DRAFT: DRAFT: DRAFT:DRAFT:DRAFT

Changes from version 3.0 of June 1994 are indicated with **bold** typeface

Edited March 26, 1995

Revised May 8, 1995

International Color Consortium Profile Format

version 3.01

May 8, 1995

International Color Consortium Founding Members:
Adobe Systems Inc.
Agfa-Gevaert N.V.
Apple Computer, Inc.
Eastman Kodak Company
FOGRA (Honorary)
Microsoft Corporation
Silicon Graphics, Inc.
Sun Microsystems, Inc.
Taligent, Inc.

Other members:

Copyright Notice

For additional information on the ICC:

Todd Newman, Chairman International Color Consortium
Silicon Graphics, Inc
2011 N. Shoreline Blvd., MS 911
Mountain View, CA 94043

Phone: (415) 390-1614
Internet: tdn@sgi.com

Contents

1 Introduction

This specification describes the International Color Profile Format. The intent of this format is to provide a cross-platform device profile format. Such device profiles can be used to translate color data created on one device into another device's native color space. The acceptance of this format by operating system vendors allows end users to transparently move profiles and images with embedded profiles between different operating systems. For example, this allows a printer manufacturer to create a single profile for multiple operating systems.

A large number of companies and individuals from a variety of industries participated in very extensive discussions on these issues. Many of these discussions occurred under the auspices of FOGRA, a German graphic arts research institute during 1993. The present specification evolved from these discussions and the ColorSync(tm) 1.0 profile format.

This is a very complex set of issues and the organization of this document strives to provide a clear, clean, and unambiguous explanation of the entire format. To accomplish this, the overall presentation is from a top-down perspective, beginning with a summary overview and continuing down into more detailed specifications to a byte stream description of format.

2 Summary

2.1 Intended Audience

This specification is designed to provide developers and other interested parties a clear description of the profile format. A nominal understanding of color science is assumed, such as familiarity with the CIELAB color space, general knowledge of device characterizations and familiarity of at least one operating system level color management system.

2.2 Organizational Description of This Specification

This specification is organized into a number of major sections and appendices. Each section and subsection are numbered for easy reference. A brief introduction is followed by a detailed summary of the issues involved in this document including, but not limited to; International Color Consortium, device profiles, the profile connection space (PCS), tagged element structure, embedded profiles, profile classifications, color transformations, and color model arbitration. The third section on device profile descriptions provides a top level view of what tags are required for each type of profile classification and a brief description of the algorithmic models associated with these classes. The fourth section describes four additional color transformation formats; device link, color space conversion, abstract transformations, and named color transforms. The fifth section is a detailed algorithmic and intent description of all of the tagged elements described in the previous sections. The sixth section provides a byte stream definition of the structures that make up the tags in section five. The seventh section describes a collection of building block numeric types. The eighth section provides a detailed description of the actual profile format and tag sequence along with a description of the profile header. The ninth section describes the necessary details to embed profiles into PICT, TIFF, and EPS files.

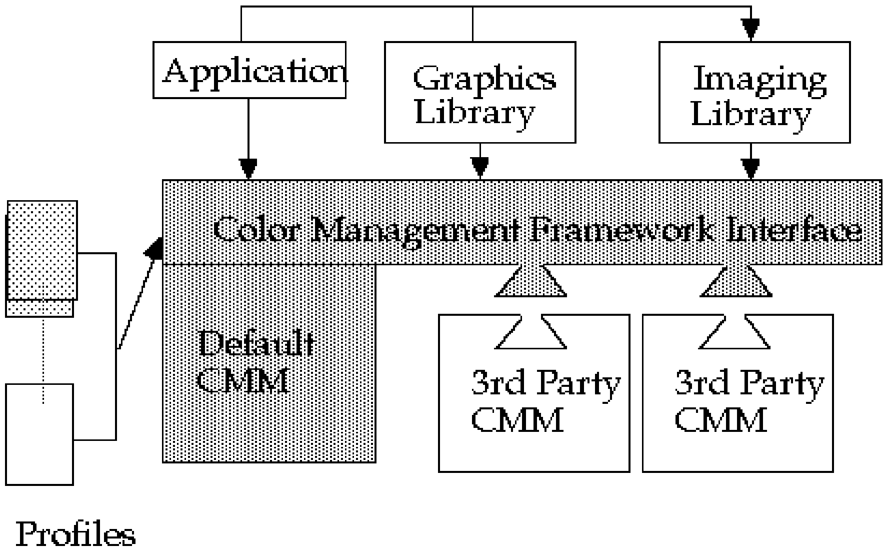
Appendix A provides cross-platform ANSI-C compatible header file example for each of the device profile and color transform formats. Appendix B specifies the 7-bit ASCII definition used in this specification. Appendix C provides a general description of the ScriptCode definition. Appendix D is a paper describing details of the profile connection space. Finally, appendix E provides a set of references for this document.

2.3 International Color Consortium

Considering the potential impact of this standard on various industries, a consortium is being formed that will administer this specification and the registration of tag signatures and descriptions. The founding members of this consortium include; Adobe Systems Inc., Agfa-Gevaert N.V., Apple Computer, Inc., Eastman Kodak Company, FOGRA (Honorary), Microsoft Corporation, Silicon Graphics, Inc., Sun Microsystems, Inc., and Taligent, Inc.. These companies have committed to fully support this specification in their operating systems, platforms and applications. In addition to these founding members, other companies that commit to support this specification are invited to join.

2.4 Device Profiles

Device profiles provide color management systems with the information necessary to convert color data between native device color spaces and device independent color spaces. This specification divides color devices into three broad classifications: input devices, display devices and output devices. For each device class, a series of base algorithmic models are described which performs the transformation between color spaces. These models provide a range of color quality and performance results. Each of the base models provides different trade-offs in memory footprint, performance and image quality. The necessary parameter data to implement these models described in the required portions on the appropriate device profile descriptions. This required data provides the information for the color management framework default color management module (CMM) to transform color information between native device color spaces. A representative architecture using these components is illustrated in the diagram below.



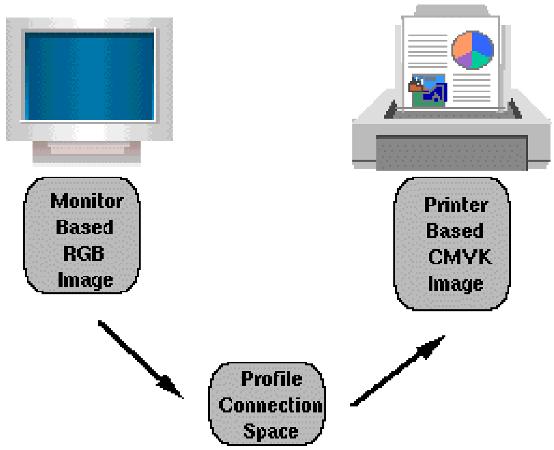
2.5 Color Spaces

The International Color Consortium Profile Format supports a variety of both device-dependent and device-independent color spaces divided into three basic families: 1) CIEXYZ based, 2) RGB based, and 3) CMY based. The CIE color spaces are defined in CIE publication 14.2 on Colorimetry. A subset of the CIEXYZ based spaces are also defined as exchange spaces. The device dependent spaces below are only representative and other device dependent color spaces may be used without needing to update the profile format specification or the software that uses it.

```
CIEXYZ          base CIE device-independent color space
        CIELAB

GRAY            monochrome device-dependent color space

RGB             base additive device-dependent color space
        HLS
        HSV

CMYK            base subtractive device-dependent color space
        CMY
```

2.6 Profile Connection Spaces

A key component of these profiles is a well-defined profile connection space. This space is the interface which provides an unambiguous connection between the input and output profiles as illustrated in the diagram below. The profile connection space is based on the CIE 1931 standard observer. This experimentally derived standard observer provides a very good representation of the human visual system color matching capabilities. Unlike device dependent color spaces, if two colors have the same CIE colorimetry they will match if viewed under the same conditions. Because the imagery is typically produced for a wide variety of viewing environments, it is necessary to go beyond simple application of the CIE system.



The profile connection space is defined as the CIE colorimetry which will produce the desired color appearance if rendered on a reference imaging media and viewed in a reference viewing environment. This reference corresponds to an ideal reflection print viewed in an ANSI standard viewing booth. Profile builders should read the Appendix "Connection Color Space for the Standard Profile Format" for further details.

The default measurement parameters for the profile connection space and all other color spaces defined in this specification are based on the ANSI CGATS.5-1993 standard, "Graphic technology - Spectral measurement and colorimetric computation for graphic arts images." Essentially this defines a standard illuminant of D50, the 1931 CIE standard observer, and 0/45 or 45/0 reflectance measurement geometry. The reference viewing condition is ANSI PH2.30-1989, which is a D50 graphic arts viewing environment.

One of the first steps in profile building involves measuring the colorimetry of a set of colors from some imaging media or display. If the imaging media or viewing environment differ from the reference, it will be necessary to adapt the measured colorimetry to that appropriate for the profile connection space. These adaptations account for such differences as white point chromaticity and luminance relative to an ideal reflector, maximum density, viewing surround, viewing illuminant, and flare. Currently, it is the responsibility of the profile builder to do this adaptation.

However, the possibility of allowing a variable illuminant in the PCS is under active consideration by the International Color Consortium. For this reason, a PCS illuminant field is in the profile header, but must be set to the CIE Illuminant D50 [X=0.9642, Y=1.0000, Z=0.8249].

The PCS is based on relative colorimetry. This is in comparison to absolute colorimetry. In absolute colorimetry colors are represented with respect to the illuminant, for example D50. In relative colorimetry, colors are represented with respect a combination of the illuminant and the media’s white, e.g. unprinted paper. The translation from relative colorimetry XYZ data, XYZr to absolute colorimetric data, XYZa, is given by

$$X_a = \left(\frac{X_{mw}}{X_i}\right) \bullet X_r$$

$$Y_a = \left(\frac{Y_{mw}}{Y_i}\right) \bullet Y_r$$

$$Z_a = \left(\frac{Z_{mw}}{Z_i}\right) \bullet Z_r$$

where XYZmw represents the media’s white and XYZi represents the illuminant white.

The actual media and actual viewing conditions will typically differ from the reference conditions. The profile specification defines tags which provide information about the actual white point and black point of a given media or display. These tags may be used by a CMM to provide functionality beyond that of the default. For example, an advanced CMM could use the tags to adjust colorimetry based on the Dmin of a specific media. A tag is also provided to describe the viewing environment. This information is useful in choosing a profile appropriate for the intended viewing method.

There are many ways of encoding CIE colorimetry. This specification provides three methods in order to satisfy conflicting requirements for accuracy and storage space. These encodings, an 8 bit/component CIELAB encoding, a 16 bit/component CIELAB encoding, and a 16 bit/component CIEXYZ encoding are described in the table below. The CIEXYZ space represents a linear transformation of the derived matching responses and the CIELAB space represents a transformation of the CIEXYZ space into one that is nearly perceptually uniform. This uniformness allows color errors to be equally weighted throughout its domain. While supporting multiple CIE encodings increases the complexity of color management, it provides immense flexibility in addressing different user requirements such as color accuracy and memory footprint.

The encoding is such that:

Interchange Space	Component	Actual Range	Encoded Range
CIE XYZ	X	0 -> 1.99997	0x0000 -> 0xffff
CIE XYZ	Y	0 -> 1.99997	0x0000 -> 0xffff
CIE XYZ	Z	0 -> 1.99997	0x0000 -> 0xffff
CIELAB (16 bit)	L*	0 -> 100.0	0x0000 -> 0xff00 *)
CIELAB (16 bit)	a*	-128.0 -> + 127.996	0x0000 -> 0xffff
CIELAB (16 bit)	b*	-128.0 -> + 127.996	0x0000 -> 0xffff
CIELAB (8 bit)	L*	0 -> 100.0	0x00 -> 0xff
CIELAB (8 bit)	a*	-128.0 -> + 127.0	0x00 -> 0xff
CIELAB (8 bit)	b*	-128.0 -> + 127.0	0x00 -> 0xff

An important point to be made is that the PCS is not necessarily intended for the storage of images. A separate series of "interchange color spaces" may be defined in a future version of this specification for this purpose. The design choices made for these spaces (colorimetric encoding, reference media, viewing conditions, etc.) might be different than that of the PCS.

2.7 Profile Element Structure

The profile structure is defined as a header followed by a tag table followed by a series of tagged elements that can be accessed randomly and individually. This collection of tagged elements provides three levels of information for developers: required data, optional data and private data. An element tag table provides a table of contents for the tagging information in each individual profile. This **table** includes a tag signature and the beginning address offset and size of the data for each individual tagged element. Signatures in this specification are defined as a four byte hexadecimal number. This tagging scheme allows developers to read in the element tag table and then randomly access and load into memory only the information necessary to their particular software application. Since some instances of profiles can be quite large, this provides significant savings in performance and memory. The detailed descriptions of the tags, along with their intent are included later in this specification.

The required tags provide the complete set of information necessary for the default CMM to translate color information between the profile connection space and the native device space. Each profile class determines which combination of tags is required. For example, a three dimensional lookup table is required for output devices, but not for display devices.

In addition to the required tags for each device profile, a number of optional tags are defined that can be used for enhanced color transformations. Examples of these tags include PostScript Level 2 support, calibration support, and others. In the case of required and optional tags, all of the signatures, an algorithmic description and intent are registered with the International Color Consortium. Private data tags allow CMM developers to add proprietary value to their profiles. By registering just the tag signature and tag type signature, developers are assured of maintaining their proprietary advantages while maintaining compatibility with the industry standard. However, the overall philosophy of this format is to maintain an open, cross-platform standard, therefore the use of private tags should be kept to an absolute minimum.

2.8 Embedded Profiles

In addition to providing a cross-platform standard for the actual disk-based profile format, this specification also describes the convention for embedding these profiles within graphics documents and images. Embedded profiles allow users to transparently move color data between different computers, networks and even operating systems without having to worry if the necessary profiles are present on the destination systems. The intention of embedded profiles is to allow the interpretation of the associated color data. Embedding specifications are described in section 10 of this document.

2.9 Profile Classifications

As stated previously, there are three basic classifications of device profiles: input, display and output profiles. Within each of these classes there can be a variety of subclasses, such as RGB scanners, CMYK scanners and many others. These basic classes have the following signatures :

```
’scnr’    input devices such as scanners and digital cameras,
’mntr’    display devices such as CRTs and LCDs,
’prtr’    output devices such as printers.
```

In addition to the three basic device profile classes, three additional color processing profiles are defined. These profiles provide a standard implementation for use by the CMM in general color processing or for the convenience of CMMs which may use these types to store calculated transforms. These three profile classes are: device link, color space conversion, and abstract profiles. Device link profiles provide a mechanism in which to save and store a series of device profiles and non-device profiles in a concatenated format as long as the series begins and ends with a device profile. This is extremely useful for workflow issues where a combination of device profiles and non-device profiles are used repeatedly. Color space conversion profiles are used as a convenient method for CMMs to convert between different non-device color spaces. Finally, the abstract color profiles provide a generic method for users to make subjective color changes to images or graphic objects by transforming the color data within the PCS.

These profiles have the following signatures :

```
'link'     device link profiles,
'spac'     color space conversion profiles,
'abst'     abstract profiles.
```

2.10 PostScript Level 2 Tags

The PostScript Level 2 tags are provided in order to control exactly the PostScript Level 2 operations that should occur for a given profile. These tags are only valid for PostScript Level 2 (and conceivably future versions of PostScript) devices, and are not generally supported in PostScript Level 1 devices. In addition, some of the tags may correspond to PostScript operations that are not supported in all PostScript Level 2 devices. Using such tags requires first checking for the available operators. All operators described in the PostScript Language Reference Manual, second edition, are available on all PostScript Level 2 devices. Documentation for extensions to PostScript Level 2 are available through Adobe's Developer Support Organization. In addition, guidelines for PostScript compatibility with this profile format are available. For details of such operator support, compatibility guidelines, the PostScript Level 2 device independent color model, or other PostScript related issues contact Adobe's Developer Support Organization.

In general, there is a straightforward relationship between the profile's header fields and tags, and these PostScript tags. It is anticipated that the various CMSs that support this profile format will also provide support for these optional PostScript tags. To verify such support contact the CMS vendors directly. In cases where such support is provided, and the desired model of operations is the same for PostScript processing as it is for CMS processing, these tags can be omitted, since all necessary information is in the profile itself. In the case where such CMS support is in question or processing different than that provided by an arbitrary CMS is desired, these tags can be populated to provide exact control over the PostScript processing. For example, if private tags are used in the profile to achieve a non-public type of processing on certain CMSs, such processing can be achieved on a PostScript device by populating the appropriate PostScript tags.

Some of the PostScript tags have a tag type of dataType. This is to match the properties of the communications channel to the data in these tags. Encoding binary data in dataType is recommended to save memory and/or reduce transmission times. Applications and drivers may convert it to ASCII Coded PostScript, Binary Coded PostScript, or Token Binary Coded PostScript or leave it in binary format to match the requirements of the communications channel. Applications and drivers are responsible for this potential conversion from binary data to channel compatible data. The data should be encoded as ASCII in dataType in those cases where the amount of data is relatively small or where the conversion from binary to channel compatible data is not available.

The PostScript contained in these tags is not self evaluating - it simply provides operands. These operands must be followed by operators like setcolorspace, setcolorrendering, and findcolorrendering.

2.11 Redundant Data Arbitration

There are several methods of color rendering described in the following structures that can function within a single CMM. If data for more than one method are included in the same profile, the following selection algorithm should be used by the software implementation: if an 8 bit or 16 bit lookup table is present, it should be used; if a lookup table is not present (and not required), the appropriate default modeling parameters are used. These default parameters are described later in this document.

2.13 Fixed Point Math

Many of the tag types contain fixed point numbers. Several references can be found (Knuth's MetaFonts, etc.) illustrating the preferability of fixed point math to pure floating point math in very structured circumstances.

2.14 Big-Endian Notation

All profile data must be encoded with the most significant byte first within the 16, 32, **and 64** bit quantities defined in the specification. This is commonly referred to as big-endian.

2.15 Rendering Intent

Rendering intent specifies the style of reproduction to be used during the evaluation of this profile in a sequence of profiles. It applies specifically to that profile in the sequence and not to the entire sequence. Typically, the user or application will set the rendering intent dynamically at runtime or embedding time.

3 Device Profile Descriptions

This section provides a top level view of what tags are required for each type of profile classification and a brief description of the algorithmic models associated with these classes. This begins with a subsection describing common tags required of all three device profiles, followed by a general description of each profile class and its required tags. A general description for each tag is included in this section.

