Adobe® PostScript® 3™

DeviceN Color Space and Color Model

Adobe® Developers Association

9 October 1997

Technical Note #5604
LanguageLevel 3

Adobe Systems Incorporated

Corporate Headquarters
345 Park Avenue
San Jose, CA 95110-2704
(408) 536-6000

Eastern Regional Office
24 New England
Executive Park
Burlington, MA 01803
(617) 273-2120

Adobe Systems Europe Limited
Adobe House, Mid New Cultins
Edinburgh EH11 4DU
Scotland, United Kingdom
+44-131-453-2211

Adobe Systems Japan
Yebisu Garden Place Tower
4-20-3 Ebisu, Shibuya-ku
Tokyo 150 Japan
+81-3-5423-8100

PN LPS5604

Contents

Examples

Preface

This Document

This is the original release for *DeviceN Color Space and Color Model,* a document that provides a detailed description of the LanguageLevel 3 extension to device-dependent color spaces and color models. This new color feature of the PostScript® language, called **DeviceN**, enables a developer to specify and render more than four device-dependent colors.

Intended Audience

This document is written for software developers who are interested in learning about the DeviceN color space or color model.

It is assumed that the developer is already familiar with how color specification and color rendering work.

Organization of This Document

Section 1, “DeviceN Color,” gives an overview of multi-component and HiFi color and their relationship to **DeviceN** color. This section also gives overviews of **DeviceN** color for both color specification and color rendering.

Section 2, “Implementing the DeviceN Color Space,” specifies the PostScript language changes to color specification for the **DeviceN** color space. Some examples of using the **DeviceN** color space are also included. Changes to the **Indexed** color space and filters that consider color are also discussed. Finally, the benefits of using the **DeviceN** color space are also listed.

Section 3, “DeviceN Color Model,” discusses changes to the page device dictionary for the new **DeviceN** color model and information about device-dependent color conversions and transfer functions.

Section 4, “Tips and Techniques,” gives some tips and techniques for working with **DeviceN** color in general and for particular application needs.

Related Publications

Supplement: PostScript Language Reference Manual (LanguageLevel 3 Specification and Adobe PostScript 3™ Version 3010 Supplement), available from the Adobe® Developers Association, describes the formal extensions to the PostScript language that have occurred since the publication of the PostScript Language Reference Manual, Second Edition. This supplement also includes all LanguageLevel 3 extensions available in version 3010.

PostScript Language Reference Manual, Second Edition (Reading, MA: Addison-Wesley, 1991) is the developer's reference manual for the PostScript language. It describes the syntax and semantics of the language, the imaging model, and the effects of the graphical operators.

Statement of Liability

DeviceN Color Space and Color Model

1 DeviceN Color

1.1 Overview of Multi-Component and HiFi Color

This document describes the extensions to the PostScript language to support general, device-dependent, *multi-component* color specification and rendering. This includes the new color space and device color model called **DeviceN**.

Multi-component color is any generic color space representing *N* color components, where *N* is greater than or equal to 1. The LanguageLevel 3, device-dependent instance of a multi-component color space is **DeviceN**. This type of color space allows for the specification of color components other than the standard set of three (RGB) or four (CMYK) color components usually used by most applications. Color rendering is achieved on a PostScript output device capable of handling multi-component output by setting **ProcessColorModel** to **DeviceN**.

HiFi color is an instance (or subset) of the multi-component color spaces, where *N* is generally greater than or equal to 5, and where the *N* inks are selected to increase the gamut of the printing device. This allows for the specification of more than the standard set of three or four color components normally used by most applications. HiFi color can be specified in the PostScript language using the **DeviceN** color space. Color rendering is achieved on any output device capable of multi-component output.

The range (gamut) of colors available on most standard (three- and four-color) devices is generally smaller than the range of colors that the human eye can see. Traditional four-color printing can only reproduce a low percentage of colors visible to the human eye. Previously, in order to extend the color gamut of a typical printing device, spot colors were used.

