0.

Walter Arrighetti Tue, 4 Nov 2014 at 1:24am

I am reading the CommonLUT specifications which are also interesting for the Academy LUT format, although there are some radical design discrepancies, mainly due to the different broad scopes of the two containers. But I still believe they could be possibly merged to get an even better Academy LUT format.
Attached is a trimmed-down version of the proposed Academy LUT format, with a more minimal set of tags that describe both the LUT itself and the interpretation of the source and target codevalues the LUT is designed to map. Listening to Peter's suggestion, I removed the really non-necessary tags and revised the syntax, so that's easier to quickly code simple parsers that interpret this simple encoding. Later versions will include (and need interpretation capability) for more advanced features.
Outstanding modifications since v4 are:

- Change of root namespace from '**`aces`**' to '**`ampas`**' (in order to be immediately mergeable with other Academy namespaces (including Common LUT's).
- Change of the file extension from `.clf` to `.clf.xml`.
- Complete removal of the **`<pipeline>`** branch, describing the whole color pipeline (and, therefore, of the *pedigree*) that led to the LUT described overall by the **`<ColorLUT>`** node (this would be the hardest part of the LUT to be interpreted and consistently managed across vendors -- so the effective action is to "temporarily" remove it).
- Fixed syntax errors and typos (so it is now a compliant XML file apart from the unregistered root namespace)

I also like to point out that the whole LUT is uniquely identified by the UUID specified in the **`id`** attribute of the **`<ColorLUT>`** node.

And, by the way, I think the above LUT file-format should not be viewed as an *alternative* to the CommonLUT format, but rather a potential *extension* to it -- which is the real power of XML after all.