Note that these descriptions assume two things; every profile contains a header, and may include additional tags beyond those listed as required in this section. The explicitly listed tags are those which are required in order to comprise a legal profile of each type.

In general, multi-dimensional tables refer to lookup tables with more than one input component.

The intent of requiring tags with profiles is to provide a common base level of functionality. If a custom CMM is not present, then the default CMM will have enough information to perform the requested color transformations. The particular models implied by the required data are also described below. While this data might not provide the highest level of quality obtainable with optional data and private data, the data provided is adequate for sophisticated device modeling.

Profile	Tag Name	Interpretation
Input Profile	AToB0Tag	none
Display Profile	AToB0Tag	none
Output Profile	**BToA0Tag**	perceptual rendering
Output Profile	**BToA1Tag**	colorimetric rendering
Output Profile	**BToA2Tag**	saturation rendering
Input Profile	grayTRCTag	depends on intent
Display Profile	grayTRCTag	additive
Output Profile	grayTRCTag	subtractive

3.1 Input Profile

This profile represents input devices such as scanners and digital cameras.

3.1.1 Monochrome Input Profiles

Tag Name	General Description
profileDescriptionTag	Structure containing invariant and localizable versions of the profile name for display
grayTRCTag	Gray tone reproduction curve (TRC)
mediaWhitePointTag	Media XYZ white point
copyrightTag	7 bit ASCII profile copyright information

The mathematical model implied by this data is $connection = grayTRC[device]$. This represents a simple tone reproduction curve adequate for most monochrome input devices. The connection values in this equation should represent the achromatic channel of the profile connection space. If the inverse of this is desired, then the following equation is used, $device = grayTRC^{-1}[connection]$

Multidimensional tables are not allowed to be included in monochrome profiles.

3.1.2 RGB Input Profiles

Tag Name	General Description
profileDescriptionTag	Structure containing invariant and localizable versions of the profile name for display
redColorantTag	Red colorant XYZ relative tristimulus values
greenColorantTag	Green colorant XYZ relative tristimulus values
blueColorantTag	Blue colorant XYZ relative tristimulus values
redTRCTag	Red channel tone reproduction curve
greenTRCTag	Green channel tone reproduction curve
blueTRCTag	Blue channel tone reproduction curve
mediaWhitePointTag	Media XYZ white point
copyrightTag	7 bit ASCII profile copyright information

The forward mathematical model implied by this data is :

$$linear_R = redTRC[device_r]$$
$$linear_G = greenTRC[device_g]$$
$$linear_B = blueTRC[device_b]$$
$$\begin{bmatrix} connection_X \\ connection_Y \\ connection_Z \end{bmatrix} = \begin{bmatrix} redColorant_X & greenColorant_X & blueColorant_X \\ redColorant_Y & greenColorant_Y & blueColorant_Y \\ redColorant_Z & greenColorant_Z & blueColorant_Z \end{bmatrix} \begin{bmatrix} linear_R \\ linear_G \\ linear_B \end{bmatrix}$$

This represents a simple linearization followed by a linear mixing model. The three tone reproduction curves linearize the raw values with respect to the luminance (Y) dimension of the CIEXYZ encoding of the profile connection space. The 3x3 matrix converts these linearized values into XYZ values for the CIEXYZ encoding of the profile connection space. The inverse model is given by the following equation,

$$\begin{bmatrix} linear_R \\ linear_G \\ linear_B \end{bmatrix} = \begin{bmatrix} redColorant_X & greenColorant_X & blueColorant_X \\ redColorant_Y & greenColorant_Y & blueColorant_Y \\ redColorant_Z & greenColorant_Z & blueColorant_Z \end{bmatrix}^{-1} \begin{bmatrix} connection_X \\ connection_Y \\ connection_Z \end{bmatrix}$$
$$device_r = redTRC^{-1}[linear_R]$$
$$device_g = greenTRC^{-1}[linear_G]$$
$$device_b = blueTRC^{-1}[linear_B]$$

Only the CIEXYZ encoding of the profile connection space can be used with matrix/TRC models. A multidimensional table tag must be included if the CIELAB encoding of the profile connection space is to be used.

3.1.3 CMYK Input Profiles

Tag Name	General Description
profileDescriptionTag	Structure containing invariant and localizable versions of the profile name for display
AToB0Tag	Device to PCS: 8 or 16 bit data
mediaWhitePointTag	Media XYZ white point
copyrightTag	7 bit ASCII profile copyright information

The AToB0Tag represents a device model described by the Lut8Type or Lut16Types. This tag provides the parameter data for an algorithm that includes a set of non-interdependent per-channel tone reproduction curves, a three dimensional lookup table and a set of non-interdependent per-channel linearization curves. The mathematical model implied by this data is described in detail in sections 6.4 and 6.5 that specify the general lookup table tag element structures.

This profile type can be used with a printer for space optimized embedding.

3.2 Display Profile

This profile represents display devices such as monitors.

3.2.1 Monochrome Display Profiles

Tag Name	General Description
profileDescriptionTag	Structure containing invariant and localizable versions of the profile name for display

grayTRCTag	Gray tone reproduction curve
mediaWhitePointTag	Media XYZ white point
copyrightTag	7 bit ASCII profile copyright information

The mathematical model implied by this data is $connection = grayTRC[device]$ This represents a simple tone reproduction curve adequate for most monochrome input devices. The connection values in this equation should represent the achromatic channel of the profile connection space. If the inverse of this is desired, then the following equation is used, $device = grayTRC^{-1}[connection]$

Multidimensional tables are not allowed to be included in monochrome profiles.

3.2.2 RGB Display Profiles

Tag Name	General Description
profileDescriptionTag	Structure containing invariant and localizable
versions of the	profile name for display
redColorantTag	Relative XYZ values of red phosphor
greenColorantTag	Relative XYZ values of green phosphor
blueColorantTag	Relative XYZ values of blue phosphor
redTRCTag	Red channel tone reproduction curve
greenTRCTag	Green channel tone reproduction curve
blueTRCTag	Blue channel tone reproduction curve
mediaWhitePointTag	Media XYZ white point
copyrightTag	7 bit ASCII profile copyright information

This model is based on a three non-interdependent per-channel tone reproduction curves to convert between linear and non-linear rgb values and a 3x3 matrix to convert between linear rgb values and relative XYZ values. The mathematical model implied by this data is :

$$linear_R = redTRC[device_r]$$
$$linear_G = greenTRC[device_g]$$
$$linear_B = blueTRC[device_b]$$
$$\begin{bmatrix} connection_X \\ connection_Y \\ connection_Z \end{bmatrix} = \begin{bmatrix} redColorant_X & greenColorant_X & blueColorant_X \\ redColorant_Y & greenColorant_Y & blueColorant_Y \\ redColorant_Z & greenColorant_Z & blueColorant_Z \end{bmatrix} \begin{bmatrix} linear_R \\ linear_G \\ linear_B \end{bmatrix}$$

This represents a simple linearization followed by a linear mixing model. The three tone reproduction curves linearize the raw values with respect to the luminance (Y) dimension of the CIEXYZ encoding of the profile connection space. The 3x3 matrix converts these linearized values into XYZ values for the CIEXYZ encoding of the profile connection space. The inverse model is given by the following equation,

$$\begin{bmatrix} linear_R \\ linear_G \\ linear_B \end{bmatrix} = \begin{bmatrix} redColorant_X & greenColorant_X & blueColorant_X \\ redColorant_Y & greenColorant_Y & blueColorant_Y \\ redColorant_Z & greenColorant_Z & blueColorant_Z \end{bmatrix}^{-1} \begin{bmatrix} connection_X \\ connection_Y \\ connection_Z \end{bmatrix}$$
$$device_r = redTRC^{-1}[linear_R]$$
$$device_g = greenTRC^{-1}[linear_G]$$
$$device_b = blueTRC^{-1}[linear_B]$$

Only the CIEXYZ encoding of the profile connection space can be used with matrix/TRC models. A multidimensional table tag must be included if the CIELAB encoding of the profile connection space is to be used.

3.3 Output Profile

This profile represents output devices such as printers and film recorders. **The LUT tags that are required by the printer profiles contain either the 8 bit or the 16 bit LUTs as described in the LUT tags**. The LUT algorithm for profile connection space to device space transformations process data sequentially through a matrix, input tables, a color LUT, and output tables.

3.3.1 Monochrome Output Profiles

Tag Name	General Description
profileDescriptionTag	Structure containing invariant and localizable versions of the profile name for display
grayTRCTag	Gray tone reproduction curve
mediaWhitePointTag	Media XYZ white point
copyrightTag	7 bit ASCII profile copyright information

The tone reproduction curve provides the necessary information to convert between a single device channel and the CIEXYZ encoding of the profile connection space.

The mathematical model implied by this data is $connection = grayTRC[device]$. This represents a simple tone reproduction curve adequate for most monochrome input devices. The connection values in this equation should represent the achromatic channel of the profile connection space. If the inverse of this is desired, then the following equation is used, $device = grayTRC^{-1}[connection]$.

Multidimensional tables are not allowed to be included in monochrome profiles.

3.3.2 RGB and CMYK Output Profiles

Tag Name	General Description
profileDescriptionTag	Structure containing invariant and localizable versions of the profile name for display
AToB0Tag	Device to PCS: 8 or 16 bit data: intent of 0
BToA0Tag	PCS to Device space: 8 or 16 bit data: intent of 0
gamutTag	Out of Gamut: 8 or 16 bit data
AToB1Tag	Device to PCS: 8 or 16 bit data: intent of 1
BToA1Tag	PCS to Device space: 8 or 16 bit data: intent of 1
AToB2Tag	Device to PCS: 8 or 16 bit data: intent of 2
BToA2Tag	PCS to Device space: 8 or 16 bit data: intent of 2
mediaWhitePointTag	Media XYZ white point
copyrightTag	7 bit ASCII profile copyright information

These tags represent a device model described by the Lut8Type or Lut16Types. The intent values described in these tags directly correlate to the value of the rendering intent header flag of the source profile in the color modeling session.

Rendering Intent	Value

perceptual	0
relative colorimetric	1
saturation	2
absolute colorimetric	3

Each of the first three intents are associated with a specific tag. The fourth intent, absolute colorimetry, is obtained by modifying the relative colorimetric intent tag based on the values which are in the mediaWhitePointTag. It is permissible to reference the same tag for all of these intents and to use the relative colorimetric intent tag when absolute colorimetry is specified. This decision is left to the profile builder.

In essence, each of these tags provides the parameter data for an algorithm that includes a 3x3 matrix, a set of non-interdependent per-channel tone reproduction curves, a three dimensional lookup table and a set of non-interdependent per-channel linearization curves. The algorithmic details of this model and the intent of each tag is given later in sections 6.4 and 6.5 that specify the general lookup table tag element structures.

4 Additional Profile Formats

4.1 DeviceLink Profile

This profile represents a one-way link or connection between devices. It does not represent any device model nor can it be embedded into images.

Tag Name	General Description
profileDescriptionTag	Structure containing invariant and localizable versions of the profile name for display
AToB0Tag	Actual transformation parameter structure this is an exclusive or) 8 or 16 bit data
profileSequenceDescTag	An array of descriptions of the profile sequence
copyrightTag	7 bit ASCII profile copyright information

The single AToB0Tag may contain any of the four possible rendering intents. The rendering intent used is indicated in the header of the profile.

The AToB0Tag represents a device model described by the Lut8Type or Lut16Types. This tag provides the parameter data for an algorithm that includes a 3x3 matrix, a set of non-interdependent per-channel tone reproduction curves, a three dimensional lookup table and a set of non-interdependent per-channel linearization curves. The algorithmic details of this model and the intent of each tag is given later in sections 6.4 and 6.5 that specify the general lookup table tag element structures. This is a pre-evaluated transform that cannot be undone.

The color space of data in the DeviceLink profile will be the same as the color space of data of the first profile in the sequence. The profile connection space will be the same as the color space of data of the last profile in the sequence.

4.2 ColorSpaceConversion Profile

This profile provides the relevant information to perform a color space transformation between the non-device color spaces and the PCS. It does not represent any device model.

Tag Name	General Description
profileDescriptionTag	Structure containing invariant and localizable versions of the profile name for display
BToA0Tag	Inverse transformation parameter structure (this is an exclusive or) 8 or 16 bit data
AToB0Tag	Actual transformation parameter structure (this is an exclusive or) 8 or 16 bit data
mediaWhitePointTag	Media XYZ white point
copyrightTag	7 bit ASCII profile copyright information

The AToB0Tag and BToA0Tag represent a model described by the Lut8Type or Lut16Types. This tag provides the parameter data for an algorithm that includes a 3x3 matrix, a set of non-interdependent per-channel tone reproduction curves, a three dimensional lookup table and a set of non-interdependent per-channel linearization curves. The algorithmic details of this model and the intent of each tag is given later in sections 6.4 and 6.5 that specify the general lookup table tag element structures.

4.3 Abstract Profile

This profile represents abstract transforms and does not represent any device model. Color transformations using abstract profiles are performed from PCS to PCS.

Tag Name	General Description
profileDescriptionTag	Structure containing invariant and localizable versions of the profile name for display
AToB0Tag	Actual transformation parameter structure (this is an exclusive or) 8 or 16 bit data
mediaWhitePointTag	Media XYZ white point
copyrightTag	7 bit ASCII profile copyright information

The AToB0Tag represents a PCS to PCS model described by the Lut8Type or Lut16Types. This tag provides the parameter data for an algorithm that includes a 3x3 matrix, a set of non-interdependent per-channel tone reproduction curves, a three dimensional lookup table and a set of non-interdependent per-channel linearization curves. The algorithmic details of this model and the intent of each tag is given later in sections 6.4 and 6.5 that specify the general lookup table tag element structures.

5 Tag Descriptions

This section specifies the individual tags used to create all possible portable profiles in the International Color Profile Format. The appropriate tag typing is indicated with each individual tag description. Note that the signature indicates only the type of data and does not imply anything about the use or purpose for which the data is intended.

In addition to the tags listed below, any of the previously defined tags in sections 3 and 4 on device profiles can also be used as optional tags if they are not used in the required set for a particular profile.

```
Tag Name                 General Description

AToB0Tag                 Multidimensional transformation structure
AToB1Tag                 Multidimensional transformation structure
AToB2Tag                 Multidimensional transformation structure
blueColorantTag          Relative XYZ values of blue phosphor
blueTRCTag               Blue channel tone reproduction curve
BToA0Tag                 Multidimensional transformation structure
BToA1Tag                 Multidimensional transformation structure
BToA2Tag                 Multidimensional transformation structure
calibrationDateTimeTag   Profile calibration date and time
charTargetTag            Characterization target such as IT8/7.2
copyrightTag             7 bit ASCII profile copyright information
deviceMfgDescTag         displayable description of device manufacturer
deviceModelDescTag       displayable description of device model
gamutTag                 Out of Gamut :   8 or 16 bit data
grayTRCTag               Gray tone reproduction curve
greenColorantTag         Relative XYZ values of green phosphor
greenTRCTag              Green channel tone reproduction curve
luminanceTag             Absolute luminance for emissive device
measurementTag           Alternative measurement specification information
mediaBlackPointTag       Media XYZ black point
mediaWhitePointTag       Media XYZ white point
namedColorTag            Dictionary for converting between named colors and
                         interchange or device color spaces
preview0Tag              Preview transformation : 8 or 16 bit data
preview1Tag              Preview transformation : 8 or 16 bit data
preview2Tag              Preview transformation : 8 or 16 bit data
profileDescriptionTag    profile description for display
profileSequenceDescTag   profile sequence from source to destination
ps2CRD0Tag               PostScript Level 2 color rendering dictionary: perceptual
ps2CRD1Tag               PostScript Level 2 color rendering dictionary: colorimetric
ps2CRD2Tag               PostScript Level 2 color rendering dictionary: saturation
ps2CRD3Tag               PostScript Level 2 color rendering dictionary: absolute
ps2CSATag                PostScript Level 2 color space array
ps2RenderingIntentTag    PostScript Level 2 Rendering Intent
redColorantTag           Relative XYZ values of red phosphor
redTRCTag                Red channel tone reproduction curve
screeningDescTag         Screening attributes description
screeningTag             Screening attributes such as frequency, angle and spot
technologyTag            Device technology information such as LCD, CRT,
                         Dye Sublimation, etc.
ucrbgTag                 Under color removal curve
viewingCondDescTag       Specifies viewing condition description
viewingConditionsTag     Specifies viewing condition parameters
```

5.1 AToB0Tag

Tag Types : lut8Type xor lut16Type
Tag Signature : ’A2B0’ 0x41324230

Device to PCS : 8 bit or 16 bit data. The processing mechanisms are described in sections 6.4 and 6.5.

5.2 AToB1Tag

Tag Types : lut8Type xor lut16Type
Tag Signature : ’A2B1’ 0x41324231

Device to PCS : 8 bit or 16 bit data. The processing mechanisms are described in sections 6.4 and 6.5.

5.3 AToB2Tag

Tag Types : lut8Type xor lut16Type
Tag Signature : ’A2B2’ 0x41324232

Device to PCS : 8 bit or 16 bit data. The processing mechanisms are described in sections 6.4 and 6.5.

5.4 blueColorantTag

Tag Type : XYZTypes
Tag Signature : ’bXYZ’ 0x6258595A

The relative XYZ values of blue phosphor or colorant.

5.5 blueTRCTag

Tag Type : curveType
Tag Signature : ’bTRC’ 0x62545243

Blue channel tone reproduction curve. The first element represents no colorant (white) or phosphors (black) and the last element represents 100 percent colorant (blue) or 100 percent phosphor (blue).

The count value specifies the number of entries in the curve table except as follows:

```
when count is 0,     then a linear response (slope equal to 1.0) is assumed,
when count is 1,     the data entry is interpreted as a simple gamma value
                     (ranging from 0 to 8 in fixed unsigned 8.8 format), and
when count is 2,     the entries are intepreted as the beginning and end
                     points of a line.
```

Gamma is interpreted canonically and NOT as an inverse.

5.6 BToA0Tag

Tag Types : lut8Type xor lut16Type
Tag Signature : ’B2A0’ 0x42324130

PCS to Device space : 8 bit or 16 bit data. The processing mechanisms are described in sections 6.4 and 6.5.

5.7 BToA1Tag

Tag Types : lut8Type xor lut16Type
Tag Signature : ’B2A1’ 0x42324131

PCS to Device space : 8 bit or 16 bit data. The processing mechanisms are described in sections 6.4 and 6.5.

5.8 BToA2Tag

Tag Types : lut8Type xor lut16Type
Tag Signature : ’B2A2’ 0x42324132

PCS to Device space : 8 bit or 16 bit data. The processing mechanisms are described in sections 6.4 and 6.5.