On devices that support multi-component output, HiFi color is a method for printing colors that would normally lie outside the gamut of three-color and four-color devices. An example of a Hifi color system is PANTONE® Hexachrome™, which uses six inks or colorants: CMYK plus orange and green. Many other HiFi color systems use CMYK process inks with additional process inks to form the extra colorants in their HiFi process color model. The CMYK inks used in HiFi color can be different from the traditional CMYK inks.

HiFi color has usually only been thought of in terms of extending the gamut of process printing on a press, not on more general printing devices that might include the broader range of PostScript printers. This is because prior to LanguageLevel 3, the PostScript language, and other PDLs, could describe colors only to RGB and CMYK devices.

In applications and PostScript interpreters that support LanguageLevel 3 extensions, multi-component color spaces (including HiFi) must be supported for both color specification and color rendering. Color specification is necessary because applications may transform data into multi-component color spaces prior to printing. Additionally, color rendering is necessary because there are advantages to performing the color transformations in the PostScript interpreter.

Any of these colors spaces – multi-component, HiFi, **DeviceN** – may be used as input color spaces in LanguageLevel 3. If the components of the input color space are different than the colorants supported for output by the device, the device will perform internal color conversions during rendering, as described in Section 3.2 of this document and also in the *Supplement: PostScript Language Reference Manual*.

1.2 Overview of the DeviceN Color Space

LanguageLevel 3 provides support for the composite specification of multi-component color, which includes HiFi color, through a new device-dependent color space called **DeviceN**.

The **DeviceN** color space allows developers to specify other than the standard three or four colorants through the **setcolorspace** operator. Developers can now specify any arbitrary number of colorants, independent of whether or not the target PostScript printer supports them. The inputs to the **setcolorspace** operator for **DeviceN** include an alternate color space in the event that the target PostScript printer does not support one or more of the colorants specified by the **DeviceN** color space. That is, devices that support only a subset (or none) of the specified colors in the **DeviceN** color space will simply revert to a backup color space, similar to the way that the **Separation** color space is defined.

Note *The alternative color space does not get invoked if the target device does not support the **DeviceN** mechanism.*

Note *The **DeviceN** color space is required in LanguageLevel 3 and is supported on all PostScript 3 printers.*

1.3 Overview of the DeviceN Color Model

The **ProcessColorModel** key has also been extended in LanguageLevel 3 to accept **DeviceN** as a color model for rendering on PostScript printers capable of multi-component output (that is, **DeviceN** can be a specific native color space for a device). PostScript printers that support other than the standard number of colorants (have other than three or four base inks to create color with), such as HiFi color printers that support the PANTONE Hexachrome system, can take full advantage of the **DeviceN** color model to more directly produce a larger gamut of colors.

Originally the market for multi-component color, and HiFi color specifically, was limited to high-end print shops that created four-color brochures, magazines and other documents that require high-quality or photorealistic color rendering. There was no standard method for application developers using desktop printers to take advantage of these features. With the **DeviceN** color model available in PostScript printers capable of multi-component output, applications developers and developers of desktop printers can easily take advantage of multi-component color or HiFi color for the extended gamut it provides.

Support for the **DeviceN** color model is optional in PostScript 3 printers. Output devices, even HiFi devices, do not necessarily have to support the **DeviceN ProcessColorModel**. For example, a HiFi output device may declare itself to support only the **DeviceCMYK ProcessColorModel**, and perform CMYK–to–HiFi color space conversion invisibly to the user.

To put this another way, a request can be made for any PostScript 3 device to act as a **DeviceN** device. Whether or not this request succeeds depends on whether the requested *N* inks are available on the device, and whether there are conversion routines available to convert device-dependent color spaces like DeviceRGB into the *N* colorants.

Devices that are not capable of certain types of multi-component output (such as HiFi output) may support the **DeviceN ProcessColorModel** anyway. For example, a CMYK printer may allow setting the **DeviceN ProcessColorModel**, and perform internal conversions to support a simulated set of colorants for proofing purposes. Even some monochrome devices (for example, imagesetters) may allow setting the **DeviceN ProcessColorModel** while producing separations.

