International Color Consortium®

White Paper #28

Level: Intermediate

Date: February 2012

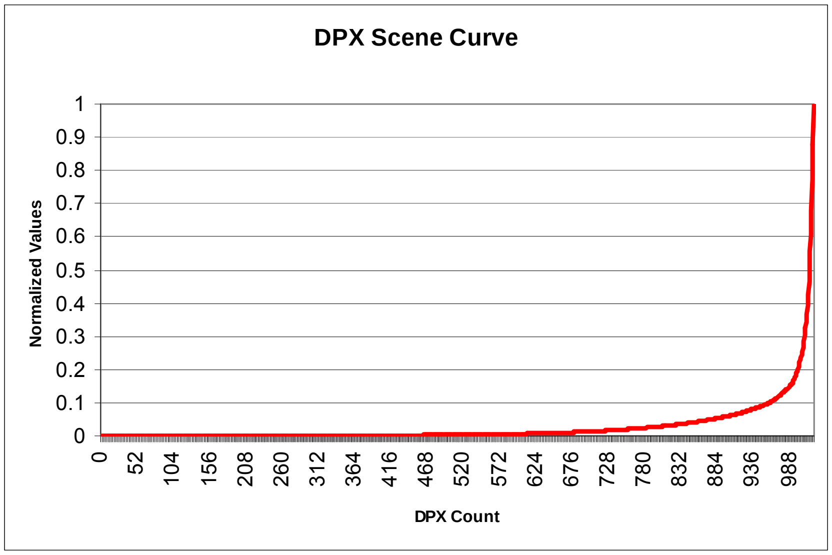
Introducing the New multiProcessingElements Tag Type

In November 2006 the ICC approved the multiProcessingElements Tag type as part of the Floating Point Encoding Range addendum to the ICC profile Specification. This addendum has since been included in the publication of ICC.1:2010. This new tag type's primary purposes are to overcome limited precision in ICC profiles by optionally allowing for the direct encoding of floating point data in an ICC profile, remove bounding restrictions for both device side and PCS encoding ranges, and provide for backwards compatibility to the existing ICC profile specification. A secondary benefit is that this new tag type provides for more flexibility in encoding transforms.

Note: the use of floating point in a CMM to apply profiles does not necessarily require the encoding of floating point in profiles.

The initial motivation for the Floating Point Device Encoding Range amendment stems from an attempt to use ICC color profiles for managing colors in Motion Picture and Digital Photography Workflows. The precision of profile elements such as LUTs, curves, and matrices is insufficient for the motion picture industry processing because:

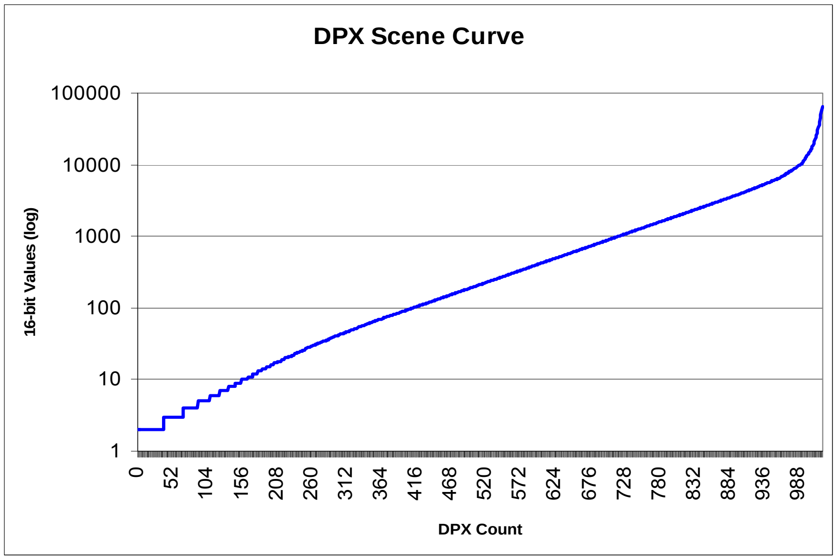
1. Encoding is limited to16 bits and therefore transform inversion cannot be performed precisely due to quantization errors. An example of such quantization errors can be found in the following example of trying to encode a DPX Scene curve in an ICC profile.