5.9 calibrationDateTimeTag

Tag Type : dateTimeType
Tag Signature : ’calt’ 0x63616C74

Profile calibration date and time. Initially, this tag matches the contents of the creationDateTime header flag. This allows applications and utilities to verify if this profile matches a vendor’s profile and how recently calibration has been performed.

5.10 charTargetTag

Tag Type : textType
Tag Signature : ’targ’ 0x74617267

This tag contains the measurement data for a characterization target such as IT8.7/2. This tag is provided so that distributed utilities can create transforms "on the fly" or check the current performance against the original device performance. The tag embeds the exact data file format defined in the ANSI or ISO standard which is applicable to the device being characterized.

Examples are the data formats described in ANSI IT8.7/1-1993 section 4.10, ANSI IT8.7/2-1993 section 4.10 and ANSI IT8.7/3 section 4.10. Each of these file formats contains an identifying character string as the first few bytes of the format, allowing an external parser to determine which data file format is being used. This provides the facilities to include a wide range of targets using a variety of measurement specifications in a standard manner.

Note: The IT8 specifications do not currently have a keyword which identifies a data set as being reference data as opposed to device response data. An addition to enable this additional data set is being considered by the IT8 committee.

5.11 copyrightTag

Tag Type : textType
Tag Signature : ’cprt’ 0x63707274

This tag contains the 7 bit ASCII text copyright information for the profile.

5.12 deviceMfgDescTag

Tag Type : textDescriptionType
Tag Signature : ’dmnd’ 0x646D6E64

Structure containing invariant and localizable versions of the device manufacturer for display. The content of this structure is described in section 6.9.

5.13 deviceModelDescTag

Tag Type : textDescriptionType
Tag Signature : ’dmdd’ 0x646D6464

Structure containing invariant and localizable versions of the device manufacturer for display. The content of this structure is described in section 6.9.

5.14 gamutTag

Tag Types : lut8Type xor lut16Type
Tag Signature : ’gamt’ 0x67616D74

Out of Gamut tag : 8 bit or 16 bit data. The processing mechanisms are described in sections 6.4 and 6.5. **The CLUT tag has a single output. If the output value value is 0, the input color is in gamut. If the output is non zero, the input color is out of gamut,with the number "n+1" being at least as far out of gamut than the number "n".**

5.15 grayTRCTag

Tag Type : curveType
Tag Signature : ’kTRC’ 0x6B545243

Gray tone reproduction curve. The tone reproduction curve provides the necessary information to convert between a single device channel and the CIEXYZ encoding of the profile connection space. The first element represents no colorant (white) or phosphors (black) and the last element represents 100 percent colorant (black) or 100 percent phosphor (white).

The count value specifies the number of entries in the curve table except as

```
follows:
when count is 0,      then a linear response (slope equal to 1.0) is assumed,
when count is 1,      then the data entry is interpreted as a simple gamma value
```

```
                          (ranging from 0 to 8 in fixed unsigned 8.8  format), and
when count is 2,          the entries are intepreted as the beginning and end points of
                           a line.
```

Gamma is interpreted canonically and NOT as an inverse.

5.16 greenColorantTag

Tag Type : XYZType
Tag Signature : ’gXYZ’ 0x6758595A

Relative XYZ values of green phosphor or colorant.

5.17 greenTRCTag

Tag Type : curveType
Tag Signature : ’gTRC’ 0x67545243

Green channel tone reproduction curve. The first element represents no colorant (white) or phosphors (black) and the last element represents 100 percent colorant (green) or 100 percent phosphor (green).

The count value specifies the number of entries in the curve table except as follows:

```
when count is 0,     then a linear response (slope equal to 1.0) is assumed,
when count is 1,     then the data entry is interpreted as a simple gamma value
                     (ranging from 0 to 8 in fixed unsigned 8.8 format), and
when count is 2,     the entries are intepreted as the beginning and end points
                     of a line.
```

Gamma is interpreted canonically and NOT as an inverse.

5.18 luminanceTag

Tag Types : XYZType
Tag Signature : ’lumi’ 0x6C756D69

The absolute luminance of devices is in candelas per meter squared as described by the Y channel. The X and Z channels are ignored in all cases.

5.19 measurementTag

Tag Type : measurementType
Tag Signature : ’meas’ 0x6D656173

Alternative measurement specification such as a D65 illuminant instead of the default D50.

5.20 mediaBlackPointTag

Tag Type : XYZType
Tag Signature : ’bkpt’ 0x626b7074

This tag specifies the media black point and is used for generating absolute colorimetry. It is referenced to the profile connection space. If this tag is not present, it is assumed to be (0,0,0).

5.21 mediaWhitePointTag

Tag Type : XYZType
Tag Signature : ’wtpt’ 0x77747074

This tag specifies the media white point and is used for generating absolute colorimetry. It is referenced to the profile connection space. If this tag is not present, it is assumed to be the same as the illuminant in the header.

5.22 namedColorTag

Tag Type : namedColorType
Tag Signature : ’ncol’ 0x6E636F6C

Named color reference transformation for converting between named color sets and the profile connection space or device color spaces.

5.23 preview0Tag

Tag Types : lut8Type xor lut16Type
Tag Signature : ’pre0’ 0x70726530

Preview transformation from PCS to device space and back to the PCS : 8 bit or16 bit data. The processing mechanisms are described in sections 6.4 and 6.5.

5.24 preview1Tag

Tag Types : lut8Type xor lut16Type
Tag Signature : ’pre1’ 0x70726531

Preview transformation from the PCS to device space and back to the PCS : 8 bit or 16 bit data. The processing mechanisms are described in sections 6.4 and 6.5.

5.25 preview2Tag

Tag Types : lut8Type xor lut16Type
Tag Signature : ’pre2’ 0x70726532

Preview transformation from PCS to device space and back to the PCS : 8 bit or 16 bit data. The processing mechanisms are described in sections 6.4 and 6.5.

5.26 profileDescriptionTag

Tag Type : textDescriptionType
Tag Signature : ’desc’ 0x64657363

Structure containing invariant and localizable versions of the profile description for display. This content of this structure is described in section 6.9. This invariant description has no fixed relationship to the actual profile disk file name.

5.27 profileSequenceDescTag

Tag Type : profileSequenceDescType
Tag Signature : ’pseq’ 0x70736571

Structure containing a description of the profile sequence from source to destination, typically used with the devicelink profile. This content of this structure is described in section 6.8.

5.28 ps2CRD0Tag

Tag Type : dataType
Tag Signature : ’psd0’ 0x70736430

PostScript Level 2 Type 1 color rendering dictionary (CRD) for the Perceptual rendering intent. This tag provides the dictionary operand to the setcolorrendering operator. This tag can be used in conjunction with the setcolorrendering operator on any PostScript Level 2 device.

See section 2.10 for the relationship between ICC profile data and PostScript Tags.

5.29 ps2CRD1Tag

Tag Type : dataType
Tag Signature : ’psd1’ 0x70736431

PostScript Level 2 Type 1 CRD for the RelativeColorimetric rendering intent. This tag provides the dictionary operand to the setcolorrendering operator. This tag can be used in conjunction with the setcolorrendering operator on any PostScript Level 2 device.

See section 2.10 for the relationship between ICC profile data and PostScript Tags.

5.30 ps2CRD2Tag

Tag Type : dataType

Tag Signature : ’psd2’ 0x70736432

PostScript Level 2 Type 1 CRD for the Saturation rendering intent. This tag provides the dictionary operand to the setcolorrendering operator. This tag can be used in conjunction with the setcolorrendering operator on any PostScript Level 2 device.

See section 2.10 for the relationship between ICC profile data and PostScript Tags.

5.31 ps2CRD3Tag

Tag Type : dataType
Tag Signature : ’psd3’ 0x70736433

PostScript Level 2 Type 1 CRD for the AbsoluteColorimetric rendering intent. This tag provides the dictionary operand to the setcolorrendering operator. This tag can be used in conjunction with the setcolorrendering operator on any PostScript Level 2 device.

See section 2.10 for the relationship between ICC profile data and PostScript Tags.

5.32 ps2CSATag

Tag Type : dataType
Tag Signature : ’ps2s’ 0x70733273

PostScript Level 2 color space array. This tag provides the array operand to the setcolorspace operator. For color spaces that fit within the original PostScript Level 2 device independent color model no operator verification need be performed. For color spaces that fit only within extensions to this model, operator verification is first required. An example of this would be for Calibrated CMYK input color spaces which are supported via an extension. In such cases where the necessary PostScript Level 2 support is not available,PostScript Level 1 color spaces, such as DeviceCMYK, can be used, or the colors can be converted on the host using a CMS. In the latter case, the PostScript Level 1 color operators are used to specify the device dependent (pre-converted) colors. The PostScript contained in this tag expects the associated color values instantiated either through setcolor or image to be inthe range [0, 1].

See section 2.10 for the relationship between ICC profile data and PostScriptTags.

5.33 ps2RenderingIntentTag

Tag Type : dataType

Tag Signature : ’ps2i’ 0x70733269

PostScript Level 2 rendering intent. This tag provides the operand to the findcolorrendering operator. findcolorrendering is not necessarily supported on all PostScript Level 2 devices, hence its existence must first be established. Standard values for ps2RenderingIntentTag are RelativeColorimetric, AbsoluteColorimetric, Perceptual, and Saturation. These intents are meant to correspond to the rendering intents of the profile’s header.

See section 2.10 for the relationship between ICC profile data and PostScript Tags.

5.34 redColorantTag

Tag Type : XYZType
Tag Signature : ’rXYZ’ 0x7258595A

Relative XYZ values of red phosphor or colorant.

5.35 redTRCTag

Tag Type : curveType
Tag Signature : ’rTRC’ 0x72545243

Red channel tone reproduction curve. The first element represents no colorant (white) or phosphors (black) and the last element represents 100 percent colorant (red) or 100 percent phosphor (red).

The count value specifies the number of entries in the curve table except as follows:

```
when count is 0,     then a linear response (slope equal to 1.0) is assumed,
when count is 1,     then the data entry is interpreted as a simple gamma value
                     (ranging from 0 to 8 in fixed unsigned 8.8 format), and
when count is 2,     the entries are intepreted as the beginning and end points of
                     a line.
```

Gamma is interpreted canonically and NOT as an inverse.

5.36 screeningDescTag

Tag Type : textDescriptionType
Tag Signature : ’scrd’ 0x73637264

Structure containing invariant and localizable versions of the screening conditions. This content of this structure is described in section 6.9.

5.37 screeningTag

Tag Type : screeningType
Tag Signature : ’scrn’ 0x7363726E

This tag contains screening information for a variable number of channels.

5.38 technologyTag

Tag Type : signatureType
Tag Signature : ’tech’ 0x74656368

Device technology information such as CRT, Dye Sublimation, etc.

The encoding is such that :

```
Technology                        signature        hex signature

Film Scanner                      ’fscn’           0x6673636E
Reflective Scanner                ’rscn’           0x7273636E
Ink Jet Printer                   ’ijet’           0x696A6574
Thermal Wax Printer               ’twax’           0x74776178
Electrophotographic Printer       ’epho’           0x6570686F
Electrostatic Printer             ’esta’           0x65737461
Dye Sublimation Printer           ’dsub’           0x64737562
Photographic Paper Printer        ’rpho’           0x7270686F
Film Writer                       ’fprn’           0x6670726E
Video Monitor                     ’vidm’           0x76696460
Video Camera                      ’vidc’           0x76696463
Projection Television             ’pjtv’           0x706A7476
Cathode Ray Tube Display          ’CRT ’           0x43525420
Passive Matrix Display            ’PMD ’           0x504D4420
Active Matrix Display             ’AMD ’           0x414D4420
Photo CD                          ’KPCD’           0x4B504344
PhotoImageSetter                  ’imgs’           0x696D6773
Gravure                           ’grav’           0x67726176
Offset Lithography                ’offs’           0x6F666673
Silkscreen                        ’silk’           0x73696C6B
Flexography                       ’flex’           0x666C6578
```

5.39 ucrbgTag

Tag Type : ucrbgType
Tag Signature : ’bfd ’ 0x62666420

Under color removal and black generation specification. This tag contains curve information for both under color removal and black generation in addition to a general description. This content of this structure is described in section 6.15.

5.40 viewingCondDescTag

Tag Type : textDescriptionType
Tag Signature : ’vued’ 0x76756564

Structure containing invariant and localizable versions of the viewing conditions. This content of this structure is described in section 6.9.

5.41 viewingConditionsTag

Tag Type : viewingConditionsType
Tag Signature : ’view’ 0x76696577

Viewing conditions parameters.

6 Tag Type Definitions

This section specifies the type and structure definitions used to create all of the individual tagged elements in International Color Profile Format. The data type description identifiers are indicated at the right margin of each data or structure definition. An effort was made to make sure one-byte, two-byte and four-byte data lies on one-byte, two-byte and four-byte boundaries respectively. **T**his required occasionally including extra spaces indicated with "reserved for padding" in some tag type definitions. Value 0 is defined to be of "unknown value" for all enumerated data structures.

All tags, including private tags, have as their first four bytes (0-3) a tag signature (a 4 byte character sequence) to identify to profile readers what kind of data is contained within a tag. This encourages tag type reuse and allows profile parsers to reuse code when tags use common tag types. The second four bytes (4-7) are reserved for future expansion and must be set to 0 in this version of the specification. Each new tag signature and tag type signature must be registered with the International Color Consortium in order to prevent signature collisions.

Where not specified otherwise, the low 16 bits of all 32 bit flags in the type descriptions below are reserved for use by the International Color Consortium.

When 7 bit ASCII text representation is specified in types below, each individual character is encoded in 8 bits with the high bit set to zero. The details are presented in **Appendix B**.

6.1 curveType

The curveType contains a 4 byte count value and a one-dimensional table of 2 byte values. The byte stream is given below.

```
byte(s)       content

0-3           ’curv’ (0x63757276) type descriptor
4-7           reserved, must be set to 0
8-11          count value specifying number of entries that follow
12-end        actual curve values starting with the zeroth entry and ending
              with the entry count-1.
```

Unless otherwise specified (see sections 5.5, 5.15, 5.17, 5.35) curve values are in the range [0.0, 1.0]. These 16 bit unsigned integers in the range 0 to 216 - 1 (65535) linearly map to curve values in the

interval [0.0, 1.0] .

6.2 dataType

The dataType is a simple data containing structure that contains either 7 bit ASCII or binary data, i.e.. textType data or transparent 8-bit bytes. The length of the string can easily be obtained from the element size portion of the tag itself. If this type is used for ASCII data, it must be terminated with a 0x00 byte.

byte(s)	content
0-3	'data' (0x64617461) type descriptor
4-7	reserved, must be set to 0
8-11	data flag, 0x00000000 represents ASCII data, 0x00000001 represents binary data, other values are reserved for future use
12-n	a string of count ASCII characters or count bytes (where count is derived from the element size portion of the tag itself

6.3 dateTimeType

This dateTimeType is a 12 byte value representation of the time and date. The actual values are encoded as a dateTimeNumber described in section 7.

byte(s)	content	Encoded As...
0-3	'dtim' (0x6474696D) type descriptor	
4-7	reserved, must be set to 0	
8-19	date and time	dateTimeNumber

6.4 lut16Type

This structure converts an input color into an output color using tables with 16 bit precision. This type contains four processing elements: a 3 by 3 matrix (only used when the input color space has three components), a set of one dimensional input lookup tables, a multidimensional lookup table, and a set of one dimensional output tables. Data is processed using these elements via the following sequence:

(matrix) -> (1d input tables) -> (multidimensional lookup table) -> (1d output tables).

++++++

byte(s)	content	Encoded As...
0-3	'mft2' (0x6D667432) [multi-function table with 2 byte precision] type descriptor	
4-7	reserved, must be set to 0	

8	Number of Input Channels	uInt8Number
9	Number of Output Channels	uInt8Number
10	Number of CLUT grid points (identical for each side)	uInt8Number
11	Reserved for padding (required to be 0x00)	
12-15	Encoded e00 parameter	s15Fixed16Number
16-19	Encoded e01 parameter	s15Fixed16Number
20-23	Encoded e02 parameter	s15Fixed16Number
24-27	Encoded e10 parameter	s15Fixed16Number
28-31	Encoded e11 parameter	s15Fixed16Number
32-35	Encoded e12 parameter	s15Fixed16Number
36-39	Encoded e20 parameter	s15Fixed16Number
40-43	Encoded e21 parameter	s15Fixed16Number
44-47	Encoded e22 parameter	s15Fixed16Number
48-49	Number of input table entries	uInt16Number
50-51	Number of output table entries	uInt16Number
52-n	input tables	
n+1-m	CLUT values	
m+1-o	output tables	
+++++		

The input, output and CLUT tables are arrays of 16 bit unsigned values. Each input table consists of a minimum of two and a maximum of 4096 two byte integers. Each input table entry is appropriately normalized to the range 0-65535. The inputTable is of size InputChannels * inputTableEntries * 2 bytes. When stored in this tag, the one-dimensional lookup tables are assumed to be packed one after another in the order described below.

The matrix is organized as an 3 by 3 array. The dimension corresponding to the matrix rows varies least rapidly and the dimension corresponding to the matrix columns varies most rapidly and is shown in matrixform below. Each matrix entry is a four byte number with one sign bit, 15 integer bits, and 16 fractional bits.

$$\begin{vmatrix} e00 & e01 & e02 \\ e10 & e11 & e12 \\ e20 & e21 & e22 \end{vmatrix}$$

When using the matrix of an output profile, and the input data is XYZ, we have

$$\begin{vmatrix} X' \\ Y' \\ Z' \end{vmatrix} = \begin{vmatrix} XX & XY & XZ \\ YX & YY & YZ \\ ZX & ZY & ZZ \end{vmatrix} \begin{vmatrix} X \\ Y \\ Z \end{vmatrix}$$

Each input X, Y or Z is an unsigned 1.15 number and each matrix entry is a signed 15.16 number. Therefore, each multiplication in the matrix multiply is 1.15 * s15.16 = s16.31 and the final sum is also s16.31. From this sum we take bits 31-16 as the unsigned integer result for X’, Y’, or Z’. These are then used as the inputs to the input tables of the multidimensionalLUT. This normalization is used since the number of fractional bits in the input data must be maintained by the matrix operation.

The matrix is mandated to be an identity matrix unless the input to the matrix is in the XYZ color space.