Note *An optional key has been added as an instance of the* ***OutputDevice*** *resource category. The name of the key is* ***DeviceN****, which is of type array; this key specifies colorants of a* ***DeviceN*** *color model supported by a device (when producing composite output). It is an array of arrays in which each subarray identifies the colorants of a* ***DeviceN*** *instance. A subarray can contain names or strings. The size of a subarray need not correspond to the number of toner stations of the device when printing. Additionally, the* ***ProcessColorModel*** *instance of the* ***OutputDevice*** *resource has been extended to support the* ***DeviceN*** *color model. Also,* ***DeviceN*** *has been added as an instance of the* ***ColorSpaceFamily*** *resource. For more information, see Table 3.4 and 3.7 of the Supplement: PostScript Language Reference Manual.*

2 Implementing the DeviceN Color Space

Section 2.1, “DeviceN Color Space Specification,” covers the PostScript implementation of the **DeviceN** color space. This section also includes some examples of using this new color space. Section 2.2, “Indexed Color Space,” covers the changes to the **Indexed** color space to support the **DeviceN** color space. Section 2.3, “Filters,” covers limitation of using filters that consider the number of color components. Finally, Section 2.4, “Benefits of Using the DeviceN Color Space,” lists some of the benefits of using the **DeviceN** color space.

2.1 DeviceN Color Space Specification

The **DeviceN** color space is specified as follows:

```
[/DeviceN [names] alternativeSpace tintTransform]
setcolorspace
```

Note *If the **names** array contains repeated names of the individual color components, a PostScript error is raised.*

Color values in the **DeviceN** color space – that is, the color component values passed to the **setcolor** operator, or image sample values – are tint components, each of which is a value in the range 0 (the minimum amount of that colorant) to 1 (the maximum amount of that colorant). The **setcolor** operator sets the current color in the graphics state to a set of tint values, the initial value for each tint being 1.

The number of color component values that must be supplied to the **setcolor** operator is determined by the number of color component names in the *names* array. Each name in the *names* array can be either a name or a string object.

When the color space is set to **DeviceN**, the interpreter determines if all of the named color components can be produced on the specific PostScript printer. If all of the named color components can be made, the *alternativeSpace* and *tintTransform* parameters are ignored. Subsequent painting operations paint the named device colorants.

If all of the named color components can not be made, subsequent painting operations are performed using the *alternativeSpace* color space. For example, since a CMYK device is unable to produce the named color component /Red, /Green, or /Blue, the *alternativeSpace* parameter will be used and the *tintTransform* procedure called to perform any tint transforms.

Note *This process can be considered an “all or nothing” situation – the presence of just one unknown color component name invokes the alternative color space and the tint transform procedure.*

The *alternativeSpace* parameter must be an array or name object that identifies the alternative color space. This can be any device-dependent or device-independent color space, excluding a special color space, such as a **Pattern** color space, an **Indexed** color space, or a **Separation** color space.

Note *The* ***alternativeSpace*** *parameter should be constructed as if it were to be used as an operand to the* ***setcolorspace*** *operator.*

The *tintTransform* procedure is a PostScript procedure that provides an *n* to *m* transformation, or mapping, from the **DeviceN** color space to the alternative color space, where *n* is the number of color components names listed in the *names* array and *m* is the number of color components needed by the alternative color space. During subsequent painting operations, if the alternative color space is being used, this procedure is used to transform *n* tint values into *m* color component values in the alternative color space.

There is only one case in which color conversion may be involved in conjunction with the DeviceN color space. This is when the alternative color space (*alternativeSpace*) has been invoked; that is, when the colors listed in the **DeviceN** color space are not supported by the current device. The **UseCIEColor** key affects only the alternative color space when the primary color space is **DeviceN**. If the alternative color space is a device color space, and if **UseCIEColor** is true, then the alternative color space will be remapped. Other than this case, there are no other color conversions involving the **DeviceN** color space.