Some of the values of the curve are shown in the following table:

10-bit DPX Count	Curve Values	Normalized values	16-bit values
0	0.001855	2.44594E-05	2
1	0.001876	2.47420E-05	2
2	0.001898	2.50279E-05	2
…	…	…	…
37	0.002837	3.74154E-05	2
38	0.002870	3.78477E-05	2
39	0.002903	3.82851E-05	3
40	0.002937	3.87274E-05	3
…	…	…	…
684	0.895955	0.0118158	774
685	0.902699	0.0119048	780
686	0.909493	0.0119944	786
…	…	…	…
1021	58.629983	0.7732119	50672
1022	66.676150	0.8793247	57627
1023	75.826544	1.0000000	65535

The curve in this example has 1024 points, one point corresponding to each input 10-bit DPX count. What must be encoded in the ICC Profile are the values in the “Normalized values” column. Since ICC Profiles only support up to 16-bit precision, the values must be converted to 16-bit. It is obvious from the “16-bit values” column that doing so results in severe quantization. This quantization becomes obvious when the curve is graphed on a log scale, shown below:



2. Current profile transforms only support bounded device-side color encodings (IE [0, 1]), but unbounded (floating point) encodings are used in the motion picture industry.
3. PCS encoding is limited to encoding range of [0, 2) for XYZ values or [0, 100] for L* and [-128, 127] for a*, and b*

   NOTE: There has been some confusion regarding ICC support of encoding above-white values. Such values can be supported by setting the media white point appropriately and using the absolute rendering intent. Clarifying language is provided in the Colorimetric Intent Image State tag specification in ICC.1:2010.

Eight new tags that use this new Tag type are defined to allow for the optional substitution of the tags that define rendering intents in an ICC profile. D2Bx/B2Dx tags can now exist in a profile in addition to A2Bx/B2Ax tags. A CMM can now optionally first look for D2Bx/B2Dx tags and use them instead of the A2Bx/B2Ax tags or Matrix/TRC tags. Explicit definition of the absolute rendering intent is also now possible through the use of D2B3 and B2D3 tags.

The eight new tags are optional which means that existing Color Management systems can ignore them as private tags. This allows for profiles to be built and embedded in images, or otherwise used in workflows that do not support the use of these new tags without breaking those workflows. It is hoped that the use and adoption of these tags will be made easier because their presence shouldn't break existing workflows.

Note: CMM Support for Multi Processing Element Tag type is **optional**. This means that MPE based tag support is NOT guaranteed to be provided and implemented by CMMs in general! Additionally, all required tags **must** be present and valid.

The D2Bx/B2Dx tags all make use of the new multiProcessingElements tag type. Important aspects of this tag type include:

1. This tag type provides for an arbitrary sequence of processing elements to perform the device to PCS, PCS to device, or device to device conversion. Processing elements can be thought of as transformation steps that convert input channel data to output channel data.

   Note: The ability to have an arbitrary sequence is a significant difference from LutAtoB and LubBtoA tag types. Additionally, with an arbitrary sequence of elements a “limited” Static Programmable CMM is implied.
2. All the processing elements encode data using 32-bit IEEE 754 floating-point encoding.
3. The absolute rendering intent can be encoded with D2B3/B2D3 tags.
4. The initial repertoire of processing elements includes N-dimensional lookup tables, NxM matrices, sets of one dimensional segmented curves, and two future expansion elements that perform no operation.
5. The PCS for D2Bx/B2Dx tags is the floating point equivalent of the PCS in A2Bx/B2Ax tags. When D2Bx tags are connected to B2Dx tags no clipping is performed in the PCS.
6. D2Bx tags can be connected to B2Ax or Matrix/TRC based tags with appropriate clipping of the PCS values as needed, and A2Bx or Matrix/TRC based tags can be connected to B2Dx tags.
7. The CMM performs NO manipulation of data between processing elements. The CMM simply passes the results from one processing element to the next processing element.
8. Generally, up to 65535 channels of floating point data can be passed between processing elements.
   a. Processing element types are not required to support the upper limit of 65535 input and 65535 output channels.
   b. The channel usage of the first and last elements in a D2Bx/B2Dx tag must agree with the channel usage requirements of both the containing D2Bx/B2Dx tag and the profile header. (Note: Currently the color fields of the profile header limit the maximum number of channels to 15).
9. The fall back behavior of using A2Bx/B2Ax tags is prescribed if processing elements are encountered that are unknown to the CMM. This allows for a graceful handling for future expansion of the processing element repertoire.

10. The device encoding range for D2Bx and B2Dx tags is unbounded, but conversions and clipping may need to be made to be compatible with A2Bx and B2Ax tags. The equivalent device encoding range of A2Bx and B2Ax tags is converted to the range of 0.0 to 1.0 when applying D2Bx and B2Dx tags.

Figure 2 shows possible scenarios using D2Bx/B2Dx profiles.

Case 1 shows connection of a profile using a D2Bx tag to a profile containing a B2Ax tag or a Matrix/TRC based profile

Case 2 shows connection of a profile containing a A2Bx tag or a Matrix/TRC based profile to a profile using a B2Dx tag

Case 3 shows connection of a profile containing a D2Bx tag to a profile containing a B2Dx tag.

Case 4 shows the use of a D2Bx tag as a device link or an abstract profile.