Each CLUT is organized as an n-dimensional array with a given number of grid points in each dimension, where n is the number of input channels(input tables) in the transform. The dimension corresponding to the first input channel varies least rapidly and the dimension corresponding to the last input channel varies most rapidly. Each grid point value contains m two byte integers, where m is the number of output functions. The first sequential two byte integer of the entry contains the function value for the first output function, the second sequential two byte integer of the entry contains the function value for the second output function, and so on until all the output functions have been supplied. The equation for computing the size of the CLUT is :

$$CLUTsize = LUTDimensions^{InputChannels} \bullet OutputChannels \bullet 2bytes$$

Each output table consists of **a minimum of two and a maximum of 4096** two byte integers. The outputTable is of size OutputChannels * outputTableEntries * 2 bytes. When stored in this tag, the one-dimensional lookup tables are assumed to be packed one after another in the order described in the following paragraph.

When using this type, it is necessary to assign each color space component to an input and output channel. The following table shows these assignments. The channels are numbered according to the order in which their table occurs. Note that additional color spaces can be added simply by defining the signature, channel assignments, and creating the tables.

Color Space	Channel 1	Channel 2	Channel 3	Channel 4
'XYZ'	X	Y	Z	
'Lab'	L	a	b	
'Luv'	L	u	v	
'YCbr'	Y	Cb	Cr	
'Yxy'	Y	x	y	
'RGB'	R	G	B	
'GRAY'	K			
'HSV'	H	S	V	
'HLS'	H	L	S	
'CMYK'	C	M	Y	K
'CMY'	C	M	Y	

6.5 lut8Type

This structure converts an input color into an output color using tables of 8 bit precision. This type contains four processing elements: a 3 by 3 matrix (only used when the input color space has three components), a set of one dimensional input lookup tables, a multidimensional lookup table, and a set of one dimensional output tables. Data is processed using these elements via the following sequence:

(matrix) -> (1d input tables) -> (multidimensional lookup table) -> (1d output tables).

byte(s)	content	Encoded As...
0-3	'mft1' (0x6D667431) [multi-function table with 1 byte precision] type descriptor	
4-7	reserved, must be set to 0	
8	Number of Input Channels	uInt8Number
9	Number of Output Channels	uInt8Number
10	Number of CLUT grid points (identical	

	for each side)	uInt8Number
11	Reserved for padding (fill with 0x00)	
12-15	Encoded e00 parameter	s15Fixed16Number
16-19	Encoded e01 parameter	s15Fixed16Number
20-23	Encoded e02 parameter	s15Fixed16Number
24-27	Encoded e10 parameter	s15Fixed16Number
28-31	Encoded e11 parameter	s15Fixed16Number
32-35	Encoded e12 parameter	s15Fixed16Number
40-43	Encoded e21 parameter	s15Fixed16Number
44-47	Encoded e22 parameter	s15Fixed16Number
48-m	input tables	
m+1-n	CLUT values	
n+1-o	output tablesss	

The input, output and CLUT tables are arrays of 8 bit unsigned values. Each input table consists of 256 one byte integers. Each input table entry is appropriately normalized to the range 0-255. The inputTable is of size InputChannels * 256 bytes. When stored in this tag, the one-dimensional lookup tables are assumed to be packed one after another in the order described below.

The matrix is organized as a 3 by 3 array. The dimension corresponding to the matrix rows varies least rapidly and the dimension corresponding to the matrix columns varies most rapidly and is shown in matrix form below.

$$\begin{vmatrix} e00 & e01 & e02 \\ e10 & e11 & e12 \\ e20 & e21 & e22 \end{vmatrix}$$

When using the matrix of an output profile, and the input data is XYZ, we have

$$\begin{vmatrix} X' \\ Y' \\ Z' \end{vmatrix} = \begin{vmatrix} XX & XY & XZ \\ YX & YY & YZ \\ ZX & ZY & ZZ \end{vmatrix} \begin{vmatrix} X \\ Y \\ Z \end{vmatrix}$$

Each input X, Y or Z is an unsigned 1.15 number and each matrix entry is a signed 15.16 number. Therefore, each multiplication in the matrix multiply is 1.15 * s15.16 = s16.31 and the final sum is also s16.31. From this sum we take bits 31-16 as the unsigned integer result for X’, Y’, or Z’. These are then scaled to the range 0-255 and used as the inputs to the input tables of the multidimensional LUT. This normalization is used since the number of fractional bits in the input data must be maintained by the matrix operation.

The matrix is mandated to be an identity matrix unless the input to the matrix is in the XYZ color space.

Each CLUT is organized as an n-dimensional array with a variable number of grid points in each dimension, where n is the number of input channels(input tables) in the transform. The dimension corresponding to the first input channel varies least rapidly and the dimension corresponding to the last input channel varies most rapidly. Each grid point value is an m-byte array. The first sequential byte of the entry contains the function value for the first output function, the second sequential byte of the entry contains the function value for the second output function, and so on until all the output functions have been supplied. The equation for computing the size of the CLUT is :

$$CLUTsize = LUTDimensions^{InputChannels} * OutputChannels\ bytes$$

Each output table consists of 256 one byte integers. The outputTable is of size OutputChannels * 256 bytes. When stored in this tag, the one-dimensional lookup tables are assumed to be packed one after another in the order described in the following paragraph.

When using this type, it is necessary to assign each color space component to an input and output channel. The following table shows these assignments. The channels are numbered according to the order in which their table occurs. Note that additional color spaces can be added simply by defining the signature, channel assignments, and creating the tables.

Color Space	Channel 1	Channel 2	Channel 3	Channel 4
'XYZ'	X	Y	Z	
'Lab'	L	a	b	
'Luv'	L	u	v	
'Yxy'	Y	x	y	
'YCbr'	Y	Cb	Cr	
'RGB'	R	G	B	
'GRAY'	K			
'HSV'	H	S	V	
'HLS'	H	L	S	
'CMYK'	C	M	Y	K
'CMY'	C	M	Y	

6.6 measurementType

The measurementType information refers only to the internal profile data and is meant to provide profile makers an alternative to the default measurement specifications.

byte(s)	content	Encoded As...
0-3	'meas' (0x6D656173) type descriptor	
4-7	reserved, must be set to 0	
8-11	encoded value for standard observer	see below
12-23	encoded XYZ tristimulus values for measurement backing	XYZNumber
24-27	encoded value for measurement geometry	see below
28-31	encoded value for measurement flare	see below
32-35	encoded value for standard illuminant	see below

The encoding for the standard observer field is such that:

Standard Observer	Encoded Value
unknown	0x00000000
1931 2 degree Observer	0x00000001
1964 10 degree Observer	0x00000002

The encoding for the measurement geometry field is such that:

Geometry	Encoded Value

unknown	0x00000000
0/45 or 45/0	0x00000001
0/d or d/0	0x00000002

The encoding for the measurement flare value is such that:

Tristimulus Value	Encoded Value
0 (0 %)	0x00000000
1.0 (or 100 %)	0x00010000

The encoding for the standard illuminant field is such that:

Standard Illuminant	Encoded Value
unknown	0x00000000
D50	0x00000001
D65	0x00000002
D93	0x00000003
F2	0x00000004
D55	0x00000005
A	0x00000006
Equi-Power (E)	0x00000007
F8	0x00000008

6.7 namedColorType

This namedColorType is a count value and array of structures that provide color coordinates for 7 bit ASCII color names. This provides users the ability to create a logo color dictionary between a named color set and a space color specification. The color space is identified by the "color space of data" field of the profile header. In order to maintain maximum portability it is strongly recommended that special characters of the 7 bit ASCII set not be used.

byte(s)	content	Encoded As...
0-3	’ncol’ (0x6E636F6C) type descriptor	
4-7	reserved, must be set to 0	
8-11	vender specific flag (lower 16 bits reserved for Consortium use)	
12-15	count of named colors	uInt32Number
15-t	prefix for each color name (maximum of 32 bytes)	7 bit ASCII, 0 term
t+1-u	suffix for each color name (maximum of 32 bytes)	7 bit ASCII, 0 term
u+1-v	first color root name (maximum of 32 bytes)	7 bit ASCII, 0 term
v+1-w	first name’s color coordinates. Color space of data	
w+1-x	second color root name (maximum of 32 bytes)	7 bit ASCII, 0 term
x+1-y	second name’s color coordinates. Color space of data	
y+1-z	the remaining count-2 name structures as described in the first two name structures (assuming count > 2)	

6.8 profileSequenceDescType

This type is an array of structures, each of which contains information from the header fields and tags from the original profiles which were combined to create the final profile. The order of the structures is the order in which the profiles were combined and includes a structure for the final profile. This provides a description of the profile sequence from source to destination, typically used with the devicelink profile.

```
byte(s) content

0-3     ’pseq’  (0x70736571) type descriptor
4-7     reserved, must be set to 0
8-11    count value specifying number of description structures in the array
12-m    ’count’ profile description structures
```

Each profile description structure has the format:

```
byte(s) content

0-3     Device manufacturer signature (from corresponding profile’s header)
4-7     Device model signature (from corresponding profile’s header
8-15    Device attributes (from corresponding profile’s header)
16-19   Device technology information such as CRT, Dye Sublimation, etc
(corresponding profile’s technologyTag)
20-m    displayable description of device manufacturer (corresponding
        profile’s deviceMfgDescTag)
m+1-n   displayable description of device model (corresponding profile’s deviceModel
```

If the deviceMfgDescTag and/or deviceModelDescTag is not present in a component profile, then a "placeholder" tag should be inserted. This tag should have 1 in the ASCII count field and a terminating null in the ASCII invariant profile description and zeros in its UniCode and ScriptCode count and code fields.

Also note that the entire tag, include the tag type, should be stored.

6.9 textDescriptionType

The textDescriptionType is a complex structure that contains three types of text description structures: 7 bit ASCII, Unicode and ScriptCode. Since no single standard method for specifying localizable character sets exists across the major platform vendors, including all three provides access for the major operating systems. The 7 bit ASCII description is to be an invariant, nonlocalizable name for consistent reference. It is preferred that both the Unicode and ScriptCode structures be properly localized.

The localized MacIntosh profile description contains 67 bytes of data, of which at most ’count’ bytes contain a ScriptCode string, including nullterminator. The ’count’ cannot be greater than 67.

The count field for each types are defined as follows:

ASCII: The count is the length of the string in bytes including the terminating null.

Unicode: The count is the number of characters including a Unicode null where a character is always two bytes.

ScriptCode: The count is the length of the string in bytes including the terminating null.

If both Unicode and ScriptCode structures cannot be localized, then the follow guidelines should be used. If Unicode is not native on the platform, then the Unicode should be filled in as 0 and ASCII data inserted in the text field. If the ScriptCode is not native on the platform, then the ScriptCode should be filled in as 0 and the ASCII data inserted in the text field.

byte(s)	content
0-3	'desc' (0x64657363) type descriptor
4-7	reserved, must be set to 0
8-11	7 bit ASCII invariant Profile description count, including terminating null (description length)
12-n-1	7 bit ASCII invariant Profile description
n-n+3	Unicode language code
n+4-n+7	Unicode localizable Profile description count (description length)
n+8-m-1	Unicode localizable Profile description
m-m+1	ScriptCode code
m+2	Localizable Macintosh Profile description count (description length)
m+3-m+69	Localizable Macintosh Profile description

6.10 s15Fixed16ArrayType

This type represents an array of generic 4 byte/32 bit fixed point quantity. The number of values is determined from the size of the tag.

byte(s)	contents
0-3	'sf32' (0x73663332) type descriptor
4-7	reserved, must be set to 0
8-n	an array of s15Fixed16Number values

6.11 screeningType

The screeningType describes various screening parameters including screen frequency, screening angle, and spot shape.

byte(s)	content	Encoded As...
0-3	'scrn' (0x7363726E) type descriptor	
4-7	reserved, must be set to 0	
8-11	screening flag	
12-15	number of channels	
16-19	channel #1 frequency	s15Fixed16Number

```
20-23   channel #1 screen angle                           s15Fixed16Number
24-27   channel #1 spot shape                             see below
28-n    frequency, screen angle and spot shape for
        additional channels
```

Flag encoding is such that:

```
Attribute                                         bit position

Use Printer Default Screens (true is 1)            0
Lines/Inch (on is 1) or Lines/cm (off is 0)        1
```

Spot function encoding is such that:

```
Spot Function Value     Encoded Value

unknown                 0
printer default         1
round                   2
diamond                 3
ellipse                 4
line                    5
square                  6
cross                   7
```

6.12 signatureType

The signatureType contains a four byte sequence used for signatures. Typically this type is used for tags that need to be registered and can be displayed on many development systems as a sequence of four characters. Sequences of less than four characters are padded at the end with spaces.

```
byte(s) content

0-3     ’sig ’ (0x73696720) type descriptor
4-7     reserved, must be set to 0
8-11    four byte signature
```

6.13 textType

The textType is a simple text structure that contains a 7 bit ASCII text string. The length of the string can easily be obtained from the element sizeportion of the tag itself. This string must be terminated with a 0x00 byte.

```
byte(s) content

0-3     ’text’ (0x74657874) type descriptor
4-7     reserved, must be set to 0
8-n     a string of count ASCII characters (where count is derived from the
        element size portion of the tag itself)
```

6.14 u16Fixed16ArrayType

This type represents an array of generic 4 byte/32 bit quantity. The number of values is determined from the size of the tag.

```
byte(s) content

0-3     'uf32' (0x75663332) type descriptor
4-7     reserved, must be set to 0
8-n     an array of u16Fixed16Number values
```

6.15 ucrbgType

This type contains curves representing the under color removal and black generation and a text string which is a general description of the method used for the ucr/bg.

```
byte(s) content

0-3     'bfd '  (0x62666420) type descriptor
4-7     reserved, must be set to 0
8-11    count value specifying number of entries in the ucr curve
12-m    actual ucr curve values starting with the zeroth entry and ending with
        the entry count-1.  Each value is a uInt16Number.  If the count is 1, the
        value is a percent.
m+1 - n count value specifying number of entries in the bg curve
n+1 - o actual bg curve values starting with the zeroth entry and ending with
        the entry count-1.  Each value is a uInt16Number.  If the count is 1, the
        value is a percent.
o+1 - p a string of ASCII characters, with a null terminator.
```

6.16 uInt16ArrayType

This type represents an array of generic 2 byte/16 bit quantity. The number of values is determined from the size of the tag.

```
byte(s) content

0-3     'ui16' (0x75693136) type descriptor
4-7     reserved, must be set to 0
8-n     an array of unsigned 16 bit integers
```

6.17 uInt32ArrayType

This type represents an array of generic 4 byte/32 bit quantity. The number of values is determined from the size of the tag.

```
byte(s) content

0-3     'ui32' (0x75693332) type descriptor
4-7     reserved, must be set to 0
8-n     an array of unsigned 32 bit integers
```

6.18 uInt64ArrayType

This type represents an array of generic 8 byte/64 bit quantity. The number of values is determined from the size of the tag.

```
byte(s) content

0-3     'ui64' (0x75693634) type descriptor
4-7     reserved, must be set to 0
8-n     an array of unsigned 64 bit integers
```

6.19 uInt8ArrayType

```
This type represents an array of generic 1 byte/8 bit quantity. The number of
values is determined from the size of the tag.

byte(s) content

0-3     'ui08' (0x75693038) type descriptor
4-7     reserved, must be set to 0
8-n     an array of unsigned 8 bit integers
```

6.20 viewingConditionsType

This type represents a set of viewing condition parameters including: absolute illuminant white point tristimulus values and absolute surround tristimulus values.

```
byte(s) content                                         Encoded As...

0-3     'view' (0x76696577) type descriptor
4-7     reserved, must be set to 0
8-19    absolute XYZ value for illuminant in cd/m2      XYZNumber
20-31   absolute XYZ value for surround in cd/m2        XYZNumber
32-35   illuminant type as described in measurementType
```

6.21 XYZType

The XYZType contains an array of three encoded values for the XYZ tristimulus values. The number of sets of values is determined from the size of the tag. The byte stream is given below. Tristimulus values must be non-negative, the signed encoding allows for implementation optimizations by minimizing the number of fixed formats.

```
byte(s) content                            Encoded As...

0-3     'XYZ ' (0x58595A20)                type descriptor
4-7     reserved, must be set to 0
8-n     an array of XYZ numbers            XYZNumber
```

7 Basic Numeric Types

7.1 dateTimeNumber

This dateTimeNumber is a 12 byte value representation of the time and date. The actual values are encoded as 16 bit unsigned integers.

```
byte(s) content

0-1     number of the year                         ( actual year, i.e. 1994 )
2-3     number of the month                        ( 1-12 )
4-5     number of the day of the month             ( 1-31 )
6-7     number of hours                            ( 0-23 )
8-9     number of minutes                          ( 0-59 )
10-11   number of seconds                          ( 0-59 )
```

7.2 s15Fixed16Number

This type represents a fixed signed 4 byte/32 bit quantity which has 16 fractional bits.

The encoding is such that : s15.16

```
Tristimulus Value         Encoded Value

-32768.0                  0x80000000
0                         0x00000000
1.0                       0x00010000
32767 + (65535/65536) 0x7fffffff
```

7.3 u16Fixed16Number

This type represents a fixed unsigned 4 byte/32 bit quantity which has 16 fractional bits.

```
++++++
```

The encoding is such that : u16.16

Tristimulus Value	Encoded Value
0	0x00000000
1.0	0x00010000
65535 + (65535/65536)	0xffffffff

7.4 uInt16Number

This type represents a generic unsigned 2 byte/16 bit quantity.

7.5 uInt32Number

This type represents a generic unsigned 4 byte/32 bit quantity.

7.6 uInt64Number

This type represents a generic unsigned 8 byte/64 bit quantity.

7.7 uInt8Number

This type represents a generic unsigned 1 byte/8 bit quantity.

7.8 XYZNumber

This type represents a set of three fixed signed 4 byte/32 bit quantity. The byte stream is given below. Tristimulus values must be non-negative. The signed encoding allows for implementation optimizations by minimizing thenumber of fixed formats.

byte(s)	content	Encoded As...
0-3	encoded X value	s15Fixed16Number
4-7	encoded Y value	s15Fixed16Number
8-11	encoded Z value	s15Fixed16Number

8 Tag Sequencing Requirements

The header is the first element in the file structure encompassing the first 128 bytes. This is immediately followed by the tag table. Tag data elements make up the rest of the file structures. There may be any number of tags and no particular order is required for the data of the tags . Each tag may have any size (up to the limit imposed by the 32 bit offsets). Exactly which tags are required or optional with which profiles have been described in section 3 on Device Profiles.

All tag data is required to start on a 4-byte boundary (relative to the start of the profile header) so that a tag starting with a long will be properly aligned without the tag handler needing to know the contents of the tag. This means that the low 2 bits of the beginning offset must be 0. The element size should be for actual data and must not include padding at the endof the tag data.