Note *To use the* ***image*** *operator with the* ***DeviceN*** *color space, the dictionary form of the image operator must be used.*

Note *The* ***currentcolor****,* ***setcolor****,* ***currentcolorspace****, and* ***setcolorspace*** *operators have been extended to handle the* ***DeviceN*** *color space. For more information, see Chapter 8, “Operators,” of the Supplement: PostScript Language Reference Manual.*

Example 1 demonstrates the use of a **DeviceN** color space where all of the specified colorants can be reproduced on the device. In this example, the three specified color component names are Red, Green, and Blue, the tint transform procedure is empty, and the alternative color space is specified as /DeviceRGB.

Example 1 *DeviceN using RGB Colorants*

```
%!PS
%DEVICEN1.PS
/inch {72 mul} def
[% Define the DeviceN color space
    /DeviceN
    [/Red /Green /Blue]
    /DeviceRGB
    {}
] setcolorspace
% Define a local array of color values
/Colors [
    {1 0 0}
    {0 1 0}
    {0 0 1}
] def
% Define other variables as needed
%... code goes here
% Then, draw objects
%... code goes here
gsave % save graphics state
1 1 1 setcolor % color = white
    fill % fill the shape
grestore % restore graphics state
0 0 0 setcolor % color = black
0.5 inch setlinewidth % set a fat line width
stroke % stroke the shape
showpage % display the page
```

Note *Complete source code for this example accompanies this document.*

Example 2 illustrates a hypothetical color space that contains six color components, none of which are available to the PostScript printer. The tint transform procedure provides an *n* to *m* mapping of **DeviceN** color component names to color component values in the alternative color space /DeviceRGB. The Ruby, Emerald, and Sapphire colors are passed through as Red, Green, and Blue, respectively. Then, the Turquoise, Amethyst, and Citrine colors are converted to Cyan (Green and Blue), Magenta (Red and Blue), and Yellow (Red and Green), respectively. Finally, the output color values are clipped to restrict their maximum value to 1.0.

Example 2 *Imaginary Color Space Converted to RGB*

```
%!PS
%DEVICEN2.PS
/inch {72 mul} def
[% Define the color space to be DeviceN
    /DeviceN
    [/Ruby /Emerald /Sapphire /Turquoise
    /Amethyst /Citrine]
    /DeviceRGB
    {%define new values and clip to 1.0
      6 -1 roll 1 index add 2 index add
      6 -1 roll 2 index add 4 index add
      6 3 roll add add 4 -1 roll pop
      3 -1 roll dup 1 gt {pop 1} if
      3 -1 roll dup 1 gt {pop 1} if
      3 -1 roll dup 1 gt {pop 1} if
    }
] setcolorspace
% Define a local array of color values
/Colors [
    {1 0 0 0 0 0}
    {0 1 0 0 0 0}
    {0 0 1 0 0 0}
    {0 0 0 1 0 0}
    {0 0 0 0 1 0}
    {0 0 0 0 0 1}
] def
% Define some variables and draw the objects
%... more code here
gsave % save graphics state
    1 1 1 1 1 1 setcolor % color = white
    fill % fill the shape
grestore % restore graphics state
0 0 0 0 0 0 setcolor % color = black
0.5 inch setlinewidth % set a fat line width
stroke % stroke the shape
showpage % display the page
```

Note *Complete source code for this example accompanies this document.*

Example 3 illustrates a PANTONE Hexachrome color space. The tint transform procedure transforms the six values into /DeviceCMYK color space values. Orange and Green are not available on CMYK devices, so the tint transform procedure must compute the correct color values. These values were determined by experimentation and are intended for purposes of illustration only. In other cases, the values may be quite different. The tint transform procedure also clips the CMYK values to 1.0, to ensure that the color values are within range.