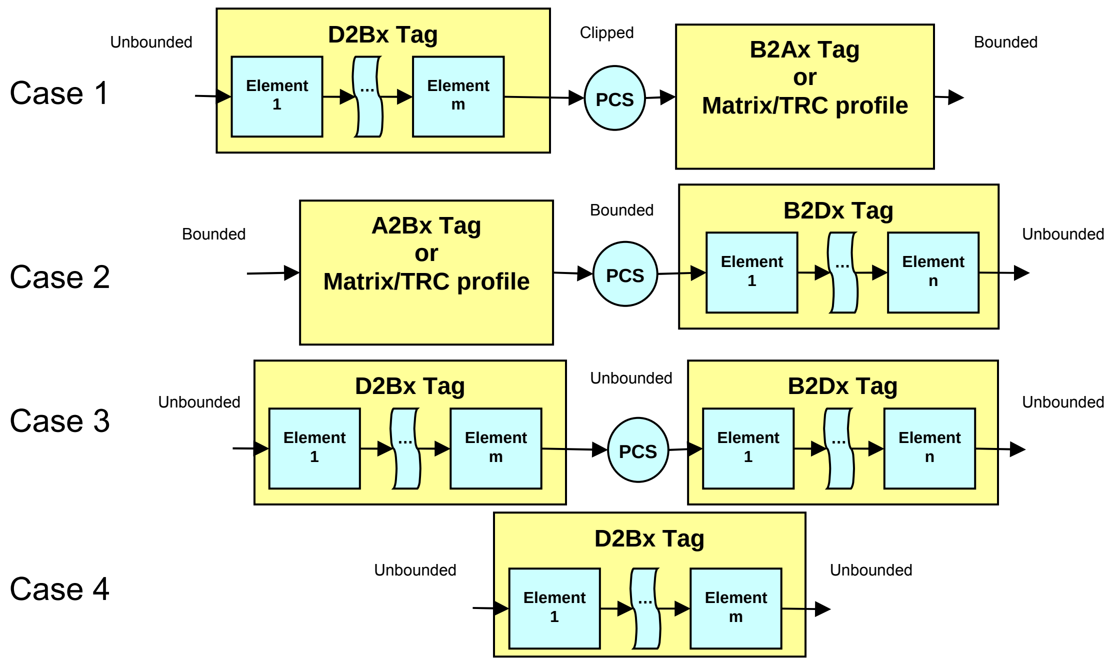


Figure 2 – D2Bx/B2Dx Scenarios

MultiProcessElementsType overview

The multiProcessingElementsType generally encodes the following:

1. A tag type signature (‘mpet’) followed by a reserved 32 bit integer set to zero.
2. The number of input and output channels associated with the tag. This should match the number of channels associated with color space fields in the profile header.

3. The number of elements in the transform along with an ordered array of processing element position entries. The Element position order defines the sequence of element processing. Each position entry contains an offset relative to the tag start as well as the size of the element data. Multiple position entries can refer to the same data
4. Data for all processing elements associated with the tag.

Processing Elements

The initial repertoire of processing element includes N-dimensional lookup tables, NxM matrices, sets of one dimensional segmented curves, and two future expansion elements that perform no operation. These all use 32-bit IEEE 754 floating-point encoding for processing and data storage purposes. An arbitrary number of these elements can be combined in any order to accomplish the purpose of defining a transform. Output channels from preceding elements are direct inputs to succeeding elements.

The Color Look Up Table or CLUT element (with signature ‘clut’) is used to store N-dimensional lookup tables. They can accept up to 16 input channels (constrained by the allowed ‘Number of grid points in each dimension’ in the CLUT element encoding) and output up to 65535 channels. The CLUT input range is from 0.0 to 1.0 since using grid points represents the sampled range, and clipping is prescribed for values outside this range. Scaling/conversion may need to be performed by a processing element before a CLUT element to get values into the range from 0.0 to 1.0. The output range of a CLUT element is the entire floating-point encoding range.

The matrix element (with signature ‘matf’) can be used to store an NxM matrix with a constant offset vector. The input and output dimensions need not be the same, and up to 65535 channels can be used for both input and output. The input and output range is the entire floating-point encoding range. (Note: This is different from other LUT tag types that can only store and use 3x3 matricies with offset).

The curve set element (with signature ‘cvst’) encodes multiple one dimensional curves. Up to 65535 separate curves can be defined. The curves are segmented to allow the entire floating-point encoding range to be used as both input and output. Up to 65535 segments are possible for each curve with positions and definitions of each segment definable. Each segment can be defined as a formula or sampled curve segment.