Device Profile and Color Transforms Tag Sequence :

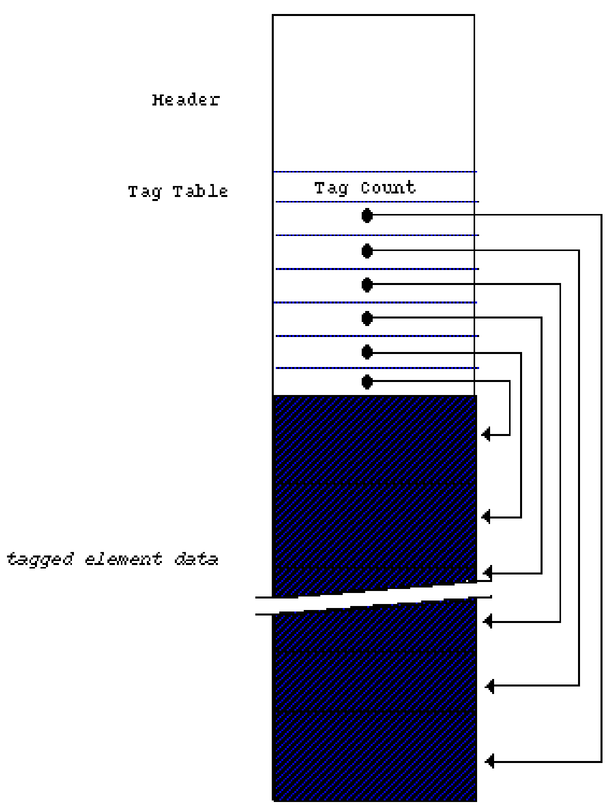
header (as described below)
tag table
(all other tag data elements)

The tag table acts as a table of contents for the tags and tag element data in the profiles. The first four bytes contain a count of the number of tags in the table itself. The tags within the table are not required to be in any particular order.

Individual Tag Structures Within Tag Table

byte(s) content

```
0-3      tag signature
4-7      offset to beginning of tag data
8-11     element size for the number of bytes in the tag data element
```



8.1 Header Description

This header provides a set of parameters at the beginning of the profile format. For color transformation profiles, the device profile dependent fields are set to zero if irrelevant. Having a fixed length header allows for performance enhancements in the profile searching and sorting operations.

byte(s)	content	Encoded As...
0-3	Profile size	
4-7	Identifies the preferred CMM to be used.	
8-11	Profile version number	see below
12-15	Profile/Device class	
16-19	Color space of data (possibly a derived space) [i.e. "the canonical input space"]	see below
20-23	Profile connection space [i.e. "the canonical output space"]	**see below**
24-35	Date and time this profile was first created	dateTimeNumber
36-39	'acsp' (0x61637370) profile file signature	
40-43	Primary platform target for the profile	
44-47	Flags to indicate various options for the CMM such as distributed processing and caching options	see below
48-51	Device manufacturer of the device for which this profile is created	**see below**
52-55	Device model of the device for which this profile is created	**see below**
56-63	Device attributes unique to the particular device setup such as media type	see below
64-67	Specifies the rendering intent of this profile for the CMM. Perceptual, relative colorimetric, saturation and absolute colorimetric are the four intents required to be supported with default values of 0, 1, 2 and 3 respectively.	see below
68-79	The XYZ values of the illuminant of the profile connection space. This must correspond to D50. It is explained in more detail in section 2.	XYZNumber
80-127	48 bytes reserved for future expansion	

CMMType :

Identifies the preferred CMM to be used. The signatures must be registered in order to avoid conflicts.

Profile Version:

Profile version number where the first 8 bits are the major version number and the next 8 bits are for the minor version number. The major and minor version numbers are set by the International Color Consortium and will match up with the profile format revisions. The current version number is 0x02 with a minor version number of 0x00.

The encoding is such that:

Bytes	content
0	Major Revision in BCD
1	Minor Revision & Bug Fix Revision in each nibble in BCD

```
2       reserved, must be set to 0
3       reserved, must be set to 0
```

Major version change can only happen if there is an incompatible change. An example of a major version change may be the addition of new required tags. Minor version change can happen with compatible changes. An example of a minor version number change may be the addition of new optional tags.

Color Space Signatures:

The encoding is such that :

Color Space	Signature	hex encoding
XYZData	'XYZ '	0x58595A20
labData	'Lab '	0x4C616220
luvData	'Luv '	0x4C757620
YCbCrData	'YCbr '	0x59436272
YxyData	'Yxy '	0x59787920
rgbData	'RGB '	0x52474220
grayData	'GRAY'	0x47524159
hsvData	'HSV '	0x48535620
hlsData	'HLS '	0x484C5320
cmykData	'CMYK'	0x434D594B
cmyData	'CMY '	0x434D5920

Profile Connection Space Signatures:

The encoding is such that:

Profile Connection Color Space	Signature	hex encoding
XYZData	'XYZ '	0x58595A20
labData	'Lab '	0x4C616220

When the profile is a DeviceLink profile, the Profile Connection Space Signature is taken from the Color Space Signatures table. (See section 4.1)

Primary Platform Flag:

Flags to indicate the primary platform/operating system framework for which the profile was created.

The encoding is such that:

Primary Platform	Signature	hex encoding
Apple Computer, Inc.	'APPL '	0x4150504C
Microsoft Corporation	'MSFT '	0x4D534654
Silicon Graphics, Inc.	'SGI '	0x53474920
Sun Microsystems, Inc.	'SUNW '	0x53554E57
Taligent, Inc.	'TGNT '	0x54474E54

ProfileFlags:

Flags to indicate various hints for the CMM such as distributed processing and caching options. The first 16 bits (low word in big-endian notation) are reserved for the Profile Consortium.

The encoding is such that:

```
Flags                                                           bit position
Embedded Profile (0 if not embedded, 1 if embedded in file)     0
Profile cannot be used independently from the embedded          1
color data (set to 1 if true, 0 if false)
```

Device manufacturer and model:

The signatures for various manufacturers and models are listed in a separate document (ICC Signatures). New signatures must be registered with the ICC.

Rendering Intent:

Rendering intent of this profile for the CMM. Perceptual, relative colorimetric, saturation and absolute colorimetric are the four intents required to be supported. The first 16 bits worth of numbers are reserved for the Profile Consortium.

The encoding is such that :

```
Rendering Intent             value
Perceptual                   0
Relative Colorimetric        1
Saturation                   2
Absolute Colorimetric        3
```

Note that this flag might not have any meaning until the profile is used in some context, e.g. in a DeviceLink or embedded source profile.

Attributes:

Attributes unique to the particular device setup such as media type. The first 16 bits are reserved for the Profile Consortium.

The encoding is such that (with "on" having value 1 and "off" having value 0)

```
Attribute                                bit position

Reflective (off) or Transparency (on)    0
Glossy (off) or Matte (on)               1s
```

9 Embedding Device Profiles within Documents

This sections details the requirements and options for embedding device profiles within PICT, EPS and TIFF documents. All profiles except abstract profile can be embedded. The complete profile must be embedded with all tags intact and unchanged.

Embedding devicelink profiles renders the color data device dependent and significantly reduces portability. This may be useful in some situations, but may also cause problems with accurate color reproduction.

9.1 PICT

Apple has defined a new QuickDraw picture comment type for embedded ICC profiles. The picture comment value of 224 is followed by a 4-byte selector that describes the type of data in the comment. Using a selector allows the flexibility to embed more ColorSync related information in the future. The following selectors are currently defined:

```
Selector    Description

   0        Begining of an ICC profile.  Profile data to follow.
   1        Continuation of ICC profile data.  Profile data to follow.
   2        End of ICC profile data.  No profile data follows.
```

Because the dataSize parameter of the PicComment procedure is a signed 16-bit value, the maximum amount of profile data that can be embedded in a single picture comment is 32763 bytes (32767 - 4 bytes for the selector). You can embed a larger profile by using multiple picture comments of selector type 1. The profile data must be embedded in consecutive order, and the last piece of profile data must be followed by a picture comment of selector type 2.

All embedded ICC profiles, including those that fit within a single picture comment, must be followed by the end-of-profile picture comment (selector 2), as shown in the following examples.

Example 1: Embedding a 20K profile.

```
PicComment kind = 224, dataSize = 20K + 4, selector = 0, profile data = 20K
PicComment kind = 224, dataSize = 4, selector = 2
```

Example 2: Embedding a 50K profile.

```
PicComment kind = 224, dataSize = 32K, selector = 0, profile data = 32K - 4

PicComment kind = 224, dataSize = 18K + 8, selector = 1, profile data = 18K + 4
PicComment kind = 224, dataSize = 4, selector = 2
```

In ColorSync 1.0, picture comment types `CMBeginProfile` (220) and `CMEndProfile` (221) are used to begin and end a picture comment. The `CMBeginProfile` comment is not supported for ICC profiles; however, the `CMEndProfile` comment can be used to end the current profile and begin using the System Profile for both ColorSync 1.0 and 2.0.

The `CMEnableMatching` (222) and `CMDisableMatching` (223) picture comments are used to begin and end color matching in both ColorSync 1.0 and 2.0.

See *Inside Macintosh: Imaging With QuickDraw* for more information about picture comments.

9.2 EPS

There are two places within EPS files that embedding International ColorConsortium (ICC) profiles are appropriate. 1) Associated with a screenpreview. 2) Associated with the page description. Embedding ICC profileswithin a screen preview is necessary so that applications using this screen preview to display a representation of the EPS page description can do so with accurate colors. Embedding ICC profiles within a page description is necessary so that sophisticated applications, such as OPI server software, can perform color conversions along with image replacement. For general information concerning PostScript's Document Structuring Conventions (DSC), the EPS file format, or specific PostScript operators, see the PostScript Language Reference Manual, second edition.

1) There are a variety of different methods of storing a screen preview within an EPS file depending on the intended environment. For cross platform applications with embedded ICC profiles, TIFF screen previews are recommended. The TIFF format has been extended to support the embedding of ICC profiles. ICC profiles can also be embedded in a platform specific manner. For example on the Macintosh, Apple has defined a method for embedding ICC profiles in PICT files.

Note that a given page description may use multiple distinct color spaces. In such cases, color conversions must be performed to a single color space to associate with the screen preview.

2) ICC profiles can also be embedded in the page description portion of an EPS file using the %%BeginICCProfile / %%EndICCProfile comments. This convention is defined as follows.

```
%%BeginICCProfile:  <profileid> <numberof> [<type> [<bytesorlines>]]
  <profileid> ::= <text>            (Profile ID)
  <numberof> ::= <int>             (Lines or physical bytes)
  <type> ::= Hex | ASCII            (Type of data)
  <bytesorlines> ::= Bytes | Lines  (Read in bytes or lines)
%%EndICCProfile (no keywords)
```

These comments are designed to provide information about embedded ICC profiles. If the type argument is missing, ASCII data is assumed. ASCII refers to an ASCII base-85 representation of the data. If the bytesorlines argument is missing, <numberof> shall be considered to indicate bytes of data. If <numberof> = -1, the number of bytes of data are unknown. In this case, to skip over the profile one must read data until the encountering the %%EndICCProfile comment.

<profileID> provides the profile's ID in order to synchronize it with PostScript's setcolorspace and findcolorrendering operators and associated operands (see below). Note that <numberof> indicates the bytes of physical data, which vary from the bytes of virtual data in some cases. With hex, each byte of virtual data is represented by two ASCII characters (two bytes of physical data). Although the PostScript interpreter ignores white space and percent signs in hex and ASCII data, these count toward the byte count.

Each line of profile data shall begin with a single percent sign followed by a space (%). This makes the entire profile section a PostScript language comment so the file can be sent directly to a printer without modification. The space avoids confusion with the open extension mechanism associated with DSC comments.

ICC profiles can be embedded within EPS files to allow sophisticated applications, such as OPI server software, to extract the profiles, and to perform color processing based on these profiles. In such situations it is desirable to locate the page description's color space and rendering intent, since this color space and rendering intent may need to be modified based on any color processing. The `%%BeginSetColorSpace/%%EndSetColorSpace` and `%%BeginRenderingIntent` / `%%EndRenderingIntent` comments are used to delimit the color space and rendering intent respectively.

```
%%BeginSetColorSpace <profileid>
  <profileid> ::= <text>            (ICC Profile ID)
%%EndSetColorSpace (no keywords)
```

`<profileid>` provides the ICC profile's ID corresponding to this color space. The ICC profile with this profile must have occured in the PostScript job using the `%%BeginICCProfile` / `%%EndICCProfile` comment convention prior to this particular `%%BeginSetColorSpace` comment.

An example usage is shown here for CIE 1931 (XYZ)-space with D65 white point that refers to the ICC profile with `<profileid>` = XYZProfile.

```
%%BeingSetColorSpace XYZProfile
[/CIEBasedABC <<
  /WhitePoint [0.9505 1 1.0890]
  /RangeABC [0 0.9505 0 1 0 1.0890]
  /RangeLMN [0 0.9505 0 1 0 1.0890]
>>] setcolorspace
%%EndSetColorSpace
```

Note that the setcolorspace command is included within the comments. The PostScript enclosed in these comments shall not perform any other operations other than setting the color space and shall have no side effects.

```
%%BeginRenderingIntent <profileid>
  <profileid> ::= <text>            (ICC Profile ID)
%%EndRenderingIntent
```

`<profileid>` provides the ICC profile's ID corresponding to this rendering intent. The ICC profile with this profile must have occured in the PostScript job using the `%%BeginICCProfile /` `%%EndICCProfile` comment convention prior to invocation of this particular `%%BeginRenderingIntent` comment.

An example usage is shown here for the the Perceptual rendering intent that refers to the ICC profile with `<profileid>` = RGBProfile.

```
%%BeginRenderingIntent RGBProfile
/Perceptual findcolorrendering pop
/ColorRendering findresource setcolorrendering
%%EndRenderingIntent
```

Note that the setcolorrendering command is included within the comments. The PostScript enclosed in these comments shall not perform any other operations other than setting the rendering intent and shall have no side effects.

9.3 TIFF

The discussion below assumes some familiarity with TIFF internal structure. It is beyond the scope of this document to detail the TIFF format, and readers are referred to the "TIFF(tm) Revision 6.0" specification, which is available from the Aldus Corporation.

The International Color Consortium (ICC) has been assigned a private TIFF tag for purposes of embedding ICC device profiles within TIFF image files. This is not a required TIFF tag, and Baseline TIFF readers are not currently required to read it. It is, however, strongly recommended that this tag be honored.

A ICCdevice profile is embedded, in its entirety, as a single TIFF field or Image File Directory (IFD) entry in the IFD containing the corresponding image data. An IFD should contain no more than one embedded profile. A TIFF file may contain more than one image, and so, more than one IFD. Each IFD may have its own embedded profile. Note, however, that Baseline TIFF readers are not required to read any IFDs beyond the first one.

The structure of the ICC Profile IFD Entry is as follows.

```
Bytes 0-1               The TIFF  Tag that identifies the field = 34675(8773.H)
Bytes 2-3               The field Type = 7 = UNDEFINED (treated as 8-bit bytes).
Bytes 4-7               The Count of values = the size of the embedded ICC profile i
Bytes 8-11              The Value Offset = the file offset, in bytes, to the beginni
                        the ICC profile.
```

Like all IFD entry values, the embedded profile must begin on a word boundary, so the Value Offset will always be an even number. A TIFF reader should have no knowledge of the internal structure of an embedded ICC profile and should extract the profile intact.