Example 3 *DeviceN Color Space Converted to CMYK*

```
%!PS
%DEVICEN4.PS
/inch {72 mul} def
[%Define the color space to be DeviceN
    /DeviceN
    [/Cyan /Magenta /Yellow /Black /Orange /Green]
    /DeviceCMYK
{
    6 dict begin
      %Save the color values, adjust for green,orange.
      %Clip the resulting values to 1.0
      /g exch def /o exch def /k exch def
      /y exch def /m exch def /c exch def
      /m m o 0.65 mul add def % add orange to magenta
      /y y o 0.96 mul add def % add orange to yellow
      /c c g 0.74 mul add def % add green to cyan
      /y y g 0.92 mul add def % add green to yellow
      c dup 1.0 gt {pop 1.0} if
      m dup 1.0 gt {pop 1.0} if
      y dup 1.0 gt {pop 1.0} if
      k dup 1.0 gt {pop 1.0} if
    end
}
] setcolorspace
% Add code to setup variables and draw objects
%... insert code here
gsave % save graphics state
    1 1 1 1 1 1 setcolor % color = white
    fill % fill the shape
grestore % restore graphics state
0 0 0 0 0 0 setcolor % color = black
0.5 inch setlinewidth % set a fat line width
stroke % stroke the shape
showpage % display the page
```

Note *Complete source code for this example accompanies this document.*

2.2 Indexed Color Space

The **Indexed** color space has been extended to allow the base color space to include **Separation** or **DeviceN**. This additional functionality allows the indexed mechanism to be used to apply tone-scale (for example, a duotone) adjustments to an individual color component. It also allows component data to be used for painting multiple color components.

2.3 Filters

Only the LZW and DCT filters consider the number of color components. For the LZW filters, the **Colors** entry in the filter dictionary has its upper limit removed. Because the JPEG standard is only defined for 1 to 4 color components of interleaved data, the DCT filters cannot be used for more than four color components.

2.4 Benefits of Using the DeviceN Color Space

There are several benefits in using **DeviceN** for color specification:

- The **DeviceN** color space can be used to more easily accommodate integrated process and spot colors as well as to support multi-component and HiFi color printing.

- The **DeviceN** color space allows the host computer to bundle an arbitrary number of components or colorants into a color space. This allows the printing system to manipulate the colors as a group instead of individually. The developer defines which of the colors produced by the printing system will make up the color space.

- The **DeviceN** color space provides greater control of the overprinting between the components. Formerly, to paint only the CMY device colorants of a CMYK device while leaving the K colorant unaltered, it was necessary to use the **Separation** color space once each for cyan, magenta, and yellow; a composite description of this color is not permitted using the DeviceCMYK color space without altering the black component. The **Device**N color space allows the creation of a composite description for the cyan, magenta, and yellow color components, leaving the black component unaltered.

- It is often desirable to use data for a single value component to paint multiple color components. For example, a single gray image can be used to produce a duotone. The **DeviceN** color space, used in conjunction with the modified indexed mechanism, provides this functionality.

3 DeviceN Color Model

Section 3.1, “Extensions to the Page Device Dictionary for the DeviceN Color Model,” discusses the changes to the **ProcessColorModel** and **SeparationColorNames** keys in the page device dictionary. Section 3.2, “Device-Dependent Color Conversions,” covers the use of the new key **UseCIEColor** to map device-dependent color to device-independent color. Section 3.3, “Transfer Functions,” describes the change to setting transfer functions for the **DeviceN** color space or model.

3.1 Extensions to the Page Device Dictionary for the DeviceN Color Model

The **ProcessColorModel** and the **SeparationColorNames** keys in the page device dictionary have been extended to support the **DeviceN** color model for color rendering. The **ProcessColorModel** key specifies the colorant model used for rendering process colors in a PostScript interpreter. This can be used to specify the **DeviceN** color model for multi-component or HiFi color, or for more specialized color models that are four-color or less, but do not match the standard color models.

The **SeparationColorNames** key is used to define the colorants of a device. When the value of **ProcessColorModel** is **DeviceN**, the names of the device colorants must all be given explicitly in the **SeparationColorNames** array. There are no default names for **DeviceN** as there are for the standard process color models such as DeviceRGB and DeviceCMYK.

If a device cannot resolve a combination of **DeviceN ProcessColorModel** and **SeparationColorNames** requests, it may revert to a previous state, or (for some capable printing systems) to some new state, where it can convert all color spaces. All color spaces will always be printed; however, the colorants used may be different than those requested.