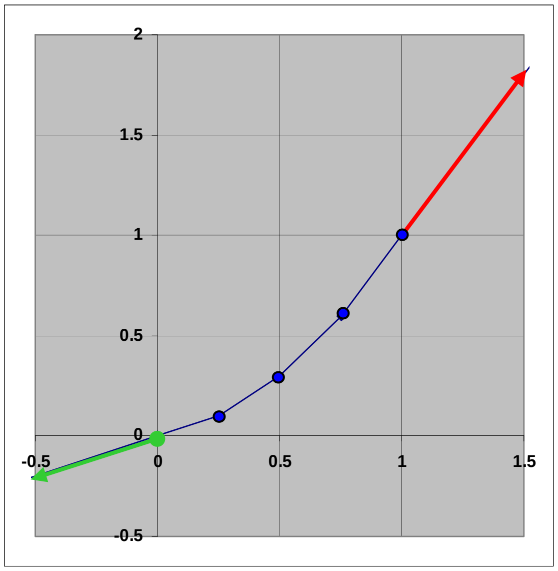
- Formula segments define a function type and provide parameters to the function.
- Sampled segments are equally spaced sample points defining a 1 dimensional look up table. An important aspect of sampled segments is that interpolation is performed from the last point of the previous curve segment. This means that the first interpolation point is **not** stored in the sampled segment.

To help exemplify this the following figure shows an example curve set element curve.

- The segmented curve contains two break points that defines three segments with break points at x=0.0 and x=1.0.
- The first segment (in green) is a formula segment that has a domain of [-∞, 0.0], and is defined by the formula y = 0.1 * x.
- The second segment (in blue) is a sampled segment that has the domain (0.0, 1.0] with four four sampled values: 0.1, 0.3, 0.6 and 1.0.

  Note: notice that the sampled segment does not store a value for the (0, 0) position as it gets its interpolation point from the first segment.

- The third segment (in red) is a formula segment that has the domain (1.0, +∞], and is defined by the formula y = 1.6 * x – 0.6.

- 

Possible additions to the processing element repertoire are also under consideration within the ICC. Two future expansion element types were included in the specification (with signatures ‘bACS’ and ‘eACS’) for expansion purposes. These elements encode a single signature value (effectively placing identification marks in the transformation chain) and have no prescribed operations – thus they define no operation on channel data. They simply pass the channel data to the next processing element.

Example

A D2Bx tag could possibly be used to encode a device model that goes from device values to spectral information and then to PCS. A sequence of processing elements can be used to model each step of the transformation.

The following example sequence of processing elements in a hypothetical D2Bx tag (i.e. the actual element boxes that would go into the D2Bx tag element boxes of figure 2) helps to show the programmable nature of this new tag type:

1. The first processing element is either an N dimensional lookup table element or a matrix element that converts the device channel data to 31 channel spectral information.
2. The second processing element uses a 3x31 matrix to convert spectral information to XYZ colorimetric information.

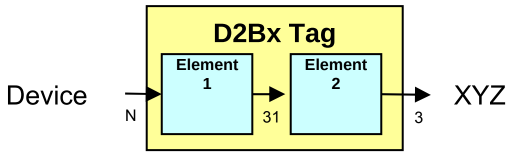


Figure 3 – Device to XYZ example

At this point, if the PCS signature in the profile header is ‘XYZ ‘ then the transformation is complete (In other words only two elements are needed to go from device to PCS). However, if the PCS signature in the profile header is ‘Lab ‘ then the following two processing elements could be included:

3. The third element uses a curve set to perform the cubic root portion of an XYZ to Lab conversion
4. The forth element uses a 3x3 matrix to complete the XYZ to Lab conversion.

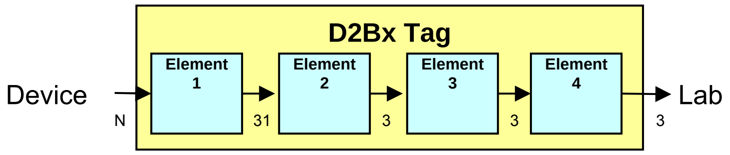


Figure 4 – Device to Lab example

It is important to note that the CMM has no understanding of what each of these processing elements is actually accomplishing. To the CMM this is interpreted as follows:

1. The device values are passed through either a matrix or N-dimensional lookup table to get 31 values.
2. A 3x31 matrix is applied to these 31 values to get three values.
3. Separate 1-dimensional curves are applied to each of these three values to get three new values.
4. A 3x3 matrix is applied to the resulting three values from step 3 to get three values that are assumed to be PCS (or Lab) values.

Note: This example is basic, and it is readily admitted that these elements could probably be combined, but it does show the programmable nature of using MPE tags.

4. Implementation

A reference implementation of a C++ programming library with applications that are capable of parsing, applying and validating profiles containing D2Bx/B2Dx tags can be found in the SampleICC project (http://sourceforge.net/projects/sampleicc/).