Appendix A : C Header File Example

This appendix provides a cross-platform conditionally compilable header file for the InterColor Profile Format.

```
/* Header file guard bands */
#ifndef INTERCOLOR_H
#define INTERCOLOR_H

#pragma ident "@(#)intercolor.h 1.1  10 Jun 1994"
/*
 * Copyright 1994 The InterColor Profile Consortium & SunSoft Inc
 *
 * Permission to use, copy, modify, and distribute this software and its
```

```
 * documentation for any purpose and without fee is hereby granted,
 * provided that the above copyright notice appear in all copies and that
 * both that copyright notice and this permission notice appear in
 * supporting documentation, and that the names of the ColorSync Profile
 * Consortium and SunSoft not be used in advertising or publicity pertaining
 *  to distribution of the software without specific, written prior
 * permission.
 *
 * SUNSOFT DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING
 * ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL
 * SUNSOFT BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR
 * ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
 * WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION,
 * ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS
 * SOFTWARE.
 */

/*
 * This version of the header file corresponds to the profile
 * specification version 3.0.
 *
 * All header file entries are pre-fixed with "ic" to help
 * avoid name space collisions. Signatures are pre-fixed with
 * icSig.
 */

/*------------------------------------------------------------------------*/
/*
 * Defines used in the specification
 */
#define icMagicNumber                   0x61637370      /* 'acsp' */
#define icVersionNumber                 0x02000000      /* 2.0, BCD */

/* Screening Encodings */
#define icPrtrDefaultScreensFalse       0x00000000      /* Bit position 0 */
#define icPrtrDefaultScreensTrue        0x00000001      /* Bit position 0 */
#define icLinesPerInch                  0x00000002      /* Bit position 1 */
#define icLinesPerCm                    0x00000000      /* Bit position 1 */

/*
 * Device attributes, currently defined values correspond
 * to the low 4 bytes of the 8 byte attribute quantity, see
 * the header for their location.
 */
#define icReflective                    0x00000000      /* Bit position 0 */
#define icTransparency                  0x00000001      /* Bit position 0 */
#define icGlossy                                0x00000000      /* Bit
position 1 */
#define icMatte                         0x00000002      /* Bit position 1 */

/*
 * Profile header flags, the low 16 bits are reserved for consortium
 * use.
 */
#define icEmbeddedProfileFalse  0x00000000      /* Bit position 0 */
#define icEmbeddedProfileTrue           0x00000001      /* Bit position 0 */
#define icUseAnywhere                   0x00000000      /* Bit position 1 */
#define icUseWithEmdeddedDataOnly       0x00000002      /* Bit position 1 */

/* Ascii or Binary data */
#define icAsciiData                     0x00000000      /* Used in dataType */
```

```
#define icBinaryData                    0x00000001

/*
 * Define used to indicate that this is a variable length array
 */
#define icAny                           1


/*------------------------------------------------------------------------*/
/*
 * Signatures, these are basically 4 byte identifiers
 * used to differentiate between tags and other items
 * in the profile format.
 *
 * Set the typedef icSignature as appropriate for your
 * Operating system.
 */
#if defined(unix) || defined(__unix)
typedef long    icSignature;
#endif Unix

/* public tags and sizes */
typedef enum {
    icSigAToB0Tag                               = 0x41324230,   /* 'A2B0' */
    icSigAToB1Tag                               = 0x41324231,   /* 'A2B1' */
    icSigAToB2Tag                               = 0x41324232,   /* 'A2B2' */
    icSigBlueColorantTag                = 0x6258595A,   /* 'bXYZ' */
    icSigBlueTRCTag                     = 0x62545243,   /* 'bTRC' */
    icSigBToA0Tag                               = 0x42324130,   /* 'B2A0' */
    icSigBToA1Tag                               = 0x42324131,   /* 'B2A1' */
    icSigBToA2Tag                               = 0x42324132,   /* 'B2A2' */
    icSigCalibrationDateTimeTag = 0x63616C74,   /* 'calt' */
    icSigCharTargetTag                  = 0x74617267,   /* 'targ' */
    icSigCopyrightTag                   = 0x63707274,   /* 'cprt' */
    icSigDeviceMfgDescTag               = 0x646D6E64,   /* 'dmnd' */
    icSigDeviceModelDescTag             = 0x646D6464,   /* 'dmdd' */
    icSigGamutTag                               = 0x676d7420,   /* 'gmt ' */
    icSigGrayTRCTag                     = 0x6b545243,   /* 'kTRC' */
    icSigGreenColorantTag               = 0x6758595A,   /* 'gXYZ' */
    icSigGreenTRCTag                    = 0x67545243,   /* 'gTRC' */
    icSigLuminanceTag                   = 0x6C756d69,   /* 'lumi' */
    icSigMeasurementTag                 = 0x6D656173,   /* 'meas' */
    icSigMediaBlackPointTag             = 0x626B7074,   /* 'bkpt' */
    icSigMediaWhitePointTag             = 0x77747074,   /* 'wtpt' */
    icSigNamedColorTag                  = 0x6E636f6C,   /* 'ncol' */
    icSigPreview0Tag                    = 0x70726530,   /* 'pre0' */
    icSigPreview1Tag                    = 0x70726531,   /* 'pre1' */
    icSigPreview2Tag                    = 0x70726532,   /* 'pre2' */
    icSigProfileDescriptionTag  = 0x64657363,   /* 'desc' */
    icSigProfileSequenceDescTag = 0x70736571,   /* 'pseq' */
    icSigPs2CRD0Tag                     = 0x70736430,   /* 'psd0' */
    icSigPs2CRD1Tag                     = 0x70736431,   /* 'psd1' */
    icSigPs2CRD2Tag                     = 0x70736432,   /* 'psd2' */
    icSigPs2CRD3Tag                     = 0x70736433,   /* 'psd3' */
    icSigPs2CSATag                      = 0x70733273,   /* 'ps2s' */
    icSigPs2RenderingIntentTag  = 0x70733269,   /* 'ps2i' */
    icSigRedColorantTag                 = 0x7258595A,   /* 'rXYZ' */
    icSigRedTRCTag                      = 0x72545243,   /* 'rTRC' */
    icSigScreeningDescTag               = 0x73637264,   /* 'scrd' */
    icSigScreeningTag                   = 0x7363726E,   /* 'scrn' */
    icSigTechnologyTag                  = 0x74656368,   /* 'tech' */
```

```
    icSigUcrBgTag                               = 0x62666420,   /* 'bfd ' */
    icSigViewingCondDescTag             = 0x76756564,   /* 'vued' */
    icSigViewingConditionsTag           = 0x76696577,   /* 'view' */
    icMaxEnumTag                                = 0xFFFFFFFF    /* enum = 4
bytes max */
} icTagSignature;

/* technology signature descriptions */
typedef enum {
    icSigFilmScanner                    = 0x6673636E,   /* 'fscn' */
    icSigReflectiveScanner              = 0x7273636E,   /* 'rscn' */
    icSigInkJetPrinter                  = 0x696A6574,   /* 'ijet' */
    icSigThermalWaxPrinter              = 0x74776178,   /* 'twax' */
    icSigElectrophotographicPrinter     = 0x6570686F,   /* 'epho' */
    icSigElectrostaticPrinter           = 0x65737461,   /* 'esta' */
    icSigDyeSublimationPrinter  = 0x64737562,   /* 'dsub' */
    icSigPhotographicPaperPrinter       = 0x7270686F,   /* 'rpho' */
    icSigFilmWriter                     = 0x6670726E,   /* 'fprn' */
    icSigVideoMonitor                   = 0x7669646D,   /* 'vidm' */
    icSigVideoCamera                    = 0x76696463,   /* 'vidc' */
    icSigProjectionTelevision           = 0x706A7476,   /* 'pjtv' */
    icSigCRTDisplay                     = 0x43525420,   /* 'CRT ' */
    icSigPMDisplay                      = 0x504D4420,   /* 'PMD ' */
    icSigAMDisplay                      = 0x414D4420,   /* 'AMD ' */
    icSigPhotoCD                                = 0x4B504344,   /* 'KPCD' */
    icSigPhotoImageSetter               = 0x696D6773,   /* 'imgs' */
    icSigGravure                                = 0x67726176,   /* 'grav' */
    icSigOffsetLithography              = 0x6F666673,   /* 'offs' */
    icSigSilkscreen                     = 0x73696C6B,   /* 'silk' */
    icSigFlexography                    = 0x666C6578,   /* 'flex' */
    icMaxEnumTechnology                 = 0xFFFFFFFF    /* enum = 4 bytes max
*/
} icTechnologySignature;

/* type signatures */
typedef enum {
    icSigCurveType                      = 0x63757276,   /* 'curv' */
    icSigDataType                               = 0x64617461,   /* 'data' */
    icSigDateTimeType                   = 0x6474696D,   /* 'dtim' */
    icSigLut16Type                      = 0x6d667432,   /* 'mft2' */
    icSigLut8Type                               = 0x6d667431,   /* 'mft1' */
    icSigNamedColorType                 = 0x6E636f6C,   /* 'ncol' */
    icSigProfileDescriptionType = 0x64657363,   /* 'desc' */
    icSigS15Fixed16Type                 = 0x73663332,   /* 'sf32' */
    icSigScreeningType                  = 0x7363726E,   /* 'scrn' */
    icSigSignatureType                  = 0x73696720,   /* 'sig ' */
    icSigTextType                               = 0x74657874,   /* 'text' */
    icSigU16Fixed16Type                 = 0x75663332,   /* 'uf32' */
    icSigUInt16Type                     = 0x75693136,   /* 'ui16' */
    icSigUInt32Type                     = 0x75693332,   /* 'ui32' */
    icSigUInt64Type                     = 0x75693634,   /* 'ui64' */
    icSigUInt8Type                      = 0x75693038,   /* 'ui08' */
    icSigViewingConditionsType  = 0x76696577,   /* 'view' */
    icSigXYZType                                = 0x58595A20,   /* 'XYZ ' */
    icMaxEnumType                               = 0xFFFFFFFF    /* enum = 4
bytes max */
} icTagTypeSignature;

/* Color Space Signatures */
typedef enum {
    icSigXYZData                                = 0x58595A20,   /* 'XYZ ' */
```

```
    icSigLabData                                = 0x4C616220,   /* 'Lab ' */
    icSigLuvData                                = 0x4C757620,   /* 'Luv ' */
    icSigYCbCrData                      = 0x59436272,   /* 'YCbr' */
    icSigYxyData                                = 0x59787920,   /* 'Yxy ' */
    icSigRgbData                                = 0x52474220,   /* 'RGB ' */
    icSigGrayData                               = 0x47524159,   /* 'GRAY' */
    icSigHsvData                                = 0x48535620,   /* 'HSV ' */
    icSigHlsData                                = 0x484C5320,   /* 'HLS ' */
    icSigCmykData                               = 0x434D594B,   /* 'CMYK' */
    icSigCmyData                                = 0x434D5920,   /* 'CMY ' */
    icMaxEnumData                               = 0xFFFFFFFF    /* enum = 4
bytes max */
} icColorSpaceSignature;

/* profileClass enumerations */
typedef enum {
    icSigInputClass                     = 0x73636E72,   /* 'scnr' */
    icSigDisplayClass                   = 0x6D6E7472,   /* 'mntr' */
    icSigOutputClass                    = 0x70727472,   /* 'prtr' */
    icSigLinkClass                      = 0x6C696E6B,   /* 'link' */
    icSigAbstractClass                  = 0x61627374,   /* 'abst' */
    icSigColorSpaceClass                = 0x73706163,   /* 'spac' */
    icMaxEnumClass                      = 0xFFFFFFFF    /* enum = 4 bytes max
*/
} icProfileClassSignature;

/* Platform Signatures */
typedef enum {
    icSigMacintosh                      = 0x4141504C,   /* 'AAPL' */
    icSigMicrosoft                      = 0x4D534654,   /* 'MSFT' */
    icSigSolaris                                = 0x53554E57,   /* 'SUNW' */
    icSigSGI                            = 0x53474920,   /* 'SGI ' */
    icSigTaligent                               = 0x54474E54,   /* 'TGNT' */
    icMaxEnumPlatform                   = 0xFFFFFFFF    /* enum = 4 bytes max
*/
} icPlatformSignature;

/*------------------------------------------------------------------------*/
/*
 * Other enums
 */

/* Measurement Flare, used in the measurmentType tag */
typedef enum {
    icFlare0                            = 0x00000000,   /* 0% flare */
    icFlare100                          = 0x00000001,   /* 100% flare */
    icMaxFlare                          = 0xFFFFFFFF    /* enum = 4 bytes max
*/
} icMeasurementFlare;

/* Measurement Geometry, used in the measurmentType tag */
typedef enum {
    icGeometryUnknown                   = 0x00000000,   /* Unknown geometry */
    icGeometry045or450                  = 0x00000001,   /* 0/45 or 45/0 */
    icGeometry0dord0                    = 0x00000002,   /* 0/d or d/0 */
    icMaxGeometry                               = 0xFFFFFFFF    /* enum = 4
bytes max */
} icMeasurementGeometry;

/* Rendering Intents, used in the profile header */
typedef enum {
```

```
    icPerceptual                                = 0,
    icRelativeColorimetric              = 1,
    icSaturation                                = 2,
    icAbsoluteColorimetric              = 3,
    icMaxEnumIntent                     = 0xFFFFFFFF    /* enum = 4 bytes max
*/
} icRenderingIntent;

/* Different Spot Shapes currently defined, used for screeningType */
typedef enum {
    icSpotShapeUnknown                  = 0,
    icSpotShapePrinterDefault           = 1,
    icSpotShapeRound                    = 2,
    icSpotShapeDiamond                  = 3,
    icSpotShapeEllipse                  = 4,
    icSpotShapeLine                     = 5,
    icSpotShapeSquare                   = 6,
    icSpotShapeCross                    = 7,
    icMaxEnumSpot                               = 0xFFFFFFFF    /* enum = 4
bytes max */
} icSpotShape;

/* Standard Observer, used in the measurmentType tag */
typedef enum {
    icStdObsUnknown                     = 0x00000000,   /* Unknown observer */
    icStdObs1931TwoDegrees              = 0x00000001,   /* 1931 two degrees */
    icStdObs1964TenDegrees              = 0x00000002,   /* 1961 ten degrees */
    icMaxStdObs                         = 0xFFFFFFFF    /* enum = 4 bytes max
*/
} icStandardObserver;

/* Pre-defined illuminants, used in measurement and viewing conditions type */
typedef enum {
    icIlluminantUnknown                 = 0x00000000,
    icIlluminantD50                     = 0x00000001,
    icIlluminantD65                     = 0x00000002,
    icIlluminantD93                     = 0x00000003,
    icIlluminantF2                      = 0x00000004,
    icIlluminantD55                     = 0x00000005,
    icIlluminantA                               = 0x00000006,
    icIlluminantEquiPowerE              = 0x00000007,   /* Equi-Power (E) */
    icIlluminantF8                      = 0x00000008,
    icMaxEnumIluminant                  = 0xFFFFFFFF    /* enum = 4 bytes max
*/
} icIlluminant;

/*------------------------------------------------------------------------*/
/*
 * Number definitions
 */

/* Fixed numbers */
typedef long            icS15Fixed16Number;
typedef unsigned long   icU16Fixed16Number;

/* Plain old int numbers */
typedef unsigned char   icUInt8Number;
typedef unsigned short  icUInt16Number;
typedef unsigned long   icUInt32Number;
typedef unsigned long   icUInt64Number[2];
```

```
/* Signed numbers */
typedef char            icInt8Number;
typedef short           icInt16Number;
typedef long            icInt32Number;
typedef long            icInt64Number[2];

/* The base date time number */
typedef struct {
    icUInt16Number      year;
    icUInt16Number      month;
    icUInt16Number      day;
    icUInt16Number      hours;
    icUInt16Number      minutes;
    icUInt16Number      seconds;
} icDateTimeNumber;

/* XYZ Number  */
typedef struct {
    icS15Fixed16Number  X;
    icS15Fixed16Number  Y;
    icS15Fixed16Number  Z;
} icXYZNumber;

/*------------------------------------------------------------------------*/
/*
 * Tag Type definitions
 */

/*
 * Many of the structures contain variable length arrays. This
 * is represented by the use of the convention.
 *
 *      type    data[icAny];
 */
/* curveType */
typedef struct {
    icSignature         sig;                    /* Signature, "curv" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icUInt32Number      count;          /* Number of entries */
    icUInt16Number      data[icAny];    /* The actual table data, real
                                         * number is determined by count
                                                         */
} icCurveType;

/* dataType */
typedef struct {
    icSignature         sig;                    /* Signature, "data" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icUInt32Number      dataFlag;               /* 0 = ascii, 1 = binary */
    icInt8Number                data[icAny];    /* Data, size determined from
tag */
} icDataType;

/* dateTimeType */
typedef struct {
    icSignature         sig;                    /* Signature, "dtim" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icDateTimeNumber    date;                   /* The date */
} icDateTimeType;

/* lut16Type */
```

```
typedef struct {
    icSignature         sig;                    /* Signature, "mft2" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icUInt8Number               inputChan;              /* Number of input
channels */
    icUInt8Number               outputChan;             /* Number of output
channels */
    icUInt8Number               clutPoints;          /* Number of clutTable
grid points */
    icInt8Number                pad;                    /* Padding for byte
alignment */
    icS15Fixed16Number  e00;                    /* e00 in the 3 * 3 */
    icS15Fixed16Number  e01;                    /* e01 in the 3 * 3 */
    icS15Fixed16Number  e02;                    /* e02 in the 3 * 3 */
    icS15Fixed16Number  e10;                    /* e10 in the 3 * 3 */
    icS15Fixed16Number  e11;                    /* e11 in the 3 * 3 */
    icS15Fixed16Number  e12;                    /* e12 in the 3 * 3 */
    icS15Fixed16Number  e20;                    /* e20 in the 3 * 3 */
    icS15Fixed16Number  e21;                    /* e21 in the 3 * 3 */
    icS15Fixed16Number  e22;                    /* e22 in the 3 * 3 */
    icUInt16Number      inputEnt;               /* Number of input table
entries */
    icUInt16Number      outputEnt;              /* Number of output table
entries */
    icUInt16Number      data[icAny];    /* Data follows see spec for size */
/*
 *  Data that follows is of this form
 *
 *  icUInt16Number      inputTable[icAny];      * The input table
 *  icUInt16Number      clutTable[icAny];               * The clut table
 *  icUInt16Number      outputTable[icAny];     * The output table
 */
} icLut16Type;

/* lut8Type, input & output tables are always 256 bytes in length */
typedef struct {
    icSignature         sig;                    /* Signature, "mft1" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icUInt8Number               inputChan;              /* Number of input
channels */
    icUInt8Number               outputChan;             /* Number of output
channels */
    icUInt8Number               clutPoints;          /* Number of clutTable
grid points */
    icInt8Number                pad;
    icS15Fixed16Number  e00;                    /* e00 in the 3 * 3 */
    icS15Fixed16Number  e01;                    /* e01 in the 3 * 3 */
    icS15Fixed16Number  e02;                    /* e02 in the 3 * 3 */
    icS15Fixed16Number  e10;                    /* e10 in the 3 * 3 */
    icS15Fixed16Number  e11;                    /* e11 in the 3 * 3 */
    icS15Fixed16Number  e12;                    /* e12 in the 3 * 3 */
    icS15Fixed16Number  e20;                    /* e20 in the 3 * 3 */
    icS15Fixed16Number  e21;                    /* e21 in the 3 * 3 */
    icS15Fixed16Number  e22;                    /* e22 in the 3 * 3 */
    icUInt8Number       data[icAny];            /* Data follows see spec for
size */
/*
 *  Data that follows is of this form
 *
 *  icUInt8Number       inputTable[256];                * The input table
 *  icUInt8Number       clutTable[icAny];               * The clut table
```

```
 *  icUInt8Number       outputTable[256];               * The output table
 */
} icLut8Type;

/* Measurement Type */
typedef struct {
    icSignature                 sig;            /* Signature, "meas" */
    icInt8Number                        reserved[4];/* Reserved, set to 0 */
    icStandardObserver          stdObserver;/* Standard observer */
    icXYZNumber                 backing;        /* XYZ for backing material */
    icMeasurementGeometry       geometry;       /* Measurement geometry */
    icMeasurementFlare          flare;  /* Measurement flare */
    icIlluminant                        illuminant;     /* Illuminant */
} icMeasurmentType;

/* Named color type */
typedef struct {
    icSignature         sig;                    /* Signature, "ncol" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icUInt32Number      vendorFlag;             /* Bottom 16 bits for IC use
*/
    icUInt32Number      count;          /* Count of named colors */
    icInt8Number                data[icAny];    /* Named color data follows */
/*
 *  Data that follows is of this form
 *
 * icInt8Number         prefix[icAny];  * Prefix for the color name, max = 32
 * icInt8Number         suffix[icAny];  * Suffix for the color name, max = 32
 * icInt8Number         root1[icAny];   * Root name for first color, max = 32
 * icInt8Number         coords1[icAny]; * Color co-ordinates of first color
 * icInt8Number         root2[icAny];   * Root name for first color, max = 32
 * icInt8Number         coords2[icAny]; * Color co-ordinates of first color
 *                      :
 *                      :
 * Repeat for root name and color co-ordinates up to (count-1)
 */
} icNamedColorType;

/* Profile sequence structure */
typedef struct {
    icSignature         deviceMfg;              /* Device Manufacturer */
    icSignature         deviceModel;    /* Decvice Model */
    icUInt64Number      attributes;             /* Device attributes */
    icSignature         technology;             /* Technology signature */
    icInt8Number                data[icAny];    /* Descriptions text follows*/
/*
 *  Data that follows is of this form
 *
 * icInt8Number         mfgDesc[icAny]; * Manufacturer text
 * icInt8Number         modelDesc[icAny];       * Model text
 */
} icDescStruct;

/* Profile sequence description type */
typedef struct {
    icSignature         sig;                    /* Signature, "pseq" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icUInt32Number      count;          /* Number of descriptions */
    icDescStruct                data[icAny];    /* Array of description struct
*/
} icProfileSequenceDescTag;
```

```
/* profileDescriptionType */
typedef struct {
    icSignature         sig;                    /* Signature, "desc" */
    icInt8Number                eserved[4];             /* Reserved, set to 0
*/
    icUInt32Number      count;          /* Description length */
    icInt8Number                data[icAny];    /* Descriptions follow */
/*
 *  Data that follows is of this form
 *
 * icInt8Number         desc[icAny]             * NULL terminated ascii string
 * icUInt32Number               ucLangCode;             * UniCode language
code
 * icUInt32Number               ucCount;                * UniCode description
length
 * icInt8Number         ucDesc[icAny;   * The UniCode description
 * icUInt16Number               scCode;         * ScriptCode code
 * icUInt8Number                scCount;                * ScriptCode count
 * icInt8Number         scDesc[64];             * ScriptCode Description
 */
} icProfileDescriptionType;

/* s15Fixed16Type */
typedef struct {
    icSignature         sig;                    /* Signature, "sf32" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icS15Fixed16Number  data[icAny];    /* Array of values */
} icS15Fixed16ArrayType;

/* screeningType */
typedef struct {
    icS15Fixed16Number  frequency;              /* Frequency */
    icS15Fixed16Number  angle;          /* Screen angle */
    icSpotShape         spotShape;              /* Spot Shape encodings below
*/
} icScreeningData;

typedef struct {
    icSignature         sig;                    /* Signature, "scrn" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icUInt32Number      screeningFlag;  /* Screening flag */
    icUInt32Number      channels;               /* Number of channels */
    icScreeningData     data[icAny];    /* Array of screening data */
} icScreeningType;

/* sigType */
typedef struct {
    icSignature         sig;                    /* Signature, "sig" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icSignature         signature;
} icSignatureType;

/* textType */
typedef struct {
    icSignature         sig;                    /* Signature, "text" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icInt8Number                data[icAny];    /* Variable array of
characters */
} icTextType;

/* u16Fixed16Type */
```

```
typedef struct {
    icSignature         sig;                    /* Signature, "uf32" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icU16Fixed16Number  data[icAny];    /* Variable array of values */
} icU16Fixed16ArrayType;

/* Structure describing either a UCR or BG curve */
typedef struct {
    icUInt32Number      count;          /* Curve length */
    icUInt16Number      curve[icAny];   /* The array of curve values */
} icUcrBgCurve;

/* Under color removal, black generation type */
typedef struct {
    icSignature         sig;                    /* Signature, "bfd " */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icUcrBgCurve                ucr;                    /* Ucr curve */
    icUcrBgCurve                bg;                     /* Bg curve */
} icUcrBgType;

/* uInt16Type */
typedef struct {
    icSignature         sig;                    /* Signature, "ui16" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icUInt16Number      data[icAny];    /* Variable array of values */
} icUInt16ArrayType;

/* uInt32Type */
typedef struct {
    icSignature         sig;                    /* Signature, "ui32" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icUInt32Number      data[icAny];    /* Variable array of values */
} icUInt32ArrayType;

/* uInt64Type */
typedef struct {
    icSignature         sig;                    /* Signature, "ui64" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icUInt64Number      data[icAny];    /* Variable array of values */
} icUInt64ArrayType;

/* uInt8Type */
typedef struct {
    icSignature         sig;                    /* Signature, "ui08" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icUInt8Number               data[icAny];    /* Variable array of values */
} icUInt8ArrayType;

/* viewingConditionsType */
typedef struct {
    icSignature         sig;                    /* Signature, "view" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icXYZNumber         illuminant;             /* In candelas per metre sq’d
*/
    icXYZNumber         surround;               /* In candelas per metre sq’d
*/
    icIlluminant                stdIluminant;   /* See icIlluminant defines */
} icViewingConditionType;

/* XYZ Type */
typedef struct {
```

```
    icSignature         sig;                    /* Signature, "XYZ" */
    icInt8Number                reserved[4];    /* Reserved, set to 0 */
    icXYZNumber         data[icAny];    /* Variable array of XYZ numbers */
} icXYZType;

/*---------------------------------------------------------------------------*/

/*
 * Tag table and profile header
 */

/* A tag */
typedef struct {
    icTagSignature      sig;                    /* The tag signature */
    icUInt32Number      offset;         /* To start of data from header */
    icUInt32Number      size;                   /* Size in bytes */
} icTag;

/* A Tag Table */
typedef struct {
    icUInt32Number      count;          /* Number of tags in the profile */
    icTag                       tags[icAny];    /* Variable array of tags */
} icTagTable;

/* The Profile header */
typedef struct {
    icUInt32Number      size;                   /* Profile size in bytes */
    icSignature         cmmId;       /* CMM associated with the profile */
    icUInt32Number      version;                /* Format version number */
    icSignature         deviceClass;    /* Type of device */
    icSignature         colorSpace;             /* Color space of data */
    icSignature         pcs;                    /* Profile Connection Space */
    icDateTimeNumber    date;                   /* Date profile was created */
    icSignature         magic;          /* The magic number ’acsp’ */
    icSignature         platform;               /* Primary Platform */
    icUInt32Number      flags;          /* Various bit settings */
    icSignature         manufacturer;   /* Device manufacturer */
    icUInt32Number      model;          /* Device model number */
    icUInt64Number      attributes;             /* Device attributes */
    icUInt32Number      renderingIntent;        /* Rendering intent */
    icXYZNumber         illuminant;             /* Profile illuminant */
    icInt8Number                reserved[48];   /* Reserved for future use */
} icHeader;

/* A profile */
typedef struct {
    icHeader            header;         /* The header */
    icTagTable          tagTable;               /* The tag table */
    icInt8Number                tags[icAny];    /* Start of tag data */
} icProfile;

/*---------------------------------------------------------------------------*/

#endif INTERCOLOR_H
```