Note *For more information on **ProcessColorModel** or **SeparationColorNames**, see Table 4.17 and Section 6.4 of the Supplement: PostScript Language Reference Manual.*

3.2 Device-Dependent Color Conversions

Device-dependent color spaces (other than **DeviceN**) will sometimes need CIE-based color correction or rendering intent support. This can be accomplished using the device-independent path by using the page device dictionary key **UseCIEColor**. Setting the **UseCIEColor** key to true causes an internal redefinition of some PostScript language operators; this will switch the color values in the device-dependent color space to the corresponding device-independent color values. During this color conversion, the color values are not actually changed; instead, a color space array is added that describes the conditions in which the color values were generated. The color rendering dictionary (CRD) can then make the needed adjustments to the color values.

If the **ProcessColorModel** is DeviceN, the conversion from standard PostScript color spaces (DeviceGray, DeviceRGB, DeviceCMYK, CIEBasedA, CIEBasedABC, CIEBasedDEF, and CIEBasedDEFG) to the native (multi-component) color space of the device requires the use of product-specific color conversion procedures. These color conversions are invisible to the user.

Undercolor removal and black generation on a device that supports the **DeviceN** model is implementation-specific.

For any PostScript interpreter, a PostScript print job may request that the device behave as **DeviceN** color model with a certain set of colorants, as specified in **SeparationColorNames**. A decision is then made at the execution of **setpagedevice** whether or not the device can behave as **DeviceN**. The **setpagedevice** operator will only succeed if there are custom conversion routines available to map all device-dependent color spaces to the requested **DeviceN** colorants of the device. It is not recommended that PostScript printer developers use this method.

Note *For more information on CRDs or rendering intents, see Sections 6.2 and 6.3 of the Supplement: PostScript Language Reference Manual.*

Note *For more information on* ***UseCIEColor****, see Section 6.1 of the Supplement: PostScript Language Reference Manual.*

3.3 Transfer Functions

If the **ProcessColorModel** is **DeviceN**, transfer functions are set using the **TransferFunction** key in the halftone dictionary for each named color component. The transfer functions established by **settransfer** or **setcolortransfer** are not used in this case.

Note *Transfer functions are not intended for page descriptions; rather, they are intended for describing device-specific behavior.*

Note *For more information on transfer functions, see Section 6.3 of the PostScript Language Reference Manual, Second Edition.*

4 Tips and Techniques

The DeviceN Color Space

The **DeviceN** color space provides the ability to do separations both in-RIP (in the printer PostScript interpreter) and on the host system. Applications that already support separations will still need to continue support of on-the-host techniques until more PostScript printers are available that support the LanguageLevel 3 printer systems-based separations feature. On-the-host separations will need to paint **DeviceN**-colored objects once for each named colorant in the color space.

Some parts of **DeviceN** color specification can be emulated; for example, each **DeviceN** colorant can be mapped to a **Separation** color space (or **setcustomcolor** can be used as described in Technical Note 5044), then an object can be painted multiple times, overprinting all but the first paint. This approach can be used if the effects of **DeviceN** color are desired.

The alternative space and the tint transform can also be used to create a best fit for the **DeviceN** color space.

Other parts of **DeviceN** color specification cannot be emulate, such as the enabling of multi-tone gray images that this allows (in combination with extending the **Indexed** color space to **DeviceN**).

The DeviceN Color Model

A PostScript interpreter that supports more than four colorants can do so using either a combination of a standard process color model and some spot colors or by using the **DeviceN** color model. The method of using a standard process color model and spot colors could be used as a different approach for working with more than the standard four colors (CMYK). The **ProcessColorModel** is set to a standard color model such as **DeviceRGB** or **DeviceCMYK**; then, the remaining colorants are added as spot colors. Only the spot colors have to be listed in the **SeparationColorNames** array. The other colors are part of a standard color model and do not have to be listed in the **SeparationColorNames** array.

Compare this with the use of the **DeviceN** color model, where no spot colors have to be specified, but all of the colors defining the current color model would have to be listed in the **SeparationColorNames** array.

Index