Appendix B : 7 Bit ASCII

The ASCII character set defines a 1-to-1 mapping of characters to 8-bit values. The characters with hex codes 0x00-0x7F, the 7-bit subset of ISO Latin-1 characters which are invariant regardless of locale.

In a InterColor invariant profile name, it is illegal to use characters not defined in this table and to maintain maximum portability, it is recommended that special characters not be used :

```
hexadecimal:
| 00 nul | 01 soh | 02 stx | 03 etx | 04 eot | 05 enq | 06 ack | 07 bel |
| 08 bs  | 09 ht  | 0a nl  | 0b vt  | 0c np  | 0d cr  | 0e so  | 0f si  |
| 10 dle | 11 dc1 | 12 dc2 | 13 dc3 | 14 dc4 | 15 nak | 16 syn | 17 etb |
| 18 can | 19 em  | 1a sub | 1b esc | 1c fs  | 1d gs  | 1e rs  | 1f us  |
| 20 sp  | 21 !   | 22 "   | 23 #   | 24 $   | 25 %   | 26 &   | 27 ’   |
| 28 (   | 29 )   | 2a *   | 2b +   | 2c ,   | 2d -   | 2e .   | 2f /   |
| 30 0   | 31 1   | 32 2   | 33 3   | 34 4   | 35 5   | 36 6   | 37 7   |
| 38 8   | 39 9   | 3a :   | 3b ;   | 3c <   | 3d =   | 3e >   | 3f ?   |
| 40 @   | 41 A   | 42 B   | 43 C   | 44 D   | 45 E   | 46 F   | 47 G   |
| 48 H   | 49 I   | 4a J   | 4b K   | 4c L   | 4d M   | 4e N   | 4f O   |
| 50 P   | 51 Q   | 52 R   | 53 S   | 54 T   | 55 U   | 56 V   | 57 W   |
| 58 X   | 59 Y   | 5a Z   | 5b [   | 5c \   | 5d ]   | 5e ^   | 5f _   |
| 60 ‘   | 61 a   | 62 b   | 63 c   | 64 d   | 65 e   | 66 f   | 67 g   |
| 68 h   | 69 i   | 6a j   | 6b k   | 6c l   | 6d m   | 6e n   | 6f o   |
| 70 p   | 71 q   | 72 r   | 73 s   | 74 t   | 75 u   | 76 v   | 77 w   |
| 78 x   | 79 y   | 7a z   | 7b {   | 7c |   | 7d }   | 7e ~   | 7f del |
```

```
decimal:
|  0 nul |  1 soh |  2 stx |  3 etx |  4 eot |  5 enq |  6 ack |  7 bel |
|  8 bs  |  9 ht  | 10 nl  | 11 vt  | 12 np  | 13 cr  | 14 so  | 15 si  |
| 16 dle | 17 dc1 | 18 dc2 | 19 dc3 | 20 dc4 | 21 nak | 22 syn | 23 etb |
| 24 can | 25 em  | 26 sub | 27 esc | 28 fs  | 29 gs  | 30 rs  | 31 us  |
| 32 sp  | 33 !   | 34 "   | 35 #   | 36 $   | 37 %   | 38 &   | 39 ’   |
| 40 (   | 41 )   | 42 *   | 43 +   | 44 ,   | 45 -   | 46 .   | 47 /   |
| 48 0   | 49 1   | 50 2   | 51 3   | 52 4   | 53 5   | 54 6   | 55 7   |
| 56 8   | 57 9   | 58 :   | 59 ;   | 60 <   | 61 =   | 62 >   | 63 ?   |
| 64 @   | 65 A   | 66 B   | 67 C   | 68 D   | 69 E   | 70 F   | 71 G   |
| 72 H   | 73 I   | 74 J   | 75 K   | 76 L   | 77 M   | 78 N   | 79 O   |
| 80 P   | 81 Q   | 82 R   | 83 S   | 84 T   | 85 U   | 86 V   | 87 W   |
| 88 X   | 89 Y   | 90 Z   | 91 [   | 92 \   | 93 ]   | 94 ^   | 95 _   |
| 96 ‘   | 97 a   | 98 b   | 99 c   |100 d   |101 e   |102 f   |103 g   |
|104 h   |105 i   |106 j   |107 k   |108 l   |109 m   |110 n   |111 o   |
|112 p   |113 q   |114 r   |115 s   |116 t   |117 u   |118 v   |119 w   |
|120 x   |121 y   |122 z   |123 {   |124 |   |125 }   |126 ~   |127 del |
```

Appendix C : PostScript Level 2 Tags

Theses tags are provided in order to control exactly the PostScript Level 2 operations that should occur for a given profile. These tags are only valid for PostScript Level 2 (and conceivably future versions of PostScript) devices, and are not generally supported in PostScript Level 1 devices. In addition, some of the tags may correspond to PostScript operations that are not supported in all PostScript Level 2 devices. Using such tags requires first checking for the available operators. All operators described in the PostScript Language Reference Manual, second edition, are available on all PostScript Level 2 devices. Documentation for extensions to PostScript Level 2 are available through Adobe’s Developer Support Organization. In addition, guidelines for PostScript compatibility with this profile format are available. For details of such operator support, compatibility guidelines, the PostScript Level 2 device independent color model, or other PostScript-related issues contact Adobe's Developer Support Organization.

In general, there is a straightforward relationship between the profile's header fields and tags, and these PostScript tags. It is anticipated that the various CMSs that support this profile format will also provide support for these optional PostScript tags. To verify such support contact the CMS vendors directly. In cases where such support is provided, and the desired model of operations is the same for PostScript processing as it is for CMS processing, these tags can be omitted, since all necessary information is in the profile itself. In the case where such CMS support is in question or processing different than that provided by an arbitrary CMS is desired, these tags can be populated to provide exact control over the PostScript processing. For example, if private tags are used in the profile to achieve a non-public type of processing on certain CMSs, such processing can be achieved on a PostScript device by populating the appropriate PostScript tags.

Some of the PostScript tags have a tag type of textType or uInt8Type. This choice is provided in order to match the properties of the communicationschannel to the data in these tags. Encoding the data in uInt8Type form is recommended to save memory and/or reduce transmission times. Applications and drivers may convert it to ASCII Coded PostScript, Binary Coded PostScript, or Token Binary Coded PostScript or leave it in binary format to match the requirements of the communications channel. Applications and drivers are responsible for this potential conversion from binary data to channel compatible data. The data should be encoded in textType in those cases where the amount of data is relatively small or where the conversion from binary to channel compatible data is not available.

The PostScript contained in these tags is not self evaluating - it simply provides operands. These operands must be followed by operators like `setcolorspace`, `setcolorrendering`, and `findcolorrendering`.

Appendix D : Profile Connection Space Explanation

Introduction

This Appendix is intended to clarify certain issues of interpretation in the International Color Consortium Profile Format.

The goal of color management is to provide the capability of maintaining control over color rendering among various devices and media that may be interconnected through a computer system or network. To achieve this goal, the color characteristics of each device are determined and encapsulated in a device profile, which is a digital representation of the relation between device coordinates and a device-independent specification of color.

By *device coordinates* we mean the numerical quantities through which a computer system communicates with a color peripheral-such as the digital code values used to drive a monitor or printer, or the digital signals received from a scanner. These quantities are usually labeled RGB (or CMYK), but the labels identify the channels of the device rather than specific visual colors; the quantities are often encoded as unsigned 8-bit integers for each channel in the typical digital interface.

The *device-independent specification* is best given in a color space based on human visual experience. Thus, a device profile provides a means of translating (or transforming) color image data from device coordinates into a visual color space or vice versa.

Furthermore, if the various profiles available to a color-management system are referenced to the same visual color space, the system can translate data from one device's coordinates to another's-while maintaining color consistency-by (conceptually) passing through the intermediary of the visual color space; the latter, then, constitutes a standard interface for color communication, allowing profiles to be connected together in a meaningful sequence. A color space used in this way may be termed a Profile Connection Space (PCS). For example, the transformation of a color image from a scanner into monitor coordinates can be described as a transformation into the PCS (via the scanner's device profile) followed by a transformation out of the PCS (via the monitor's device profile). In practice, these successive transformations may be implemented in a variety of ways, and the image may never actually be represented in the PCS on disk or in computer memory. Thus, the PCS is to be regarded as a convenient reference for the definition of profiles-as an intermediate, or virtual, stage of the image processing-, in contrast to an interchange or exchange color space, which is an encoding for the storage and transmission of color images. The issues regarding the choice or design of a PCS are somewhat different from those related to an interchange space; this Appendix is concerned only with PCS issues.

A PCS consists of a coordinate system for color space and an interpretation of the data represented in that coordinate system. In fact, multiple coordinate systems can easily be supported in the same or different color-management systems, as long as they share a common interpretation, since it is usually a well-defined and relatively simple mathematical task to transform from one coordinate system to another. However, if the interpretation of the represented colors is different, there may be no satisfactory way of translating the data from one to another.

The purpose of this paper is to present an unambiguous interpretation for the PCS implicit in the International Color Consortium Profile Format. It is especially important in the heterogeneous environments currently found on desktop platforms and networks to establish this interpretation in an open, non-proprietary specification, so that different color-management systems can communicate with each other and exchange profiles within and across platforms and operating systems.

Colorimetry and Its Interpretation

The issue of interpretation has received little attention in the recent past, because it has been widely believed that the choice of a suitable coordinate system-preferably one founded on CIE colorimetry, a system of measurement and quantification of visual color stimuli created and promoted by the Commission Internationale de l'Eclairage-would suffice to guarantee device independence. The notion was that colorimetric matching of the renderings on various media was the key to satisfactory color reproduction, and that interpretation was not needed. However, although colorimetry can be an essential element of a successful approach to color management, it is usually necessary to modify thecolorimetric specification for renderings on different media. Different media require different physical color stimuli, in certain cases, because they will be viewed in different environments-e.g., different surroundconditions or illuminants; the observers, therefore, will experience differentadaptive effects. In order to preserve the same color appearance in these different environments, the colorimetry must be corrected to compensate for the adaptation of the human visual system and for physical differences in the viewing environments, such as flare. Although color appearance is still an active research topic, the most common forms of adaptation are understood reasonably well, so that the required corrections in the colorimetry for different viewing conditions can be modeled with sufficient accuracy.

There are other reasons why the colorimetry may be altered for specific media. For instance, hard-copy media-even those intended for the same viewing conditions-differ considerably in their dynamic range and color gamut. A well-crafted rendering of an image on a specific medium will take advantage of the capabilities of that medium without creating objectionable artifacts imposed by its limitations. For instance, the tone reproduction of the image should provide sufficient contrast in the midtones without producing blocked-up shadows or washed-out highlights. The detailed shape of the tone curve will depend on the minimum and maximum densities (Dmin and Dmax) attainable in the medium. Clearly, there is considerable art involved in shaping the tone-reproduction and color-reproduction characteristics of different media, and much of this art is based on subjective, aesthetic judgments. As a result, the substrate (paper, transparency material, etc.) and the colorants used in a medium will be exploited to impart a particular "personality" to the reproduction that is characteristic of the medium.

Furthermore, the desired behavior of a color-management system depends strongly on artistic intent. If the output medium is identical to the input medium-say, 35-mm slides-, the desired behavior is typically to create a duplicate of the original. But if the two media are different, it is not so obvious what the default behavior should be. In some cases, the intent may be to retain all or part of the personality of the original; in other cases, it may be more important to remove the personality of the original and replace itwith a fresh rendering that has the full personality of the output medium. Sometimes the simulation of a third medium may be important-as when an image is displayed on a monitor to preview a rendering on a dye-diffusion printer, retaining (as well as possible) the personality of an original image scanned from a photographic print! It is essential to the success of color-management systems that a broad range of options be kept open. The interpretation of the PCS merely defines the particular default behavior that will be facilitated bythe system without explicit intervention by the application or user. Alternative behaviors are not excluded by this choice; they simply will not be the default and will require more work.

With this context in mind, we present the following interpretation:

- The PCS represents desired color appearances.

Here, the term *desired* is used to indicate that the interpretation is oriented towards colors to be produced on an output medium. It also is used to imply that these colors are not restricted by the limitations of any particular output medium. It is helpful here to conceptualize a "reference reproduction medium", with a large gamut and dynamic range, as the target medium for the desired colors. Consequently, it is the responsibility of the output device profiles to clip or compress these colors into the gamut of the actual output media. And, of course, "desired" also implies the expression of artistic intent.

The term *color appearance* is used to imply that adaptive effects are taken into account. Associated with the reference reproduction medium is a "reference viewing environment". More precisely, therefore, the PCS represents the "desired color appearances" in terms of *the CIE colorimetry of the colors to be rendered on the reference medium and viewed in the reference environment*. Output profiles for media that are viewed in different environments are responsible for modifying the colorimetry to account for the differences in the observer’s state of adaptation (and any substantial differences in flare light present in these environments), so that color appearance is preserved. Similarly, input profiles are responsible for modifying the colorimetry of the input media to account for adaptation and flare; they also have the responsibility to account for the artistic intent implicit in the word "desired".

We define the *reference reproduction medium* as an idealized print, to be viewed in reflection, on a "paper" that is a perfect, non-selective diffuser (i.e., Dmin = 0), with colorants having a large dynamic range and color gamut.We define the *reference viewing environment* to be the standard viewing booth (ANSI PH-2.30); in particular, it is characterized by a "normal" surround-i.e., where the illumination of the image is similar to the illumination of the rest of the environment-, and the adapting illuminant is specified to have the chromaticity of D50 (a particular daylight illuminant).

Color Measurements

The PCS, so interpreted, represents colors for a hypothetical reference medium; device profiles must relate these colors to those that can be measuredon real media. For consistency of results, these measurements must be made in accordance with the principles of CIE colorimetry.

For one particular class of media-namely, those intended for the graphic arts-, the colorimetry should conform to graphic-arts standards for color measurement.*1* Here, the illuminant is specified to be D50, so that no corrections need to be applied for chromatic adaptation. The colorimetry standard is based on a theoretical D50 illuminant, as defined by the CIE in the form of a tabulated spectral distribution. However, the fluorescent D50 simulators found in typical professional viewing booths have rather different spectral distributions, and the color stimuli produced can be noticeably different.*2* Often, better results can be obtained by basing the colorimetry on the actual, rather than the theoretical, illumination source; unfortunately, there is no standardized, practically realizable source.

For other, non-graphic-arts, media, the illuminant may be different from D50. In general, for best results, the actual illumination spectrum should be used in the color measurements. And if the chromaticity of the illuminant is different from that of D50, corrections for chromatic adaptation will be needed and will be incorporated into the device-profile transforms. This aspect of the PCS interpretation provides flexibility to the color-management system. For example, it will be possible to transform data from a medium intended for tungsten illumination to a medium intended for cool-white-fluorescent: the input profile handles the adaptation from tungstento D50, and the output profile handles the adaptation from D50 to cool-white.

Since substantial flare (perhaps 2-3%) may be present in an actual viewing environment,*3* the colorimetry is defined in an ideal, flareless measurement environment; in this way, difficult telescopic color measurements in the viewing environment can be avoided, and simple contact instruments and/or controlled laboratory conditions can be used instead. (Corrections should be applied to the data for any appreciable flare in the actual measurement environment and instruments.)

Colorimetry Corrections and Adjustments in Output Profiles

The implications of this interpretation should be emphasized: the creator of a profile is obliged to correct and adjust the PCS data for various effects. Since the PCS is interpreted with an output orientation, we will first examinethe nature of these corrections and adjustments for output profiles. Then, in the next section, we will discuss the consequences for input profiles.

Let us look at a number of possible output paths:

Output to reflection print media: Included here are computer-driven printers, off-press proofing systems, offset presses, gravure printing, photographic prints, etc. These are generally intended for "normal" viewing environments; but corrections may be needed-e.g., for chromatic adaptation, if the illuminant's chromaticity is other than that of D50.

In the simplest scenario, the user desires to reproduce colors colorimetrically (aside from adaptive corrections) so as to attain an appearance match. A distinction can be made between "absolute" and "relative" colorimetry in this context. Absolute colorimetry coincides with the CIE system: color stimuli are referenced to a perfectly reflecting diffuser. All reflection print media have a reflectance less than 1.0 and cannot reproduce densities less than their particular Dmin. In a cross-rendering task, the choice of absolute colorimetry leads to a close appearance match over most of the tonal range, but, if the Dmin of the input medium is different from that of the output medium, the areas of the image that are left blank will be different. This circumstance has led to the use of relative colorimetry, in which the color stimuli are referenced to the paper (or other substrate). Thischoice leads to a cross-rendering style in which the output image may be lighter or darker overall than the input image, but the blank areas will coincide. Both capabilities must be supported, since there are users in both camps. However, the default chosen for International Color Consortium profiles is relative colorimetry.

This can be made more precise: the default "colorimetric" transform will effectively apply a scaling operation in the CIE 1931 XYZ color space:

Xout = (Xpaper / XD50) X (EQ 1)
Yout = (Ypaper / YD50) Y (EQ 1)
Zout = (Zpaper / ZD50) Z (EQ 3)

where XYZ are the coordinates of a color in the PCS, (XYZ) out are the coordinates of the corresponding color to be produced on the output medium, (XYZ)D50 are the coordinates of the lightest neutral represented in the PCS (namely, one with the chromaticity of D50 and a luminance of 1.0), and (XYZ) paper are the coordinates of the output paper (or other substrate) adapted to the PCS illuminant (D50). Thus, the lightest neutral in the PCS will be rendered as blank paper-regardless of the reflectance or color cast ofthe paper-; other neutrals and colors will be rescaled proportionately and will be rendered darker than the paper. Output on different reflection print media will then agree with the PCS and with each other in relative colorimetry and, therefore, in relative appearance.

In other cases, the preference may be for absolute colorimetry. This means that, within the limitations of the output medium, the CIE colorimetry of the output image should agree with values represented in the PCS. I.e., Xout = X, Yout = Y, and Zout = Z. One way of achieving this result is to apply a separate transformation to the PCS values, outside of the device profile (e.g., in application or system software):

X' = (XD50 / Xpaper) X (EQ 4)
Y' = (YD50 / Ypaper) Y (EQ 5)
Z' = (ZD50 / Zpaper) Z (EQ 6)

The relative values, X' Y' Z', can then be processed through the default colorimetric transform (i.e., they are effectively substituted for XYZ in Equations 1-3) to achieve the desired result.

This capability depends on the availability to the color-management software of the colorimetry of the paper. The "medium white point" tag in the profile can be used for this purpose and should represent the adapted, absolute colorimetry of the lightest neutral that the device and/or medium can render (usually the blank substrate).

In either case, it may happen that the dynamic range and/or color gamut of the output medium is not sufficient to encompass all the colors encoded in the PCS. Some form of clipping will then occur-in the highlights, in the shadows, or in the most saturated colors. While an appearance match may be achieved over much of color space, there will be a loss of detail in some regions. If this is objectionable, the operator should have an option for selecting a more explicit form of gamut compression to be applied to the colors as part of the output profile. International Color Consortium profiles support two styles of controlled gamut compression-"photographic" and "saturation-preserving"-in addition to the "colorimetric" option, which clips abruptly at the gamut boundary. (An important case requiring explicit gamut compression is that of input from a transparency, where the dynamic range, even of the corrected colors, may exceed that of any reflection print medium.) Note that an explicit compression maps colors from the dynamic range and gamut of the reference medium to the range and gamut of the actual medium, so that only (XYZ)D50-i.e., the lightest PCS neutral-will be rendered as blank paper, just as in the relative-colorimetric case. This time, however, the entire tone scale may be readjusted, to keep the shadows from blocking up and to maintain proper midtones, and some in-gamut colors may be adjusted to make room for out-of-gamut colors.

Output to transparency media: This category might include overhead transparencies and large-format color-reversal media, as well as slide-production systems. Transparency materials are normally intended to be viewed by projection (using a tungsten lamp) in a dim or darkened room; in some cases, however, they are placed on a back-lit viewer for display, and in others they are used as a graphic-arts input medium, in which case they are examined on a light box or light table with the aid of a loupe. Accordingly, there are several possible viewing conditions for transparencies, requiring somewhat different corrections.

Typical color-reversal films have a much larger dynamic range than reflection media and higher midscale contrast. Their tone-reproduction characteristics have evolved empirically, but it may be plausible to explain them as partially compensating for dark-surround adaptation and the flare conditions typical in a projection room. The state of brightness adaptation in a projection room is also different from that in a reflection environment. To the extent that these explanations are valid, the colorimetry should be corrected for these effects. Furthermore, in some of these environments the visual system is partially adapted to a tungsten source, and chromatic corrections should be applied for the difference between tungsten and D50.

A "colorimetric" rendering, in this case, will actually produce an appearance match to the colors in the PCS, rather than a colorimetric match-i.e., the colors measured on the resulting transparency will differ from those encoded in the PCS, but will appear the same when the transparency is viewed in its intended environment as the PCS colors would if rendered on the reference medium and viewed in reflection.

Note that the lightest neutral, (XYZ)D50,will be rendered at or near Dmin of the transparency in the default (relative) colorimetric transform. An absolute-colorimetric rendering can be generated in software, as described above for reflection-print media.

Explicit gamut compression can be provided as an option; it normally would notbe needed for images input from photographic media, but it may be useful for input from computer graphics, since some of the highly saturated colors available on a computer color monitor fall outside the gamut of transparency media.

Negative media: Here the target colors are those of a reflection print to be made from the negative. No adaptive corrections are required, unless the printis intended to be viewed under an illuminant other than D50. Explicit gamut compression is a useful option, and both relative and absolute colorimetric matches can be provided as in the case of direct-print media.

Monitor display: The viewing conditions of a CRT monitor may require some corrections to the colorimetry, due to the effects of surround and flare. Also, if the monitor's white point is other than D50, chromatic adaptation must be accounted for. When corrections for these effects are applied, the colors in the display should match the appearance of those in the PCS and should provide accurate and useful feedback to the operator.

In most cases, the rendering should be "colorimetric" (possibly including adaptive corrections), in order to achieve this result. (As for reflection print media, this would be "relative" by default, but "absolute" colorimetry is also supported.) In other cases (video production, perhaps), it may be moreimportant to the user to create a pleasing image on the monitor (without having out-of-gamut colors block up, for instance) than to preserve an appearance match to the PCS; for that purpose, explicit gamut compression would be a useful option.

In many scenarios, the monitor display is not the end product, but rather a tool for an operator to use in controlling the processing of images for other renderings. For this purpose, it will be possible to simulate on the monitor the colors that would be obtained on various other output media. The PCS colors are first transformed into the output-device coordinates, using any preferred style of gamut compression. Then they are transformed back to the PCS by using the (colorimetric) inverse output transform. (These two steps can be replaced by an equivalent "preview" transform.) Finally they are transformed (colorimetrically) into monitor coordinates for previewing. The result of compression to the output gamut should then be visible in the displayed image.

Colorimetry Corrections and Adjustments in Input Profiles

The purpose of an input profile is to transform an image into the PCS-i.e., tospecify the colors that are desired in the output. Since there are many possible intentions that a user might have for these colors, we cannot impose many restrictions on the nature of the transforms involved. Bearing in mind the capabilities of the output profiles, as just outlined, we can suggest the possibilities available to various classes of input profiles.

Scanned reflection prints: Here the intended viewing environment may be identical to the reference, but, if not, adaptive corrections should be applied to the colorimetry. In the simplest case, the profile may consist of atransformation from scanner signals to the colorimetry of the medium. In this case, the personality of the input medium has been preserved. If the output rendering is also "colorimetric", the result will be an appearance match to the original. Indeed, if the output medium is the same as the input medium, the result should be a close facsimile or duplicate of the original.

By default, the rendering is based on relative colorimetry, as discussed above. Therefore, it should be remembered, when creating an input profile, that the (XYZ)D50 point of the PCS will be mapped to the Dmin of the output medium. This implies that the Dmin of the input medium must be mapped to the (XYZ)D50 point of the PCS, in order to facilitate the duplication of an original and a relative-colorimetry match when cross-rendering.

In order to enable the alternative of absolute colorimetry, the "white point" field in the header of the input profile should be used to specify the colorimetry of the paper. This allows the absolute colorimetry of the originalto be computed from relative colorimetry represented in the PCS, by analogy toEquations 1-3 above. These absolute color stimuli can then be converted to relative colorimetry for output by using the "white point" field of the output profile in Equations 4-6.

There are other possibilities, however. The input profile could be designed to remove some or all of the personality of the input medium, so that the PCS encoding makes use of more of the gamut and dynamic range of the reference medium. In these cases, it will probably be best to choose some form of explicit gamut compression in the output profile. The result may differ in appearance considerably from the original and will constitute a fresh rendering tuned to the capabilities and limitations of the output medium.

In any case, a calibrated color monitor, if available, can be used to display an accurate preview of the result.

Scanned transparencies: Since transparencies are intended for viewing in a variety of environments, different kinds of adaptive corrections may be applied to the colorimetry of the input medium to obtain colors in the PCS. For instance, the device profile might transform scanner signals into the colorimetry of a reference print that would have the same appearance in the reference environment as the transparency produces in a projection environment. (Note that there may be no actual reflection print medium that has sufficient dynamic range to reproduce all of these color appearances). In this scenario, the personality of the color-reversal film or other transparency material is retained, even though the colorimetry has been modified for the PCS; still, this may be loosely termed a "colorimetric" transform, since the only corrections are for flare and adaptation.

As in the case of input prints, there are other possibilities: some or all of the personality of the input medium can be removed, according to artistic intent, yielding different results, which also depend on the style of gamut compression selected for output.Normally, the Dmin of the input medium should be mapped to (XYZ)D50 in the PCS. The absolute, adapted XYZ of the Dmin color is recorded in the "medium white point" tag.

Scanned negatives: Photographic negatives, of course, are not intended for direct viewing. Therefore, the colorimetry that is relevant here might be thatof a hypothetical reflection print made from the negative and intended for viewing in the reference environment. No adaptive corrections should be applied. The personality of the result is that of the negative-positive systemas a whole. Again, other possibilities exist, depending on artistic intent.

Computer graphics: Such imagery is usually synthesized in the RGB space of a display monitor that provides visual feedback to the operator. Thus, adaptive corrections may need to be applied to the colorimetry of the monitor to definethe colorimetry of a reference print having the same appearance. The personality here is that of the synthetic image on the monitor screen.

Scene capture: This pathway refers to video cameras, electronic still cameras, and other technologies (such as Photo CD(tm)) that provide a capability of approximately determining the colorimetry of objects in a real-world scene. Inmost cases, the tone scale must be adjusted to provide enough contrast for viewing the reference medium in the reference environment; the colorfulness of the image should also be enhanced somewhat for that environment. The personality of the result, of course, depends on the nature of these adjustments.

Colorimetric input: In some cases, input colors are specified that are intended to be processed colorimetrically, without any tone shaping or chromatic enhancement. This might be the case, for instance, when a scene-capture device is used to record the colorimetry of real-world objects for scientific reasons, rather than for creating a pleasing reproduction. It may also be the case when particular spot colors are specified in colorimetricterms. In these cases, the specified colorimetric values are left intact in the transformation to the PCS; no adaptive corrections or adjustments are applied. The PCS values should be represented in relative colorimetry, and the"white point" tag specifies the reference point for the scaling. In some cases this reference point will have a luminance of 1.0, and there will be no difference between relative and absolute colorimetry. In other cases the reference point will have the colorimetry of (say) the paper stock used in a spot-color sample book or of a particular light neutral in a scene. In most ofthese cases, the preferred output rendering will also be "colorimetric". By default, as before, this will entail relative colorimetry; absolute colorimetry can be achieved, outside of the default transforms, by taking account of the "white point" tags of the input and output profiles and converting appropriately.

An image of this kind can be said to have no personality.

As can be inferred from some of these examples, the user may have a choice of input profiles having different intents, as well as a choice among output transforms having different intents. The end result depends on both of these choices, which, for the most predictable color reproduction, should be made in coordination. To aid in this coordination, there are profile tags that specify the rendering intent and that distinguish between input transforms that are colorimetric (aside from possible corrections for flare and adaptation) and those that have applied adjustments to the colorimetry.

Techniques for Colorimetry Corrections

As we have seen, if the viewing conditions of the medium are different from the reference (e.g., projected slides or video viewed in dim or dark surround), corrections to the colorimetry of the reproduction should be applied.[*4*] These should be designed to correct for differences in the flare light present in these environments, as well as the effects of non-normal surround, brightness adaptation to the absolute radiant flux of the illumination, and any other effects that are found to be significant. And if the medium was intended to be viewed under an illuminant of different chromaticity than that of D50, the profile should incorporate corrections for chromatic adaptation; these can simply be based on a linear scaling in XYZ (which happens automatically in the CIELAB system); alternatively, it can be based on the linear Von Kries transformation[*5*] (or, if preferred, a more sophisticated, nonlinear model of color appearance, such as that of Hunt[*6*] or Nayatani[*7*]).

If the creators of device profiles universally apply these corrections to their colorimetric data, the PCS will have a universal, unambiguous interpretation, and images rendered "colorimetrically" will evoke (as nearly as possible) the same appearance, regardless of the medium and viewing environment of the reproduction. In this way, the same image can be rendered on photographic transparency material, various reflective print media, CRT’s, etc., and will, by and large, appear similar to the viewer. This goal cannot be achieved simply by matching the colorimetry of the reproductions. Various forms of explicit gamut compression and input effects can be made available for situations where other goals are important; the recommended PCS interpretation does not limit these possibilities in any way: it merely facilitates the default behavior of the color-management system.

---

[1]IT8.7/3, "Graphic technology--Input data for characterization of 4-color process printing", draft standard of Subcommittee 4 (Color) of ANSI Committee IT8 (Digital Data Exchange Standards), 14 December 1992, Paragraph 4.2.

[2]D. Walker, "The Effects of Illuminant Spectra on Desktop Color Reproduction", in *Device-Independent Color Imaging,* R. Motta and H. Berberian, ed., Proc. SPIE, **1909**, 1993, pp. 236-246.

[3]R.W.G. Hunt, *The Reproduction of Colour*, Fourth Edition, Fountain Press, 1987, pp. 52-53.

[4]Hunt, *op. cit*., pp. 56-61.

[5]R.W.G. Hunt, *Measuring Color*, Ellis Horwood, pp. 70-71.

[6]*Ibid.*, pp. 146-173.

[7]Y. Nayatani, K. Takahama, and H. Sobagaki, "Prediction of color appearance under various adapting conditions", *Color Res. Appl.*, **11**, 62 (1986).

Appendix E: References

ANSI

ANSI PH2.30-1989, American National Standards Institute, 11 West 42nd Street, New York, New York, 10036.

CGATS and IT8

CGATS and IT8 documents are available from NPES The Association for Suppliers of Printing and Publishing Technologies, 1899 Preston White Drive, Reston, VA 22091-4367.

Knuth MetaFonts

MetaFont:The Program (Computers & Typesetting ; D), Donald Knuth, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1986

Macintosh Localization References

Guide to Macintosh Software Localization, Apple Computer, Inc., Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1992 and *Inside Macintosh*, Text, Apple Computer, Inc.,Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1993.

PostScript Level 2

*PostScript*TM. *Language Reference Manual*, Second Edition, Adobe Systems Inc., Second Edition,

Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1990.

Unicode

The Unicode Standard, Worldwide Character Encoding, The Unicode Consortium, Version 1.0, Volumes 1 & 2, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1990.

Index